RM0090
Reference manual

STM32F405/415, STM32F407/417, STM32F427/437 and STM32F429/439 advanced Arm®-based 32-bit MCUs

Руководство

Пересказал А. Косенко
aikos55@gmail.com

1

[1] Кому не лень, проставьте ссылки оглавления сами.

Введение

Это описание памяти и периферии микроконтроллеров STM32F405xx/07xx, STM32F415xx/17xx, STM32F42xxx и STM32F43xxx для разработчиков.

Сопутствующие документы

Есть на сайте STMicroelectronics (http://www.st.com):

- STM32F40x and STM32F41x datasheets
- STM32F42x and STM32F43x datasheets
- Про Arm® Cortex®-M4 с FPU, см. *STM32F3xx/F4xxx Cortex®-M4 with FPU programming manual (PM0214)*.

1. Типографские соглашения

1.1. Сокращения для регистров

read/write (rw)	Чтение/запись
read-only (r)	Только чтение
write-only (w)	Только запись
read/clear (rc_w1)	Чтение/очистка записью 1, запись 0 не влияет.
read/clear (rc_w0)	Чтение/очистка записью 0, запись 1 не влияет.
read/clear by read (rc_r)	Чтение/очистка. Чтение сбрасывает в 0. Запись 0 не влияет.
read/set (rs)	Чтение/установка записью 1. Запись 0 не влияет.
read-only write trigger (rt_w)	Чтение. Запись 0 или 1 включает событие, но бита не изменяет.
toggle (t)	Software can only toggle this bit by writing ‘1’. Writing ‘0’ has no effect.
Reserved (Res.)	Reserved bit, must be kept at reset value.

1.2. Словарь

- Ядро CPU включает два отладочных порта:
  - JTAG debug port (JTAG-DP) с 5-контактным интерфейсом по протоколу Joint Test Action Group (JTAG).
  - SWD debug port (SWD-DP) с 2-контактным интерфейсом (такты и данные) по протоколу Serial Wire Debug (SWD).

  Протоколы JTAG и SWD есть в *Cortex®-M4 with FPU Technical Reference Manual*
- **Слово**: данные/команда 32-бит длины.
- **Полуслово**: данные/команда 16-бит длины.
- **Байт**: данные 8-бит длины.
- **Двойное слово**: данные 64-бит длины.
- **IAP** (in-application programming): это возможность перезаписи Flash памяти MCU во время работы программы.
- **ICP** (in-circuit programming): это возможность перезаписи Flash памяти MCU по протоколам JTAG, SWD или загрузчиком на плате прикладной системы.
- **I-Code**: шина команд от ядра CPU к Flash памяти для предвыборки команд.
- **D-Code**: шина данных (загрузка литералов и отладка) между CPU и Flash памятью.
- **Байты опций**: конфигурация продукта, записанная в Flash.
- **OBL**: Загрузчик байтов опций.
- **AHB**: продвинутая высокопроизводительная шин.
- **CPU**: refers to the Cortex®-M4 with FPU core.

1.3. Доступность периферии

Это есть в описаниях устройств.

2. Архитектура памяти и шины

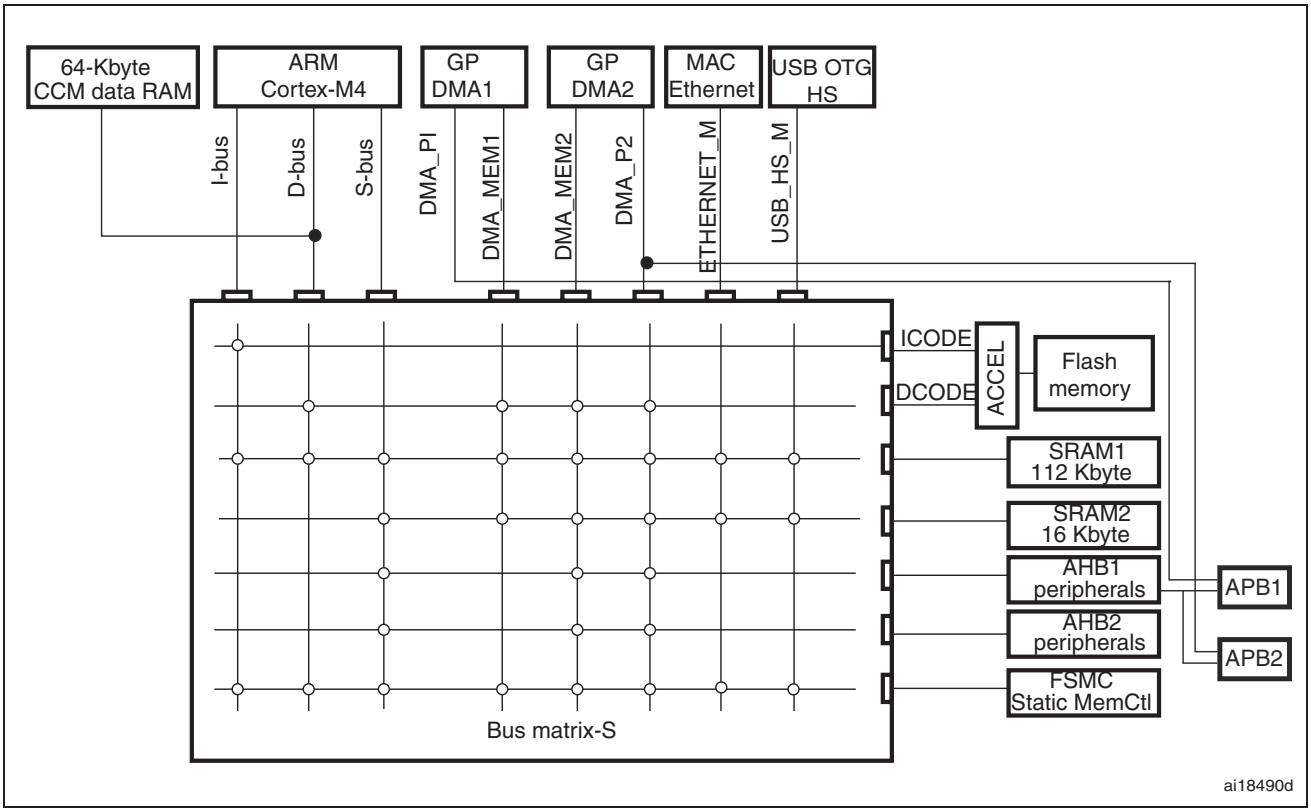
2.1. Архитектура системы

В STM32F405xx/07xx и STM32F415xx/17xx, основная система состоит из 32-bit многоуровневой матрицы AHB, соединяющей:

- восемь ведущих:
  - I-bus, D-bus and S-bus ядра Cortex®-M4 с FPU
  - Шину памяти DMA1
  - Шину памяти DMA2
  - Шину периферии DMA2
  - Шину Ethernet DMA
  - Шину USB OTG HS DMA
- Семь ведомых:
  - Шину внутренней Flash памяти ICode
  - Шину внутренней Flash памяти DCode
  - Основную внутреннюю SRAM1 (112 KB)
  - Дополнительную внутреннюю SRAM2 (16 KB)
  - Периферию AHB1, включая мосты между AHB и APB и периферию APB
  - Периферию AHB2
  - FSMC

Доступ идёт от ведущего к ведомому, позволяя одновременную работу нескольких высокоскоростных устройств. The 64-кбайт CCM (core coupled memory) RAM данных не является частью матрицы и доступна только через CPU. См. Рис. 1.

Рис. 1. Архитектура STM32F405xx/07xx и STM32F415xx/17xx.

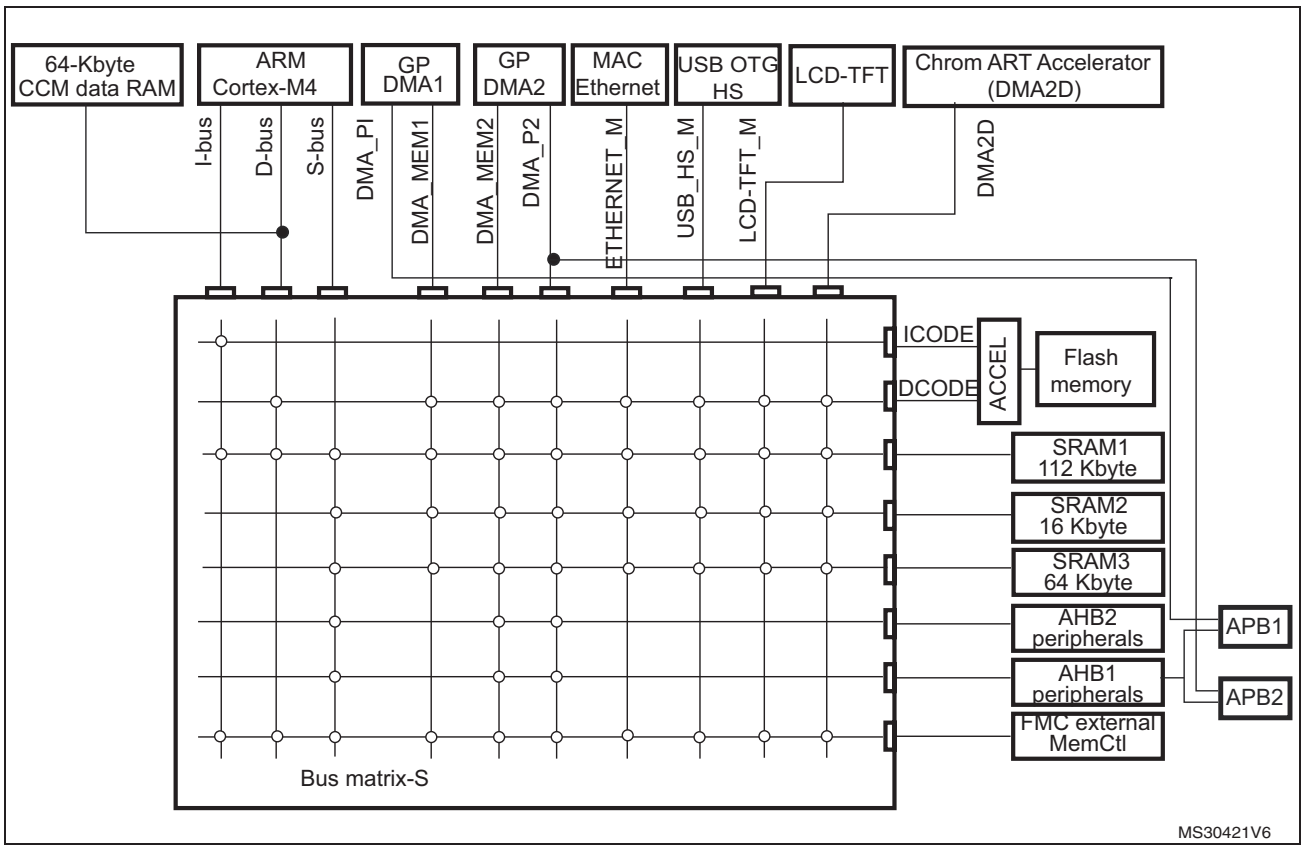


В STM32F42xx и STM32F43xx, основная система состоит из 32-bit многоуровневой матрицы AHB, соединяющей:

- Десять ведущих:
  - I-bus, D-bus and S-bus ядра Cortex®-M4 с FPU
  - Шину памяти DMA1
  - Шину памяти DMA2
  - Шину периферии DMA2
  - Шину Ethernet DMA
  - Шину USB OTG HS DMA
  - Шину LCD Controller DMA
  - Шину памяти DMA2D (Chrom-Art AcceleratorTM)
- Восемь ведомых:
  - Шину внутренней Flash памяти ICode
  - Шину внутренней Flash памяти DCode
  - Основную внутреннюю SRAM1 (112 KB)
  - Дополнительную внутреннюю SRAM2 (16 KB)
  - Дополнительную внутреннюю SRAM3 (64 KB)
  - Периферию AHB1, включая мосты между AHB и APB и периферию APB
  - Периферию AHB2
  - FMC

Доступ идёт от ведущего к ведомому, позволяя одновременную работу нескольких высокоскоростных устройств. The 64-кбайт CCM (core coupled memory) RAM данных не является частью матрицы и доступна только через CPU. См. Рис. 2.

Рис. 2. Архитектура STM32F42xxx и STM32F43xxx.



2.1.1. Шина I-Bus

Подключает шину команд ядра Cortex®-M4 с FPU к матрице шин. Используется для выборки команд (внутренняя Flash/SRAM память или внешняя память через FSMC/FMC).

2.1.2. Шина D-Bus

Подключает шину данных ядра Cortex®-M4 с FPU от 64-Kbyte CCM данных к матрице шин. Используется для обмена данными и отладки (внутренняя Flash память или внешняя память через FSMC/FMC).

2.1.3. Шина S-Bus

Подключает системную шину ядра Cortex®-M4 с FPU к матрице шин. Используется для обмена данными с периферией и SRAM. По ней можно грузить и команды, но это не так эффективно, как через ICode. Работает с внутренними SRAM1, SRAM2 и SRAM3, периферией AHB1, включая периферию APB, AHB2 и внешнюю память через FSMC/FMC.

2.1.4. Шина DMA памяти

Подключает интерфейс ведущего шины DMA памяти к матрице шин. Работает с внутренними SRAM1, SRAM2 и SRAM3 и внешней памятью через FSMC/FMC.

2.1.5. Шина DMA периферии

Подключает интерфейс ведущего DMA периферии к матрице шин для доступа к периферии и передач память-память. Работает с периферией AHB с APB, внутренними SRAM1, SRAM2 и SRAM3 и внешней памятью через FSMC/FMC.

2.1.6. Шина Ethernet DMA

Подключает интерфейс ведущего Ethernet DMA к матрице шин для доступа к памяти. Работает с внутренними SRAM1, SRAM2 и SRAM3, внутренней Flash и внешней памятью через FSMC/FMC.

2.1.7. Шина USB OTG HS DMA

Подключает интерфейс ведущего USB OTG HS DMA к матрице шин для доступа к памяти. Работает с внутренними SRAM1, SRAM2 и SRAM3, внутренней Flash и внешней памятью через FSMC/FMC.

2.1.8. Шина контроллера LCD-TFT DMA

Подключает интерфейс ведущего DMA контроллера LCD-TFT к матрице шин для доступа к памяти. Работает с внутренними SRAM1, SRAM2 и SRAM3, внутренней Flash и внешней памятью через FMC.

2.1.9. Шина DMA2D

Подключает интерфейс ведущего DMA2D графического ускорителя к матрице шин для доступа к памяти. Работает с внутренними SRAM1, SRAM2 и SRAM3, внутренней Flash и внешней памятью через FMC.

2.1.10. Шинная Матрица

Это арбитр ведущих. Используется кольцевой алгоритм Round Robin.

2.1.11. Мосты AHB/APB (APB)

Два моста AHB/APB дают полностью синхронное соединение между AHB и 2 шинами APB. С гибким выбором частоты периферии.

Карта адресов периферии приведена в Таблице 1 ниже.

После сброса каждого устройства их тактирование выключено (кроме SRAM и Flash) и перед использованием его нужно включить а регистрах `RCC_AHBENR` или `RCC_APBxENR`.

NB. При 16- или 8-бит доступе к регистрам APB мост преобразует 16- или 8-бит данные в 32-бит вектор.

2.2. Организация памяти

Память программ, данных, регистры и порты В/Выв организованы в единое линейное 4ГБ адресное пространство.

Всё кодируется в формате Little Endian. Байт слова с меньшим номером содержит младшие биты, байт с большим номером - старшие биты.

Адресное пространство разделено на 8 основных блоков по 512МБ.

Области памяти, не реализованные в кристалле или периферии называются "резервными".

2.3. Карта памяти

Границы адресов регистров:

Границы адресов	Периферия	Шина
0xA000 0000 - 0xA000 0FFF	FSMC (STM32F405xx/07xx и STM32F415xx/17xx) FMC (STM32F42xxx and STM32F43xxx)	AHB3
0x5006 0800 - 0x5006 0BFF	RNG	AHB2
0x5006 0400 - 0x5006 07FF	HASH	
0x5006 0000 - 0x5006 03FF	CRYP	
0x5005 0000 - 0x5005 03FF	DCMI	
0x5000 0000 - 0x5003 FFFF	USB OTG FS	
0x4004 0000 - 0x4007 FFFF	USB OTG HS	AHB1
0x4002 B000 - 0x4002 BBFF	DMA2D	
0x4002 8000 - 0x4002 93FF	ETHERNET MAC	
0x4002 6400 - 0x4002 67FF	DMA2	
0x4002 6000 - 0x4002 63FF	DMA1	
0x4002 4000 - 0x4002 4FFF	BKPSRAM	
0x4002 3C00 - 0x4002 3FFF	Flash interface register	
0x4002 3800 - 0x4002 3BFF	RCC	
0x4002 3000 - 0x4002 33FF	CRC	
0x4002 2800 - 0x4002 2BFF	GPIOK	
0x4002 2400 - 0x4002 27FF	GPIOJ	
0x4002 2000 - 0x4002 23FF	GPIOI	
0x4002 1C00 - 0x4002 1FFF	GPIOH	
0x4002 1800 - 0x4002 1BFF	GPIOG	
0x4002 1400 - 0x4002 17FF	GPIOF	
0x4002 1000 - 0x4002 13FF	GPIOE	
0x4002 0C00 - 0x4002 0FFF	GPIOD	
0x4002 0800 - 0x4002 0BFF	GPIOC	
0x4002 0400 - 0x4002 07FF	GPIOB	
0x4002 0000 - 0x4002 03FF	GPIOA	
0x4001 6800 - 0x4001 6BFF	LCD-TFT	
0x4001 5800 - 0x4001 5BFF	SAI1	

Границы адресов	Периферия	Шина
0x4001 5400 - 0x4001 57FF	SPI6	APB2
0x4001 5000 - 0x4001 53FF	SPI5	
0x4001 4800 - 0x4001 4BFF	TIM11	
0x4001 4400 - 0x4001 47FF	TIM10	
0x4001 4000 - 0x4001 43FF	TIM9	
0x4001 3C00 - 0x4001 3FFF	EXTI	
0x4001 3800 - 0x4001 3BFF	SYSCFG	
0x4001 3400 - 0x4001 37FF	SPI4	
0x4001 3000 - 0x4001 33FF	SPI1	
0x4001 2C00 - 0x4001 2FFF	SDIO	
0x4001 2000 - 0x4001 23FF	ADC1 - ADC2 - ADC3	
0x4001 1400 - 0x4001 17FF	USART6	
0x4001 1000 - 0x4001 13FF	USART1	
0x4001 0400 - 0x4001 07FF	TIM8	
0x4001 0000 - 0x4001 03FF	TIM1	
0x4000 7C00 - 0x4000 7FFF	UART8	APB1
0x4000 7800 - 0x4000 7BFF	UART7	
0x4000 7400 - 0x4000 77FF	DAC	
0x4000 7000 - 0x4000 73FF	PWR	
0x4000 6800 - 0x4000 6BFF	CAN2	
0x4000 6400 - 0x4000 67FF	CAN1	
0x4000 5C00 - 0x4000 5FFF	I2C3	
0x4000 5800 - 0x4000 5BFF	I2C2	
0x4000 5400 - 0x4000 57FF	I2C1	
0x4000 5000 - 0x4000 53FF	UART5	
0x4000 4C00 - 0x4000 4FFF	UART4	
0x4000 4800 - 0x4000 4BFF	USART3	
0x4000 4400 - 0x4000 47FF	USART2	
0x4000 4000 - 0x4000 43FF	I2S3ext	
0x4000 3C00 - 0x4000 3FFF	SPI3 / I2S3	
0x4000 3800 - 0x4000 3BFF	SPI2 / I2S2	
0x4000 3400 - 0x4000 37FF	I2S2ext	
0x4000 3000 - 0x4000 33FF	IWDG	
0x4000 2C00 - 0x4000 2FFF	WWDG	
0x4000 2800 - 0x4000 2BFF	RTC & BKP	
0x4000 2000 - 0x4000 23FF	TIM14	

Границы адресов	Периферия	Шина
0x4000 1C00 - 0x4000 1FFF	TIM13	APB1
0x4000 1800 - 0x4000 1BFF	TIM12	
0x4000 1400 - 0x4000 17FF	TIM7	
0x4000 1000 - 0x4000 13FF	TIM6	
0x4000 0C00 - 0x4000 0FFF	TIM5	
0x4000 0800 - 0x4000 0BFF	TIM4	
0x4000 0400 - 0x4000 07FF	TIM3	
0x4000 0000 - 0x4000 03FF	TIM2	

2.3.1. Встроенная SRAM

STM32F405xx/07xx и STM32F415xx/17xx имеют 4 Кбайта резервной SRAM (см. 5.1.2: Резервный домен) плюс 192 Кбайта системной SRAM.

STM32F42xxx STM32F43xxx имеют 4 Кбайта резервной SRAM (см. 5.1.2: Резервный домен) плюс 256 Кбайт системной SRAM.

Встроенная SRAM доступна байтами, полусловами (16 бит) и словами (32 бита). CPU выполняет чтение и запись с 0 состояниями ожидания. Встроенная SRAM разделена на три блока:

- SRAM1 и SRAM2 по адресам с 0x2000 0000, доступная всем ведущим AHB.
- SRAM3 (есть у STM32F42xxx и STM32F43xxx) по адресам с 0x2002 0000, доступная всем ведущим AHB.
- CCM (core coupled memory) по адресам с 0x1000 0000, доступная только CPU через D-bus.

Ведущие AHB поддерживают одновременный доступ к SRAM (от Ethernet или USB OTG HS). Например, Ethernet MAC может работать с SRAM2 пока CPU работает с SRAM1 или SRAM3.

CPU имеет доступ к SRAM1, SRAM2 и SRAM3 по Системной шине, I-Code/D-Code при загрузке из SRAM или изменении картирования (см. 9.2.1: Регистр картирования памяти SYSCFG (SYSCFG_MEMRMP) в контроллере SYSCFG). Программа в SRAM работает быстрее при включённом картировании.

2.3.2. Обзор Flash памяти

Интерфейс Flash памяти управляет доступом CPU AHB I-Code и D-Code к Flash памяти. Имеет механизмы стирания, программирования, чтения и защиты записи. Выполнение программ ускоряется за счёт предвыборки команд и кеширования. Память организована как:

- Блок основной памяти, разделённый на сектора.
- Системная память, откуда может загружаться устройство (если выбрано).
- 512 OTP (one-time programmable) байтов для данных пользователя.
- Байты опций для конфигурации чтения и защиты записи, уровня BOR, программы/железа сторожевого таймера и сброса в режимах Standby или Stop.

См. 3. Интерфейс встроенной Flash памяти.

2.3.3. Атомарный доступ

Память Cortex®-M4 с FPU включает два отображаемых (bit-band) региона. Здесь одно слово региона синонимов соответствует одному биту отображаемого региона. Запись в слово синоним равна операции чтения/модификации/записи в бит отображаемого региона.

В STM32F4xx и регистры периферии и SRAM это отображаемые регионы, что позволяет выполнять словный доступ к отдельным битам. Это доступно только для ядра Cortex®-M4 с FPU, но не для других ведущих устройств шин (т.е. DMA).

Отображение выполняется по формуле:

$$bit\_word\_addr = bit\_band\_base + (byte\_offset \times 32) + (bit\_number \times 4)$$

где:

bit_word_addr адрес слова в памяти синонимов

bit_band_base начальный адрес региона синонимов

byte_offset номер байта в отображаемом регионе с нужным битом

bit_number позиция нужного бита (0-7).

Пример:

Отображаем бит 2 байта в SRAM по адресу 0x2000 0300 в регион синонимов:

0x2200 6008 = 0x2200 0000 + (0x300*32) + (2*4).

Запись по адресу 0x2200 6008 эквивалентна операции чтения/модификации/записи в бит 2 байта в SRAM по адресу 0x2000 0300.

Чтение по адресу 0x2200 6008 возвращает значение (0x01 или 0x00) бита 2 байта в SRAM по адресу 0x2000 0300.

Совсем точно описано в *Cortex®-M4 with FPU programming manual*.

2.4. Конфигурация загрузки

Из-за фиксированной карты памяти область кода начинается с адреса 0x0000 0000 (доступ по шинам ICode/DCode), а область данных (SRAM) начинается с адреса 0x2000 0000 (доступ по системной шине). CPU всегда выбирает вектор сброса по шине ICode, т.е. область загрузки должна быть в области кода (обычно Flash память). Но в STM32F4xx можно грузиться из SRAM, а не только из основной флэш памяти или системной памяти.

Есть 3 режима загрузки, определяемых ножками BOOT[1:0].

Таблица 2. Режимы загрузки.

BOOT[1]	BOOT[0]	Откуда грузить
x	0	Основная флэш память
0	1	Системная память
1	1	Встроенная SRAM

Значения на ножках BOOT запоминаются на 4-м переднем фронте SYSCLK после сброса. Для выбора режима загрузки классно устанавливать ножки BOOT1 и BOOT0 после Сброса.

Также ножки BOOT опрашиваются при выходе из режима Standby. Следовательно, в этом режиме они должны быть в нужном состоянии. После истечения этой задержки CPU извлекает указатель стека по адресу 0x0000 0000 и начинает выполнение с адреса, хранящегося в векторе 0x0000 0004.

Ножка BOOT0 выделена специально, а BOOT1 - общая с ножкой GPIO, и после опроса может использоваться для совершенно посторонних целей.

NB. При загрузке из SRAM вам надо в коде инициализации программы с помощью регистра смещения таблицы исключений NVIC переместить таблицу векторов в SRAM.

В STM32F42xxx и STM32F43xxx есть возможность грузиться из любого банка основной флэш памяти. По умолчанию выбран Банк 1. Банк 2 выбирается установкой бита `BFB2` в байтах опций пользователя. Если он стоит и на ножках загрузки выбрана основная флэш память, то устройство загружается из системной памяти и загрузчик переходит на выполнение программы в Банке 2. За деталями идите в AN2606.

Встроенный загрузчик

Встроенный загрузчик размещается в системной памяти на этапе производства и используется для программирования флэш памяти через один из последовательных интерфейсов:

- USART1 (PA9/PA10).
- USART1 (PA9/PA10).
- CAN2 (PB5/13)
- USB OTG FS в режиме Устройства (DFU: device firmware upgrade).

Периферия USART работает от внутреннего генератора 16 MHz (HSI). CAN и USB OTG FS, работают при наличии внешнего (HSE), кратного 1 MHz (от 4 до 26 MHz).

За деталями снова идите в AN2606.

Физическое картирование в STM32F405xx/07xx и STM32F415xx/17xx

После выбора ножек можно изменять область кода (в этом случае код можно выполнять через шину ICode вместо Системной). Это делается с помощью 9.2.1. Регистр картирования памяти SYSCFG (SYSCFG_MEMRMP) в контроллере SYSCFG.

Можно картировать:

- Основную Flash память
- Системную память
- Встроенную SRAM1 (112 KB)
- FSMC банк 1 (NOR/PSRAM 1 и 2)

Таблица 3. Картирование памяти в STM32F405xx/07xx и STM32F415xx/17xx

Адреса	Основная Flash память	Встроенная SRAM	Системная память	FSMC
0x2001 C000 - 0x2001 FFFF	SRAM2 (16 KB)	SRAM2 (16 KB)	SRAM2 (16 KB)	SRAM2 (16 KB)
0x2000 0000 - 0x2001 BFFF	SRAM1 (112 KB)	SRAM1 (112 KB)	SRAM1 (112 KB)	SRAM1 (112 KB)
0x1FFF 0000 - 0x1FFF 77FF	Системная память	Системная память	Системная память	Системная память
0x0810 0000 - 0x0FFF FFFF	Резерв	Резерв	Резерв	Резерв
0x0800 0000 - 0x080F FFFF	Flash память	Flash память	Flash память	Flash память
0x0400 0000 - 0x07FF FFFF	Резерв	Резерв	Резерв	FSMC банк 1 NOR/PSRAM 2 (128 MB Aliased)
0x0000 0000 - 0x000F FFFF(1)(2)	Flash (1 MB) Aliased	SRAM1 (112 KB) Aliased	Системная память (30 KB) Aliased	FSMC банк 1 NOR/PSRAM 1 (128 MB Aliased)

1. Если FSMC легла по адресу 0x0000 0000, то можно картировать только первые два региона Банка 1 контроллера памяти(банк 1 NOR/PSRAM 1 и NOR/PSRAM 2). В режиме картирования внешняя память доступна для CPU через шину ICode вместо системной.
2. При отображении области загрузки, соответствующая память остаётся доступной и по старым адресам.

Физическое картирование в STM32F42xxx и STM32F43xxx

После выбора ножек можно изменять область кода (в этом случае код можно выполнять через шину ICode вместо Системной). Это делается с помощью 9.2.1. Регистр картирования памяти SYSCFG (SYSCFG_MEMRMP) в контроллере SYSCFG.

Можно картировать:

- Основную Flash память
- Системную память
- Встроенную SRAM1 (112 KB)
- FMC bank 1 (NOR/PSRAM 1 and 2)
- FMC SDRAM bank 1

Таблица 4. Картирование памяти в STM32F42xxx и STM32F43xxx

Адреса	Основная Flash память	Встроенная SRAM	Системная память	FMC
0x2002 0000 - 0x2002 FFFF	SRAM3 (64 KB)	SRAM3 (64 KB)	SRAM3 (64 KB)	SRAM3 (64 KB)
0x2001 C000 - 0x2001 FFFF	SRAM2 (16 KB)	SRAM2 (16 KB)	SRAM2 (16 KB)	SRAM2 (16 KB)
0x2000 0000 - 0x2001 BFFF	SRAM1 (112 KB)	SRAM1 (112 KB)	SRAM1 (112 KB)	SRAM1 (112 KB)
0x1FFF 0000 - 0x1FFF 77FF	Системная память	Системная память	Системная память	Системная память
0x0810 0000 - 0x0FFF FFFF	Резерв	Резерв	Резерв	Резерв
0x0800 0000 - 0x081F FFFF	Flash память	Flash память	Flash память	Flash память
0x0400 0000 - 0x07FF FFFF	Резерв	Резерв	Резерв	FMC bank 1 NOR/PSRAM 2 (128 MB Aliased)
0x0000 0000 - 0x001F FFFF(1)(2)	Flash (2 MB) Aliased	SRAM1 (112 KB) Aliased	Системная память (30 KB) Aliased	FMC банк 1 NOR/PSRAM 1 (128 MB Aliased) or FMC SDRAM банк 1 (128 MB Aliased)

1. Если FSMC легла по адресу 0x0000 0000, то можно картировать только первые два региона Банка 1 контроллера памяти(банк 1 NOR/PSRAM 1 и NOR/PSRAM 2). В режиме картирования внешняя память доступна для CPU через шину ICode вместо системной.
2. При отображении области загрузки, соответствующая память остаётся доступной и по старым адресам.

3. Интерфейс встроенной Flash памяти

3.1. Введение

Интерфейс Flash памяти управляет доступом CPU AHB I-Code и D-Code к Flash памяти. Имеет механизмы стирания, программирования, чтения и защиты записи. Выполнение программ ускоряется за счёт предвыборки команд и кеширования.

3.2. Основные свойства

- Чтение Flash памяти
- Запись/стирание Flash памяти
- Защита чтения/записи
- Предзагрузка по I-Code
- 64 линии кэша по 128 бит на I-Code
- 8 линий кэша по 128 бит на D-Code

Рис. 3. Интерфейс Flash памяти в системе (STM32F405xx/07xx и STM32F415xx/17xx)

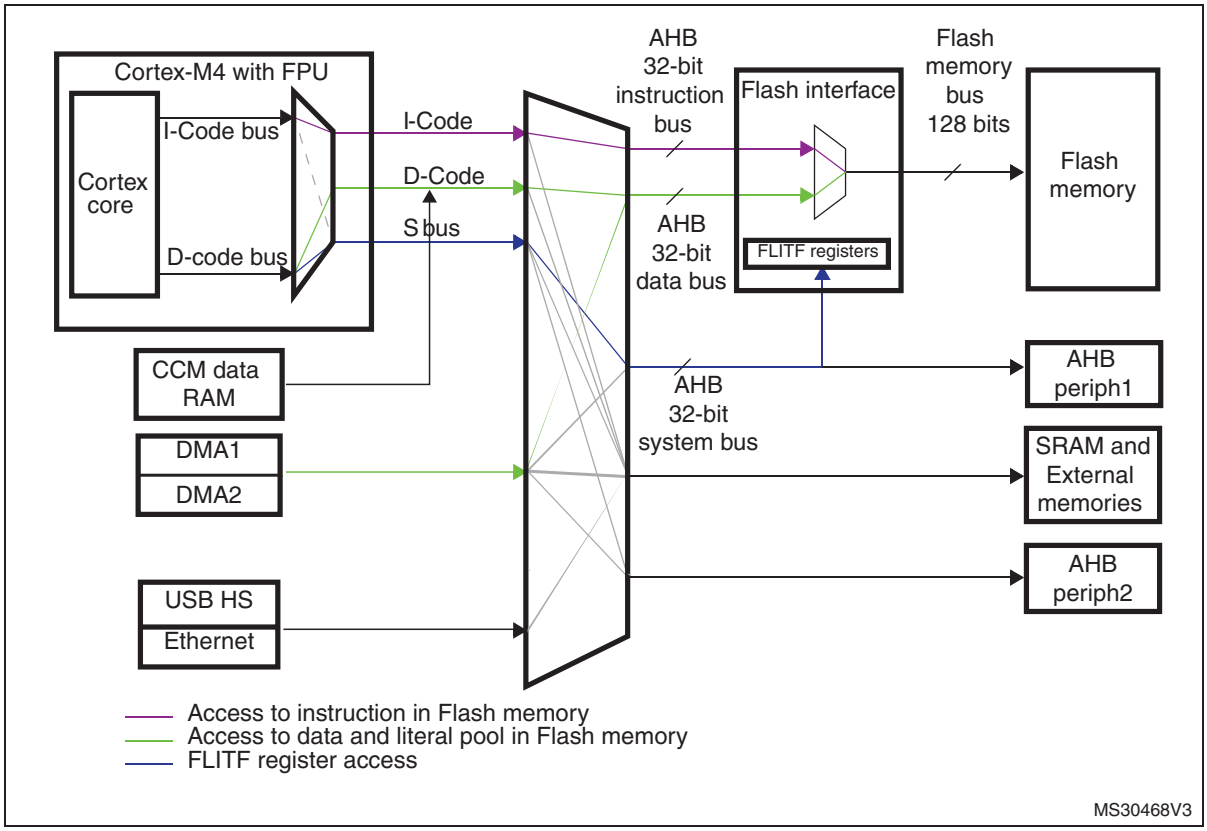
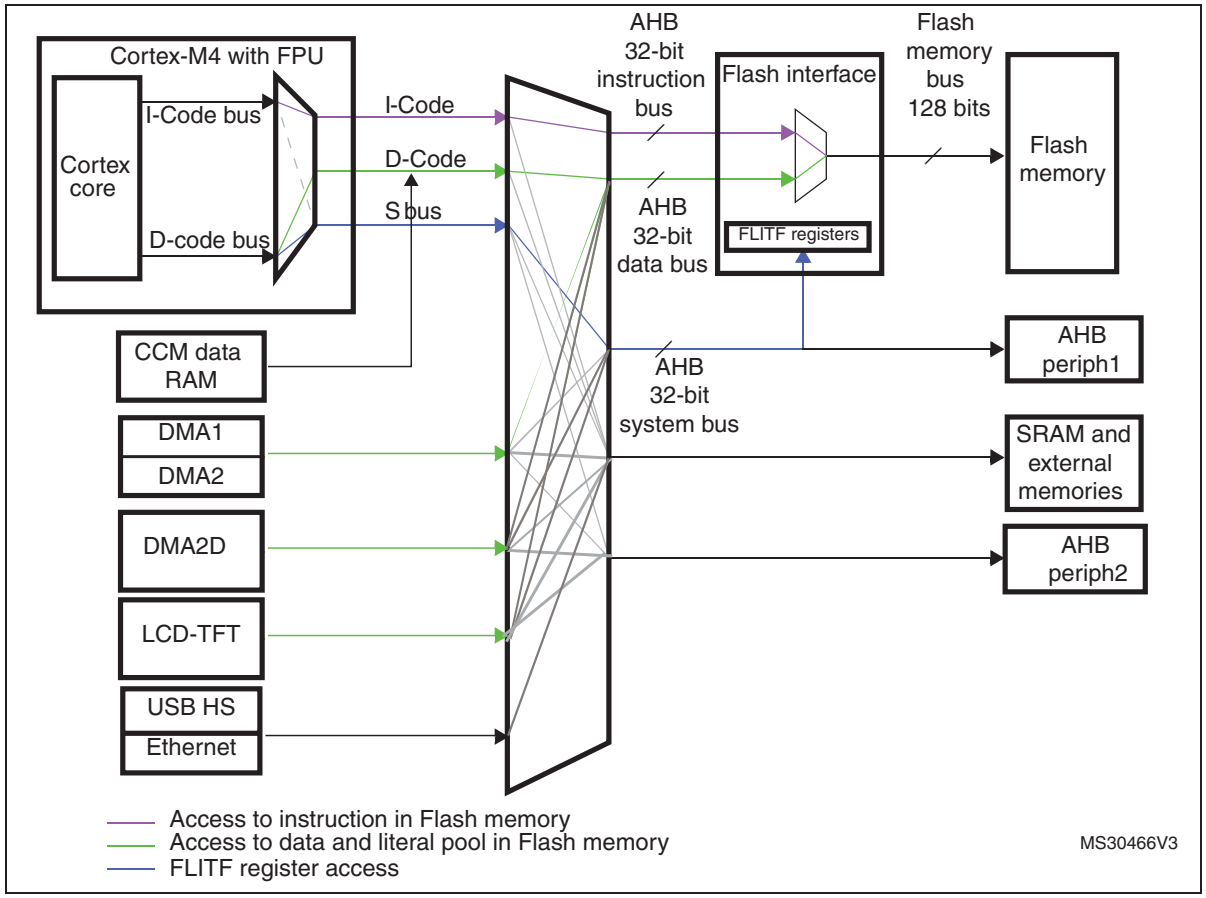


Рис. 3. Интерфейс Flash памяти в системе (STM32F42xxx и STM32F43xxx)



3.3. Встроенная Flash память в STM32F405xx/07xx и STM32F415xx/17xx

Характеристики:

- Ёмкость до 1 Мбайт
- 128 бит ширина чтения
- Запись байтами, полусловами, словами и двойными словами
- Секторное и массовое стирание
- Организация:
  - Основной блок из 4 секторов по 16 Кбайт, 1 сектор из 64 Кбайт, и 7 секторов по 128 Кбайт
  - Системная память для начальной загрузки в соответствующем режиме
  - 512 OTP (one-time programmable) байтов для данных пользователя. Область OTP содержит 16 дополнительных байтов для блокировкинужных блоков данных OTP.
  - Байты Опций для конфигурации защиты чтения и записи, уровня BOR, программы/железа сторожевого таймера и сброса в режимах Standby и Stop.
- Экономные режимы (См. Управление питанием (PWR))

Таблица 5. Организация Flash модуля в STM32F40x и STM32F41x

Блок	Имя	Базовый адрес	Размер
Основная память	Сектор 0	0x0800 0000 - 0x0800 3FFF	16 Кбайт
	Сектор 1	0x0800 4000 - 0x0800 7FFF	16 Кбайт
	Сектор 2	0x0800 8000 - 0x0800 BFFF	16 Кбайт
	Сектор 3	0x0800 C000 - 0x0800 FFFF	16 Кбайт
	Сектор 4	0x0801 0000 - 0x0801 FFFF	64 Кбайт
	Сектор 5	0x0802 0000 - 0x0803 FFFF	128 Кбайт
	Сектор 6	0x0804 0000 - 0x0805 FFFF	128 Кбайт
	. . .	. . .	. . .
	Сектор 11	0x080E 0000 - 0x080F FFFF	128 Кбайт
Системная память		0x1FFF 0000 - 0x1FFF 77FF	30 Кбайт
Область OTP		0x1FFF 7800 - 0x1FFF 7A0F	528 байт
Байты Опций		0x1FFF C000 - 0x1FFF C00F	16 байт

3.4. Встроенная Flash память в STM32F42xxx и STM32F43xxx

Характеристики:

- Ёмкость до 2 Мбайт в двух банках с поддержкой чтения при записи (RWW)
- 128 бит ширина чтения
- Запись байтами, полусловами, словами и двойными словами
- Секторное, по-банковое и массовое стирание
- Организация из двух банков:
  - Для каждого банка (1 Мбайт): основной блок из 4 секторов по 16 Кбайт, 1 сектор из 64 Кбайт, и 7 секторов по 128 Кбайт
  - Системная память для начальной загрузки в соответствующем режиме
  - 512 OTP (one-time programmable) байтов для данных пользователя. Область OTP содержит 16 дополнительных байтов для блокировкинужных блоков данных OTP.
  - Байты Опций для конфигурации защиты чтения и записи, уровня BOR, программы/железа сторожевого таймера и сброса в режимах Standby и Stop.
- Организация двух банков на 1 Мбайт устройствах
  Разрешается установкой бита `DB1M` опций.
  При этом последние 512 Кбайт одного банка (секторы [8:11]) перестраиваются также как и

первые 512 Кбайт.

Нумерация секторов при двух банках отличается от одного банка: одинарный банк содержит 12 секторов, а два банка содержат 16 (см. Таблицу 7.).

Для операции стирания нумерация секторов зависит от бита DB1M опций.

- Если DB1M чист, то стирание выполняется с номером сектора по умолчанию.
- Если DB1M стоит, то для стирания в банке 2 номер сектора надо задавать (от 12 до 19) в регистре FLASH_CR.

Таблица 6. Flash модуль - 2 Мбайта, два банка (STM32F42xxx и STM32F43xxx)

Блок	Банк	Имя	Базовый адрес	Размер
Основная память	Банк 1	Сектор 0	0x0800 0000 - 0x0800 3FFF	16 Кбайт
		Сектор 1	0x0800 4000 - 0x0800 7FFF	16 Кбайт
		Сектор 2	0x0800 8000 - 0x0800 BFFF	16 Кбайт
		Сектор 3	0x0800 C000 - 0x0800 FFFF	16 Кбайт
		Сектор 4	0x0801 0000 - 0x0801 FFFF	64 Кбайт
		Сектор 5	0x0802 0000 - 0x0803 FFFF	128 Кбайт
		Сектор 6	0x0804 0000 - 0x0805 FFFF	128 Кбайт
		-	-	-
		Сектор 11	0x080E 0000 - 0x080F FFFF	128 Кбайт
	Банк 2	Сектор 12	0x0810 0000 - 0x0810 3FFF	16 Кбайт
		Сектор 13	0x0810 4000 - 0x0810 7FFF	16 Кбайт
		Сектор 14	0x0810 8000 - 0x0810 BFFF	16 Кбайт
		Сектор 15	0x0810 C000 - 0x0810 FFFF	16 Кбайт
		Сектор 16	0x0811 0000 - 0x0811 FFFF	64 Кбайт
		Сектор 17	0x0812 0000 - 0x0813 FFFF	128 Кбайт
		Сектор 18	0x0814 0000 - 0x0815 FFFF	128 Кбайт
			-	-
		Сектор 23	0x081E 0000 - 0x081F FFFF	128 Кбайт
Системная память			0x1FFF 0000 - 0x1FFF 77FF	30 Кбайт
OTP			0x1FFF 7800 - 0x1FFF 7A0F	528 байт
Байты опций	Банк 1		0x1FFF C000 - 0x1FFF C00F	16 байт
	Банк 2		0x1FFE C000 - 0x1FFE C00F	16 байт

Таблица 7. Flash модуль - 1 Мбайт, один банк и два банка (STM32F42xxx и STM32F43xxx)

1 Мбайт один банк (по умолчанию)			1 Мбайт два банка		
DB1M=0			DB1M=1		
Память	Номер сектора	Размер сектора	Память	Номер сектора	Размер сектора
1MB	Сектор 0	16 Кбайт	Банк 1 512KB	Сектор 0	16 Кбайт
	Сектор 1	16 Кбайт		Сектор 1	16 Кбайт
	Сектор 2	16 Кбайт		Сектор 2	16 Кбайт
	Сектор 3	16 Кбайт		Сектор 3	16 Кбайт
	Сектор 4	64 Кбайт		Сектор 4	64 Кбайт
	Сектор 5	128 Кбайт		Сектор 5	128 Кбайт
	Сектор 6	128 Кбайт		Сектор 6	128 Кбайт
	Сектор 7	128 Кбайт		Сектор 7	128 Кбайт
	Сектор 8	128 Кбайт		Сектор 12	16 Кбайт
	Сектор 9	128 Кбайт		Сектор 13	16 Кбайт
	Сектор 10	128 Кбайт		Сектор 14	16 Кбайт

1 Мбайт один банк (по умолчанию)			1 Мбайт два банка		
	Сектор 11	128 Кбайт	Банк 2 512KB	Сектор 15	16 Кбайт
	-	-		Сектор 16	64 Кбайт
	-	-		Сектор 17	128 Кбайт
	-	-		Сектор 18	128 Кбайт
	-	-		Сектор 19	128 Кбайт

Таблица 8. Flash модуль - 1 Мбайт, один банк (STM32F42xxx и STM32F43xxx)

Блок	Банк	Имя	Базовый адрес	Размер
Основная память	Один банк	Сектор 0	0x0800 0000 - 0x0800 3FFF	16 Кбайт
		Сектор 1	0x0800 4000 - 0x0800 7FFF	16 Кбайт
		Сектор 2	0x0800 8000 - 0x0800 BFFF	16 Кбайт
		Сектор 3	0x0800 C000 - 0x0800 FFFF	16 Кбайт
		Сектор 4	0x0801 0000 - 0x0801 FFFF	64 Кбайт
		Сектор 5	0x0802 0000 - 0x0803 FFFF	128 Кбайт
		Сектор 6	0x0804 0000 - 0x0805 FFFF	128 Кбайт
		Сектор 7	0x0806 0000 - 0x0807 FFFF	128 Кбайт
		Сектор 8	0x0808 0000 - 0x0809 FFFF	128 Кбайт
		Сектор 9	0x080A 0000 - 0x080B FFFF	128 Кбайт
		Сектор 10	0x080C 0000 - 0x080D FFFF	128 Кбайт
		Сектор 11	0x080E 0000 - 0x080F FFFF	128 Кбайт
Системная память			0x1FFF 0000 - 0x1FFF 77FF	30 Кбайт
OTP			0x1FFF 7800 - 0x1FFF 7A0F	528 байт
Байты Опций			0x1FFF C000 - 0x1FFF C00F	16 байт
			0x1FFE C000 - 0x1FFE C00F	16 байт

Таблица 9. Flash модуль - 1 Мбайт, два банка (STM32F42xxx и STM32F43xxx)

Блок	Банк	Имя	Базовый адрес	Размер
Основная память	Банк 1	Сектор 0	0x0800 0000 - 0x0800 3FFF	16 Кбайт
		Сектор 1	0x0800 4000 - 0x0800 7FFF	16 Кбайт
		Сектор 2	0x0800 8000 - 0x0800 BFFF	16 Кбайт
		Сектор 3	0x0800 C000 - 0x0800 FFFF	16 Кбайт
		Сектор 4	0x0801 0000 - 0x0801 FFFF	64 Кбайт
		Сектор 5	0x0802 0000 - 0x0803 FFFF	128 Кбайт
		Сектор 6	0x0804 0000 - 0x0805 FFFF	128 Кбайт
		Сектор 7	0x0806 0000 - 0x0807 FFFF	128 Кбайт
	Банк 2	Сектор 12	0x0808 0000 - 0x0808 3FFF	16 Кбайт
		Сектор 13	0x0808 4000 - 0x0808 7FFF	16 Кбайт
		Сектор 14	0x0808 0000 - 0x0808 BFFF	16 Кбайт
		Сектор 15	0x0808 C000 - 0x0808 FFFF	16 Кбайт
		Сектор 16	0x0809 0000 - 0x0809 FFFF	64 Кбайт
		Сектор 17	0x080A 0000 - 0x080B FFFF	128 Кбайт
		Сектор 18	0x080C 0000 - 0x080D FFFF	128 Кбайт
		Сектор 19	0x080E 0000 - 0x080F FFFF	128 Кбайт
Системная память			0x1FFF 0000 - 0x1FFF 77FF	30 Кбайт
OTP			0x1FFF 7800 - 0x1FFF 7A0F	528 байт
Байты Опций	Банк 1		0x1FFF C000 - 0x1FFF C00F	16 байт
	Банк 2		0x1FFE C000 - 0x1FFE C00F	16 байт

3.5. Интерфейс чтения

3.5.1. Частота тактов CPU и время чтения Flash

Для правильного чтения из Flash надо в регистре FLASH_ACR установить число состояний ожидания (LATENCY) в зависимости от частоты тактов CPU (HCLK) и напряжения питания.

При напряжении питания ниже 2.1 В надо отключать буфер предвыборки. См. Таблицу 10 и Таблицу 11.

NB:

- На STM32F405xx/07xx и STM32F415xx/17xx:
  - если VOS = '0', максимальное значение f_{HCLK} = 144 MHz.
  - если VOS = '1', максимальное значение f_{HCLK} = 168 MHz.
- На STM32F42xxx и STM32F43xxx:
  - если VOS[1:0] = '0x01', максимальное значение f_{HCLK} = 120 MHz.
  - если VOS[1:0] = '0x10', максимальное значение f_{HCLK} = 144 MHz. Можно до 168 MHz в режиме **over-drive**.
  - если VOS[1:0] = '0x11, максимальное значение f_{HCLK} = 168 MHz. Можно до 180 MHz в режиме **over-drive**.
  - Режим **over-drive** недоступен при V_{DD} от 1.8 до 2.1 V.

См. 5.1.4. Регулятор напряжения для STM32F42xxx и STM32F43xxx о режиме **over-drive**.

Таблица 10. Ожидания и частота CPU (HCLK) (STM32F405xx/07xx и STM32F415xx/17xx)

Ожидание	HCLK (MHz)			
	2.7 V - 3.6 V	2.4 V - 2.7 V	2.1 V - 2.4 V	1.8 V - 2.1 V Предвыборка выкл.
0 WS (1 CPU cycle)	0 < HCLK ≤30	0 < HCLK ≤24	0 < HCLK ≤22	0 < HCLK ≤20
1 WS (2 CPU cycles)	30 < HCLK ≤60	24 < HCLK ≤48	22 < HCLK ≤44	20 <HCLK ≤40
2 WS (3 CPU cycles)	60 < HCLK ≤90	48 < HCLK ≤72	44 < HCLK ≤66	40 < HCLK ≤60
3 WS (4 CPU cycles)	90 < HCLK ≤120	72 < HCLK ≤96	66 < HCLK ≤88	60 < HCLK ≤80
4 WS (5 CPU cycles)	120 < HCLK ≤150	96 < HCLK ≤120	88 < HCLK ≤110	80 < HCLK ≤100
5 WS (6 CPU cycles)	150 < HCLK ≤168	120 < HCLK ≤144	110 < HCLK ≤132	100 < HCLK ≤120
6 WS (7 CPU cycles)		144 < HCLK ≤168	132 < HCLK ≤154	120 < HCLK ≤140
7 WS (8 CPU cycles)			154 < HCLK ≤168	140 < HCLK ≤160

Таблица 11. Ожидания и частота CPU (HCLK) (STM32F42xxx и STM32F43xxx)

Ожидание	HCLK (MHz)			
	2.7 V - 3.6 V	2.4 V - 2.7 V	2.1 V - 2.4 V	1.8 V - 2.1 V Prefetch OFF
0 WS (1 CPU cycle)	0 <HCLK ≤30	0 <HCLK ≤24	0 <HCLK ≤22	0 < HCLK ≤20
1 WS (2 CPU cycles)	30 <HCLK ≤60	24 < HCLK ≤48	22 <HCLK ≤44	20 <HCLK ≤40
2 WS (3 CPU cycles)	60 <HCLK ≤90	48 < HCLK ≤72	44 < HCLK ≤66	40 < HCLK ≤60
3 WS (4 CPU cycles)	90 <HCLK ≤120	72 < HCLK ≤96	66 <HCLK ≤88	60 < HCLK ≤80
4 WS (5 CPU cycles)	120 <HCLK ≤150	96 < HCLK ≤120	88 < HCLK ≤110	80 < HCLK ≤100
5 WS (6 CPU cycles)	150 <HCLK ≤180	120 <HCLK ≤144	110 < HCLK ≤132	100 < HCLK ≤120
6 WS (7 CPU cycles)		144 <HCLK ≤168	132 < HCLK ≤154	120 < HCLK ≤140
7 WS (8 CPU cycles)		168 <HCLK ≤180	154 <HCLK ≤176	140 < HCLK ≤160
8 WS (9 CPU cycles)			176 <HCLK ≤180	160 < HCLK ≤168

После сброса в регистре FLASH_ACR стоят частота CPU = 16 MHz и 0 ожиданий (WS).

Вот последовательность подстройки числа ожиданий:

Увеличение частоты CPU:

- Пишем число ожиданий в биты LATENCY регистра FLASH_ACR
- Проверяем его установку чтением регистра FLASH_ACR
- Изменяем источник тактов CPU в битах SW регистра RCC_CFGR
- Если надо, изменяем предделитель тактов CPU битами HPRE регистра RCC_CFGR
- Проверяем новые установки чтением регистра RCC_CFGR (биты SWS и/или HPRE).

Уменьшение частоты CPU:

- Изменяем источник тактов CPU в битах SW регистра RCC_CFGR
- Если надо, изменяем предделитель тактов CPU битами HPRE регистра RCC_CFGR
- Проверяем новые установки чтением регистра RCC_CFGR (биты SWS и/или HPRE)
- Пишем число ожиданий в биты LATENCY регистра FLASH_ACR
- Проверяем его установку чтением регистра FLASH_ACR

3.5.2. Адаптивный ускоритель памяти (ART Accelerator™)

Он реализует очередь предвыборки команд и кэш переходов, увеличивая скорость выполнения равную частоте CPU в 180 MHz.

Предвыборка команд

Из Flash памяти за раз читаются 128 бит, или 4 команды по 32 бит или 8 команд по 16 бит, что полезно для последовательно исполняемых команд. Предвыборка следующих команд по шине I-Code идёт одновременно с выполнением предыдущих. Она разрешается установкой бита PRFTEN регистра FLASH_ACR.

При наличии переходов, когда следующая команда ещё не прочитана, появляются состояния ожидания.

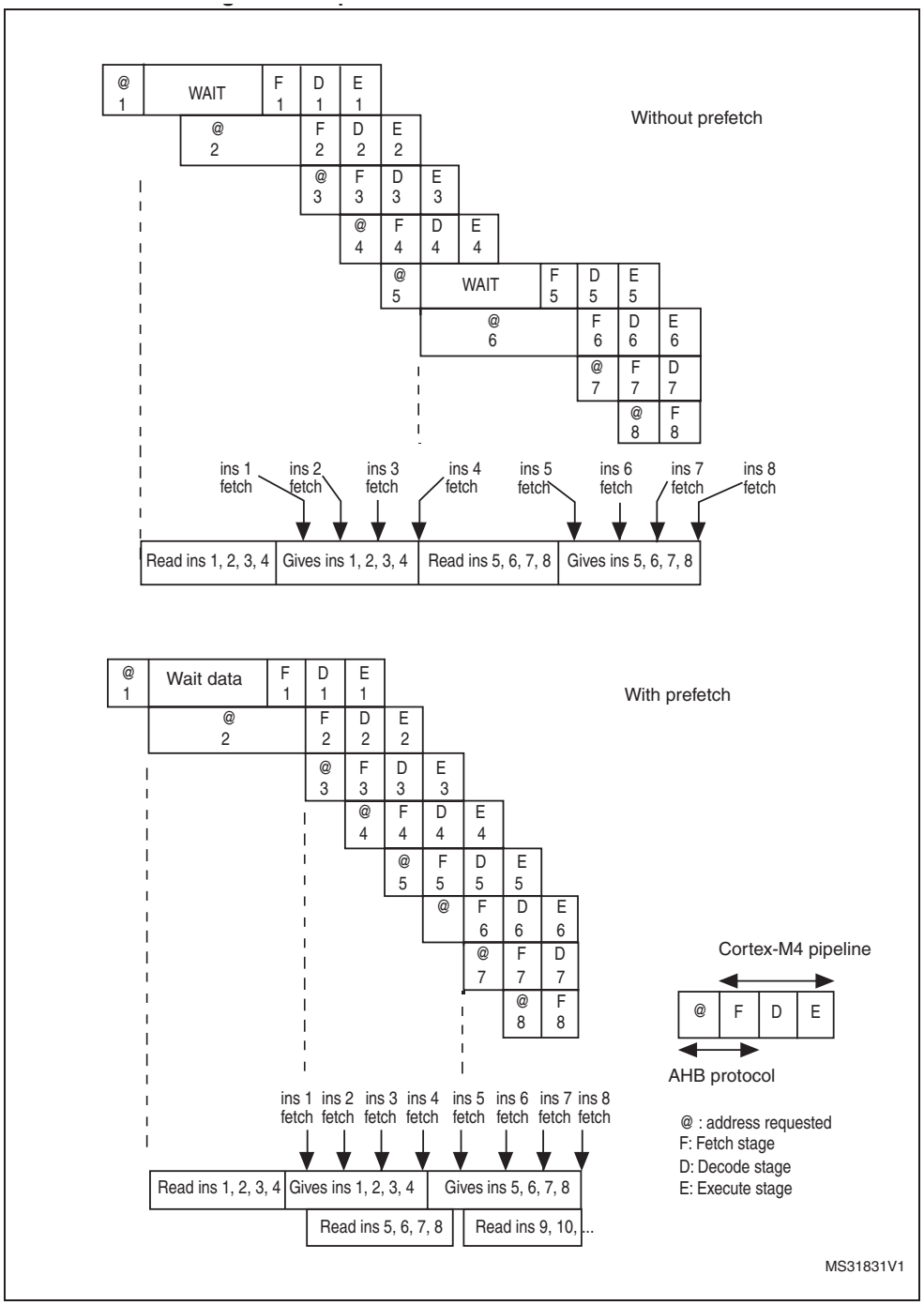


Рис. 5. Выполнение 32-bit команд с и без предвыборки с 3 WS

Кэш память команд

Для ограничения потерь при переходах можно запоминать 64 строки по 128 бит в кэш памяти команд. Разрешается это установкой бита ICEN регистра FLASH_ACR. При появлении команды перехода и отсутствии нужного места в работающей или предзагруженной строке прочитанная строка пишется в кэш память команд. Теперь имеющаяся затребованная CPU команда читается из кэша. При заполнении кэша освобождается наименее использованная строка команд (алгоритм LRU).

Управление данными

Литеральные пулы извлекаются из Flash памяти по шине D-Code на стадии выполнения конвейера CPU, который приостанавливается до получения данных. Поэтому шина AHB D-Code приоритетнее шины AHB I-Code.

Для часто используемых данных можно разрешить кэш данных установкой бита DCEN регистра FLASH_ACR. Он работает подобно кэшу команд, но имеет размер 8 колонок по 128 бит.

NB: Данные в секторе конфигурации пользователя не кэшируются.

3.6. Операции стирания и записи

Для этих операций частота тактов CPU (HCLK) должна быть не ниже 1 MHz. Сброс во время записи/стирания может разрушить содержимое Flash памяти.

Попытка чтения Flash на STM32F4xx во время записи/стирания приводит к остановке шины, то есть во время записи/стирания операции чтения быть не должно.

На STM32F42xxx и STM32F43xxx с двумя банками читать можно из одного во время записи/стирания другого банка.

3.6.1. Разблокирование регистра управления Flash

После сброса запись в Flash запрещена в регистре FLASH_CR. Разрешаем её так:

1. Пишем KEY1 = 0x45670123 в регистр FLASH_KEYR
2. Пишем KEY2 = 0xCDEF89AB в регистр FLASH_KEYR

Иная последовательность заблокирует регистр FLASH_CR до следующего сброса.

Блокируют регистр FLASH_CR установкой бита LOCK регистра FLASH_CR.

NB: При записи регистр FLASH_CR недоступен пока стоит бит BSY регистра FLASH_SR. Попытка записи при стоящем BSY останавливает шину AHB вплоть до его очистки.

3.6.2. Параллелизм записи/стирания

Число записываемых за раз байтов задаётся в поле PSIZE регистра FLASH_CR до начала операции записи. Оно ограничивается напряжением питания и наличием внешнего напряжения V_{PP} .

Стирать можно секторами, банками или всю Flash память. Время стирания зависит от значения PSIZE.

Таблица 12. Параллелизм записи/стирания (PSIZE)

	2.7 - 3.6 V с V_{PP}	2.7 - 3.6 V	2.4 - 2.7 V	2.1 - 2.4 V	1.8 V - 2.1 V
Параллелизм	x64	x32	x16		x8
PSIZE(1:0)	11	10	1		0

NB: Запись/стирание с негодными установками параллелизма могут дать непредсказуемый результат даже при нормальном первичном контроле (может не сохраниться).

Внешнее V_{PP}, (от 8 до 9 V) подают на контакт V_{PP}. Напряжение должно сохраняться даже при потреблении более 10 mA. Использовать V_{PP} рекомендуют только на производстве. Подавать V_{PP} нужно не более часа, дабы не сжечь Flash память.

3.6.3. Стирание

Flash память можно стирать секторами или целиком (массовое стирание). Массовое стирание не затрагивает секторы OTP и конфигурации.

Стирание сектора

Процедура:

1. Смотрим, что Flash не занята (бит BSY регистра FLASH_SR)
2. Ставим бит SER и выбираем сектор из 12 для STM32F405xx/07xx и STM32F415xx/17xx или из 24 для STM32F42xxx and STM32F43xxx в основном блоке памяти в поле SNB регистра FLASH_CR
3. Ставим бит STRT регистра FLASH_CR
4. Ждём очистки бита BSY

Стирание банка в STM32F42xxx и STM32F43xxx

Процедура:

1. Смотрим, что Flash не занята (бит BSY регистра FLASH_SR)
2. Ставим бит MER или MER1 регистра FLASH_CR
3. Ставим бит STRT регистра FLASH_CR
4. Ждём очистки бита BSY

Массовое стирание

Процедура:

1. Смотрим, что Flash не занята (бит BSY регистра FLASH_SR)
2. Ставим бит MER регистра FLASH_CR на STM32F405xx/07xx и STM32F415xx/17xx
3. Ставим биты MER и MER1 регистра FLASH_CR на STM32F42xxx и STM32F43xxx
4. Ставим бит STRT регистра FLASH_CR
5. Ждём очистки бита BSY

NB: Если стоят и MERx и SER регистра FLASH_CR, то выполняется массовое стирание.

Если ставить бит STRT при чистых битах MERx и SER, то поведение непредсказуемо. Запрещено.

3.6.4. Запись

Стандартная запись

Процедура:

1. Смотрим, что основная Flash не занята (бит BSY регистра FLASH_SR)
2. Ставим бит PG регистра FLASH_CR
3. Пишем данные по нужному адресу основного блока или области OTP:
   - Байтами при параллелизме x8
   - Полусловами при параллелизме x16
   - Словами при параллелизме x32
   - Двойными словами при параллелизме x64
4. Ждём очистки бита BSY

NB: Запись с переходом бита из ‘1’ в '0’ стирания не требует, а для записи ‘1’ оно нужно.

Если стирание и запись затребованы одновременно, то сначала идёт стирание.

Ошибки записи

Писать во Flash данные, пересекающие границу 128-бит, запрещено. Запись не выполняется и в регистре FLASH_SR ставится флаг ошибки адреса PGAERR.

Выравнивание адреса записи должно соответствовать выбранному параллелизму (x8, x16, x32, x64). Иначе запись не выполняется и в регистре FLASH_SR ставится флаг ошибки параллелизма PGPERR.

Несоблюдение последовательности записи приводит к прерыванию операции и в регистре FLASH_SR ставится флаг ошибки последовательности PGSERR.

Запись и кэширование

Если запись в Flash память имеет связана с кэшем данных, то изменяются данные и в памяти и в кэше. Стирание также касается и данных в кэше. Надо убедиться, что они изменятся до обращения к ним. Рекомендуется сливать кэши установкой битов DCRST и ICRST в регистре FLASH_CR.

NB: Сливать можно только отключённый I/D кэш (I/DCEN = 0).

3.6.5. Чтение при записи (RWW)

В STM32F42xxx и STM32F43xxx можно читать из одного банка во время записи или стирания другого. А вот записывать так нельзя.

Чтение банка 1 при стирании банка 2

Процедура:

1. Смотрим, что Flash не занята (бит BSY регистра FLASH_SR)
2. Ставим бит MER или MER1 регистра FLASH_CR
3. Ставим бит STRT регистра FLASH_CR
4. Ждём очистки бита BSY

Чтение банка 1 при записи банка 2

Процедура:

1. Смотрим, что Flash не занята (бит BSY регистра FLASH_SR)
2. Ставим бит PG регистра FLASH_CR
3. Пишем данные по нужному адресу основного блока или области OTP.

4. Ждём очистки бита BSY

3.6.6. Прерывания

Прерывание конца операции записи или стирания (снятие бита BSY регистра FLASH_SR) разрешается установкой бита EOPIE регистра FLASH_CR. В этом случае ставится бит конца операции EOP регистра FLASH_SR.

При появлении ошибки в регистре FLASH_SR ставятся флаги:

- PGAERR, PGPERR, PGSERR (Ошибки записи)
- WRPERR (Ошибка защиты записи)
- RDERR (Ошибка защиты чтения) только STM32F42xxx и STM32F43xxx.

Прерывание по ошибке разрешается установкой бита ERRIE регистра FLASH_CR, одновременно ставится бит ошибки OPERR регистра FLASH_SR.

NB: При обнаружении нескольких ошибок (например, при передачах DMA в Flash память), флаги ошибок не снимаются вплоть до конца успешной записи.

Таблица 13. Прерывания Flash

Прерывание	Флаг	Бит разрешения
Конец операции	EOP	EOPIE
Ошибка защиты записи	WRPERR	ERRIE
Ошибка записи	PGAERR, PGPERR, PGSERR	ERRIE
Ошибка защиты чтения	RDERR	ERRIE

3.7. Байты Опций

3.7.1. Описание Байтов Опций

Пишутся конечным пользователем в секторе конфигурации пользователя.

Таблица 14. Организация байтов Опций

Адрес	[63:16]	[15:0]
0x1FFF C000	Резерв	ROP & опции пользователя (RDP & USER)
0x1FFF C008	Резерв	SPRMOD и биты защиты записи nWRP секторов 0 до 11
0x1FFE C000	Резерв	Резерв
0x1FFE C008	Резерв	SPRMOD и биты защиты записи nWRP секторов 12 до 23

Таблица 15. Описание байтов Опций для STM32F405xx/07xx и STM32F415xx/17xx

Байты Опций (слово, адрес 0x1FFF C000)	
RDP: Защита чтения.	
Биты 15:8	0xAA: Уровень 0, защиты нет 0xCC: Уровень 2, зашита чипа (отладка и загрузка из RAM выключены) Иное: Уровень 1, защита чтения памяти (отладка ограничена)
USER: Опции пользователя: – Событие сторожевого таймера: Аппаратное или программное – Событие сброса при входе в режим Stop – Событие сброса при входе в режим Standby	
Бит 7	nRST_STDBY 0: При входе в Standby выдаётся сброс 1: Сброс не выдаётся
Бит 6	nRST_STOP 0: При входе в Stop выдаётся сброс 1: Сброс не выдаётся
Бит 5	WDG_SW 0: Аппаратно независимый сторожевой таймер 1: Программно независимый сторожевой таймер

Бит 4	0x1: Не используется
Биты 3:2	BOR_LEV: Уровень сброса BOR (падение питания). 00: BOR уровень 3 (VBOR3) 01: BOR уровень 2 (VBOR2) 10: BOR уровень 1 (VBOR1) 11: BOR выкл., используется уровень порога POR/PDR См. Электрические характеристики.
Биты 1:0	0x1: Не используется
Байты Опций (слово, адрес 0x1FFF C008)	
Биты 15:12	0xF: Не используется
nWRP: Защита секторов 0 до 11 Flash памяти.	
Биты 11:0	nWRPi 0: Сектор защищён 1: Защиты сектора нет

Таблица 16. Описание байтов Опций для STM32F42xxx и STM32F43xxx

Байты Опций (слово, адрес 0x1FFF C000)	
RDP: Защита чтения.	
Биты 15:8	0xAA: Уровень 0, защиты нет 0xCC: Уровень 2, зашита чипа (отладка и загрузка из RAM выключены) Иное: Уровень 1, защита чтения памяти (отладка ограничена)
USER: Опции пользователя: – Событие сторожевого таймера: Аппаратное или программное – Событие сброса при входе в режим Stop – Событие сброса при входе в режим Standby	
Бит 7	nRST_STDBY 0: При входе в Standby выдаётся сброс 1: Сброс не выдаётся
Бит 6	nRST_STOP 0: При входе в Stop выдаётся сброс 1: Сброс не выдаётся
Бит 5	WDG_SW 0: Аппаратно независимый сторожевой таймер 1: Программно независимый сторожевой таймер
Бит 4	BFB2: Загрузка из двух банков 0: Из банка 1 Flash или системной памяти в (по состоянию ножек Boot, по умолчанию) 1: Всегда из системной памяти (Режим двух банков).
Биты 3:2	BOR_LEV: Уровень сброса BOR (падение питания). 00: BOR уровень 3 (VBOR3) 01: BOR уровень 2 (VBOR2) 10: BOR уровень 1 (VBOR1) 11: BOR выкл., используется уровень порога POR/PDR См. Электрические характеристики.
Биты 1:0	0x1: Не используется
Байты Опций (слово, адрес 0x1FFF C008)	
Бит 15	SPRMOD: Режим защиты для битов nWPRi 0: защита записи сектора i nWPRi (по умолчанию) 1: биты nWPRi задают защиту PCROP сектора i (Сектор)
Бит 14	DB1M: организация банков 1 Мбайт Flash памяти 0: один банк 1 Мбайт Flash (непрерывные адреса в банке 1) 1: два банка по 512 Кбайт Flash памяти. См. Таблицу 7 и Таблицу 9.
Биты 13:12	0x2: Не используется
nWRP: Защита банка 1 (Секторы 0 до 11).	

Биты 11:0	nWRPi: — SPRMOD сброшен (по умолчанию): 0: Защита записи сектора i включена. 1: Защита записи сектора i выключена. — SPRMOD стоит: 0: Защита PCROP сектора i включена. 1: Защита PCROP сектора i выключена.
Биты 15:0	0xFFFF: Не используется
Бит 15:12	0xF: Не используется
nWRP: Защита банка 2 (Секторы 12 до 23).	
Биты 11: 0	nWRPi: — SPRMOD сброшен (по умолчанию): 0: Защита записи сектора i включена. 1: Защита записи сектора i выключена. — SPRMOD стоит: 0: Защита PCROP сектора i включена. 1: Защита PCROP сектора i выключена.

3.7.2. Запись Байтов Опций Пользователя

Для работы с этим сектором надо снять бит OPTLOCK регистра FLASH_OPTCR . Для этого:

1. Пишем OPTKEY1 = 0x0819 2A3B в регистр FLASH_OPTKEYR
2. Пишем OPTKEY2 = 0x4C5D 6E7F в регистр FLASH_OPTKEYR

Восстанавливается защита программной установкой бита OPTLOCK.

Запись для STM32F405xx/07xx и STM32F415xx/17xx

Процедура:

1. Смотрим, что Flash не занята (бит BSY регистра FLASH_SR)
2. Пишем нужное в регистр FLASH_OPTCR
3. Ставим бит старта OPTSTRT регистра FLASH_CR
4. Ждём очистки бита BSY

NB: Сектор конфигурации стирается, а затем туда пишется содержимое регистра FLASH_OPTCR.

Запись для STM32F42xxx и STM32F43xxx

Байты Опций банков 1 и 2 можно изменять только одновременно. Процедура:

1. Смотрим, что Flash не занята (бит BSY регистра FLASH_SR)
2. Пишем нужное для банка 2 в регистр FLASH_OPTCR1
3. Пишем нужное для банка 1 в регистр FLASH_OPTCR
4. Ставим бит старта OPTSTRT регистра FLASH_CR
5. Ждём очистки бита BSY

NB: Сектор конфигурации стирается (банк 1 и банк 2), а затем туда пишется содержимое регистров FLASH_OPTCR и FLASH_OPTCR1.

3.7.3. Защита чтения (RDP)

Область пользователя Flash памяти имеет три уровня защиты от чтения:

- Уровень 0: без защиты
  Ставится записью 0xAA в байт опций RDP, есть доступ (запись можно запретить отдельно) к Flash памяти или резервной SRAM во всех вариантах загрузки (Flash, отладка или RAM).
- Уровень 1: защита чтения
  Это режим по умолчанию после каждого стирания байтов Опций. Ставится записью любого байта, кроме (0xAA и 0xCC) в байт опций RDP. При этом:
  - При загрузке из RAM или системной памяти, или при подключённой отладке, доступ (чтение, стирание, запись) к Flash или резервной SRAM вызывает ошибку шины.

- При загрузке из Flash памяти доступ (чтение, стирание, запись) к Flash памяти и резервной SRAM из кода пользователя разрешён.

При активном уровне 1 попытка записи в байт RDP значения уровня 0 сначала вызывает массовое стирание Flash и резервной SRAM. Стирается только область кода пользователя. Остальные байты Опций перед массовым стиранием не изменяются. Область OTP вообще не изменяется. Увеличение уровня защиты (0->1, 1->2, 0->2) к стиранию не приводит.

- Уровень 2: выключена защита чтения отладка/чип
  Включается записью 0xCC в байт опций RDP. При этом:
  - Активна защита уровня 1.
  - Загрузка из RAM или системной памяти запрещён.
  - JTAG, SWV (single-wire viewer), ETM, и скан границ выключены.
  - Байты Опций пользователя изменить нельзя.
  - При загрузке из Flash памяти доступ (чтение, стирание, запись) к Flash памяти и резервной SRAM из кода пользователя разрешён.

  Возврат из защиты чтения уровня 2 на уровень 0 или 1 невозможен.

NB: При уровне защиты 2 порт JTAG постоянно выключен (действует как предохранитель JTAG). Следовательно, скан границ невозможен. STMicroelectronics не может выполнять анализ дефектных частей с включённым уровнем защиты 2.

Таблица 17. Доступ и уровень защиты чтения

Область памяти	Уровень защиты	Отладка, Загрузка из RAM или Системной памяти			Загрузка из Flash памяти		
		Чтение	Запись	Стирание	Чтение	Запись	Стирание
Основная Flash память и Резервная SRAM	Уровень 1	НЕТ		НЕТ(1)	ДА		
	Уровень 2	НЕТ			ДА		
Байты Опций	Уровень 1	ДА			ДА		
	Уровень 2	НЕТ			НЕТ		
OTP	Уровень 1	НЕТ		NA	ДА		NA
	Уровень 2	НЕТ		NA	ДА		NA

1. Основная Flash память и Резервная SRAM при изменении уровня RDP с 1 на 0 стираются. Область OTP остаётся неизменной.

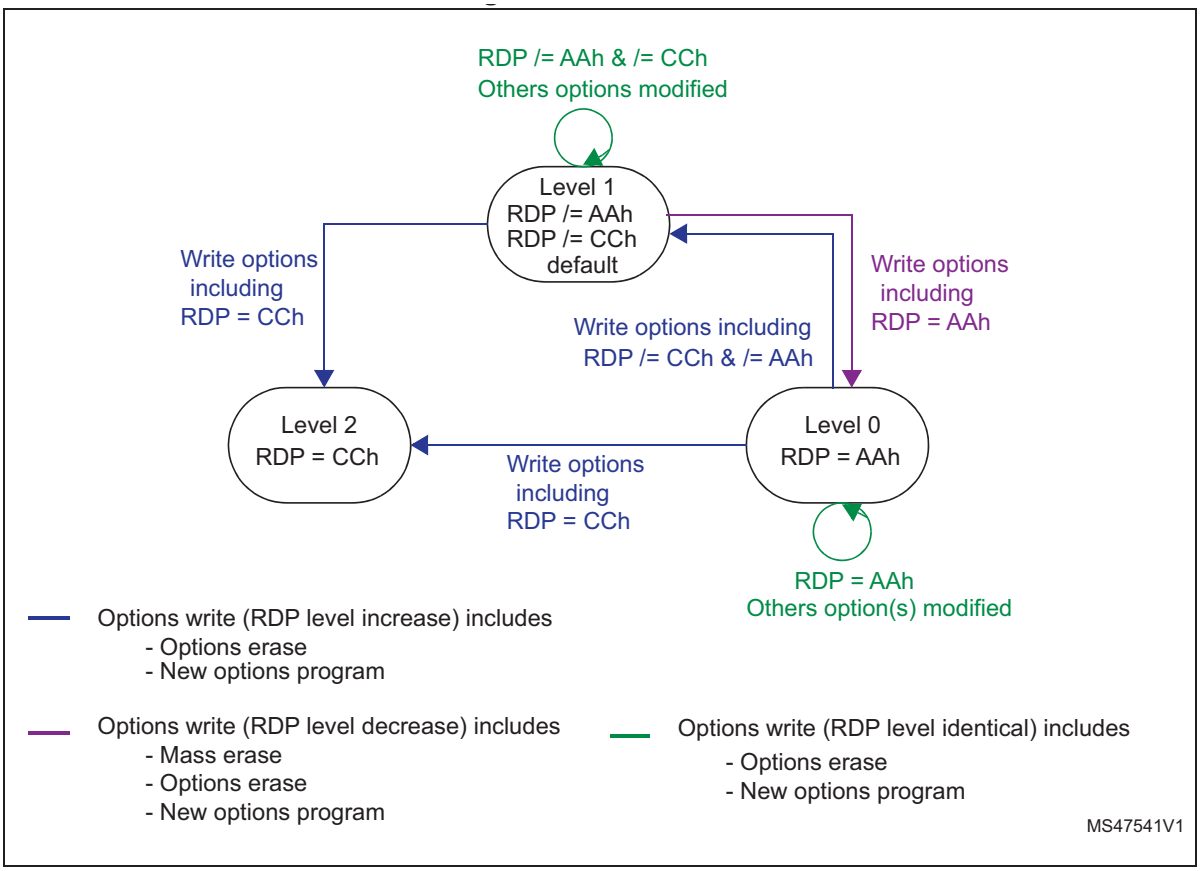


Рис. 6. Переход между уровнями RDP

3.7.4. Защита записи

Защитить от записи можно до 24 секторов пользователя Flash памяти установкой в 0 соответствующих битов `nWRPi` (0 ≤i ≤11) в регистрах `FLASH_OPTCR` или `FLASH_OPTCR1`. При установленной защите записи хоть одного сектора с установленной защитой, массовое стирание не выполняется.

Попытка стирания/записи в защищённую часть Flash памяти (сектор с защитой, блокированную часть OTP или незаписываемую часть Flash вроде ICP), в регистре `FLASH_SR` ставится флаг нарушения защиты `WRPERR`.

На STM32F42xxx и STM32F43xxx в режиме PCROP защита записи сектора включается высоким уровнем `nWRPi`. Сектор PCROP защищается автоматически.

NB: При уровне защиты чтения `RDP` равном 1 писать или стереть сектор Flash **i** невозможно если подключён зонд отладки (JTAG или SW) или загрузчик выполняется из RAM, даже при `nWRPi` = 1.

Флаг ошибки защиты записи

Флаг ошибки `WRPERR` в регистре `FLASH_SR` ставится если

Затребовано стирание:

- Массовое и банка и сектора (`MER` или `MER`/`MER1` и `SER` = 1)
- Сектора с недопустимым номером номера сектора (поле `SNB`)
- Массовое при наличии защищённого сектора (`MER` или `MER`/`MER1` = 1 и `nWRPi` = 0, где 0 ≤i ≤11 в регистре `FLASH_OPTCRx`
- Защищённого сектора. (`SER` = 1, `SNB` = i и `nWRPi` = 0, где 0 ≤i ≤11 в регистре `FLASH_OPTCRx`)
- Защищённой от чтения Flash.

Затребована запись:

- В системную память или в резервную часть особого сектора пользователя.
- В сектор конфигурации пользователя.
- В сектор, защищённый битом сектор.
- В заблокированную область OTP
- В защищённую от чтения Flash.

3.7.5. Защита от чтения собственного кода (PCROP)

Доступна на STM32F42xxx и STM32F43xxx и защищает от чтения по D-bus отмеченные секторы Flash памяти пользователя (0 до 23). Включается битом SPRMOD регистра FLASH_OPTCR:

- SPRMOD = 0: nWRPi отмечают сектора с защитой записи
- SPRMOD = 1: nWRPi отмечают сектора с защитой чтения/записи (PCROP).

Сектора с защитой PCROP доступны только через шину ICODE для выборки команд:

- Чтение защищённых секторов по D-bus ставит ошибку RDERR.
- Запись/стирание защищённых секторов ставит ошибку WRPERR.

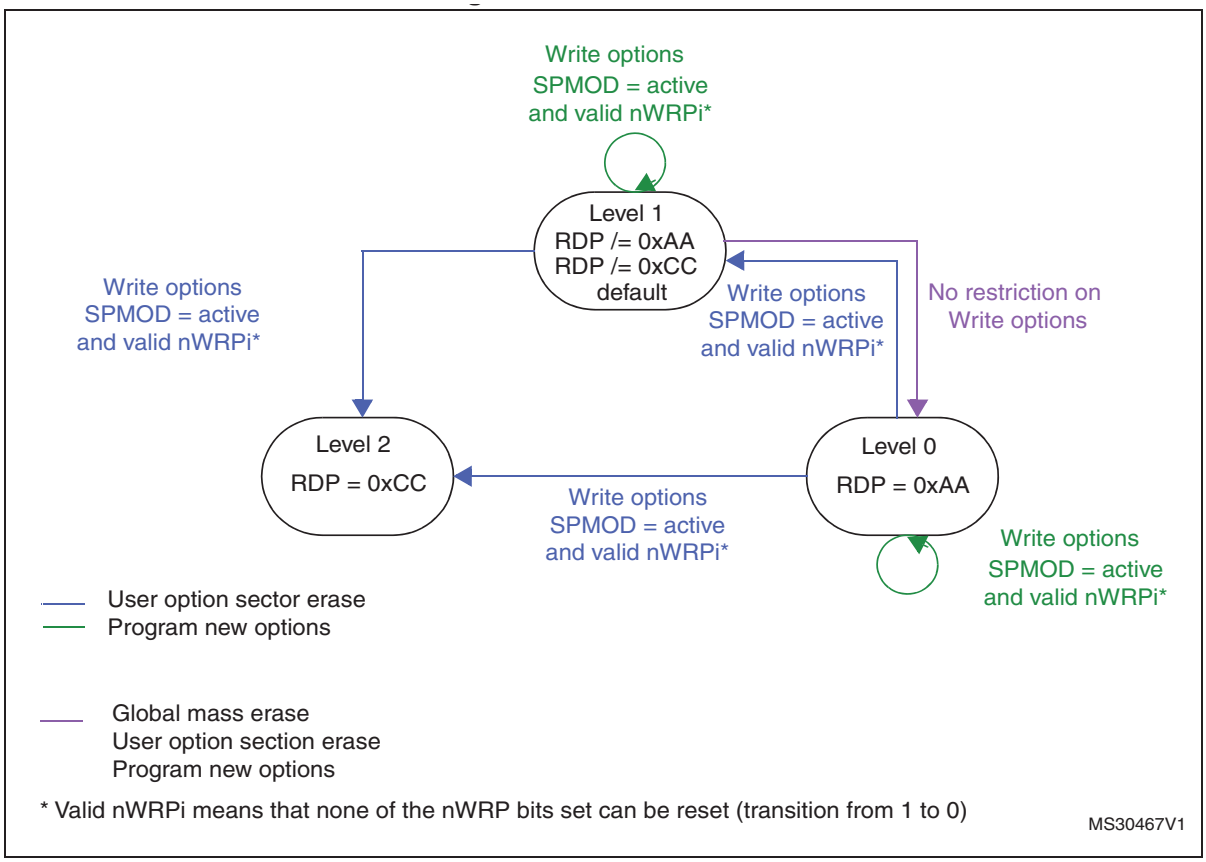


Рис 7. Уровни PCROP

Бит SPRMOD и/или защита PCROP секторов пользователя снимаются только при переходе уровня RDP из 1 в 0. Иначе байты Опций не изменяются и ставится флаг ошибки WRPERR. Изменять их (BOR_LEV, RST_STDBY, ..) только при снятых активных битах nWRPi и активном SPRMOD.

NB: В режиме PCROP (SPRMOD =1) у битов nWRPi активное значение инверсное, при SPRMOD = 1 защищены секторы с nWRPi =1.

3.8. Однократно записываемые байты (OTP)

Таблица 18. Однократно записываемая часть области OTP

Блок	[128:96]	[95:64]	[63:32]	[31:0]	Адрес начала
0	OTP0	OTP0	OTP0	OTP0	0x1FFF 7800
	OTP0	OTP0	OTP0	OTP0	0x1FFF 7810
1	OTP1	OTP1	OTP1	OTP1	0x1FFF 7820
	OTP1	OTP1	OTP1	OTP1	0x1FFF 7830
. . .	. . .				. . .
15	OTP15	OTP15	OTP15	OTP15	0x1FFF 79E0
	OTP15	OTP15	OTP15	OTP15	0x1FFF 79F0
Блок замков	LOCKB15 ... LOCKB12	LOCKB11 ... LOCKB8	LOCKB7 ... LOCKB4	LOCKB3 ... LOCKB0	0x1FFF 7A00

Область OTP разделена на 16 блоков данных 16 по 32 байта и один блок замков из 16 байтов. Стирать их нельзя. Один байт замка LOCKBi (0 ≤i ≤15) блокирует соответствующий блок данных. LOCKBi=0x00 разрешает запись в блок данных, LOCKBi=0xFF запрещает её, иначе поведение становится непонятным.

3.9. Регистры интерфейса Flash

3.9.1. Регистр управления доступом (FLASH_ACR) для STM32F405xx/07xx и STM32F415xx/17xx

Разрешает функции ускорения и управляет временем доступа в соответствии с частотой CPU.

Смещение: 0x00

По сбросу: 0x0000 0000

Доступ: без состояний ожидания, слово, полуслово, байт.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			DCRST	ICRST	DCEN	ICEN	PRFTEN	Reserved					LATENCY[2:0]		
			rw	w	rw	rw	rw						rw	rw	rw

— Биты 31:13 Резерв, изменять нельзя.

— Бит 12 **DCRST**: Сброс кэша данных

Пишут при отключенном кэше D.

0: Ничего.

1: Сброс.

— Бит 11 **ICRST**: Сброс кэша команд

Пишут при отключенном кэше I.

0: Ничего.

1: Сброс.

— Бит 10 **DCEN**: Включение кэша данных

0: Выкл.

1: Вкл.

— Бит 9 **ICEN**: Включение кэша команд

0: Выкл.

1: Вкл.

— Бит 8 **PRFTEN**: Включение предвыборки команд команд

0: Выкл.

1: Вкл.

— Биты 7:3 Резерв, изменять нельзя.

— Биты 2:0 **LATENCY[2:0]**: число циклов ожидания (0 до 7)

Это отношение периода SYSCLK и времени доступа к флэш памяти.

3.9.2. Регистр управления доступом (FLASH_ACR) для STM32F42xxx и STM32F43xxx

Разрешает функции ускорения и управляет временем доступа в соответствии с частотой CPU.

Смещение: 0x00

По сбросу: 0x0000 0000

Доступ: без состояний ожидания, слово, полуслово, байт.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			DCRST	ICRST	DCEN	ICEN	PRFTEN	Reserved				LATENCY[3:0]			
			rw	w	rw	rw	rw					rw	rw	rw	rw

— Биты 31:13 Резерв, изменять нельзя.

— Бит 12 **DCRST**: Сброс кэша данных
Пишут при отключенном кэше D.
0: Ничего.
1: Сброс.

— Бит 11 **ICRST**: Сброс кэша команд
Пишут при отключенном кэше I.
0: Ничего.
1: Сброс.

— Бит 10 **DCEN**: Включение кэша данных
0: Выкл.
1: Вкл.

— Бит 9 **ICEN**: Включение кэша команд
0: Выкл.
1: Вкл.

— Бит 8 **PRFTEN**: Включение предвыборки команд
0: Выкл.
1: Вкл.

— Биты 7:3 Резерв, изменять нельзя.

— Биты 2:0 **LATENCY[3:0]**: число циклов ожидания (0 до 15)
Это отношение периода SYSCLK и времени доступа к флэш памяти.

3.9.3. Регистр ключа (FLASH_KEYR)

Разрешает доступ к регистру управления FLASH_CR.

Смещение: 0x04

По сбросу: 0x0000 0000

Доступ: без состояний ожидания, слово.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

— Биты 31:0 **KEY[31:0]**: Ключ снятия защиты
Последовательно пишут:
a) KEY1 = 0x45670123
b) KEY2 = 0xCDEF89AB

3.9.4. Регистр ключа опций (FLASH_KEYR)

Разрешает запись и стирание сектора конфигурации пользователя (FLASH_OPTCR).

Смещение: 0x08

По сбросу: 0x0000 0000

Доступ: без состояний ожидания, слово.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OPTKEYR[31:16															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPTKEYR[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

— Биты 31:0 **OPTKEYR[31:0]**: Ключ снятия защиты
Последовательно пишут:
a) OPTKEY1 = 0x08192A3B
b) OPTKEY2 = 0x4C5D6E7F

3.9.5. Регистр состояния (FLASH_SR) для STM32F405xx/07xx и STM32F415xx/17xx

Выдаёт состояние операций записи и стирания Flash.

Смещение: 0x0C

По сбросу: 0x0000 0000

Доступ: без состояний ожидания, слово, полуслово, байт.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															BSY
															r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								PGSERR	PGPERR	PGAERR	WRPERR	Reserved		OPERR	EOP
								rc_w1	rc_w1	rc_w1	rc_w1			rc_w1	rc_w1

— Биты 31:17 Резерв, изменять нельзя.

— Бит 16 **BSY**: Занято

0: Свободно.

1: Занято.

— Биты 15:8 Резерв, изменять нельзя.

— Бит 7 **PGSERR**: Ошибка последовательности записи

Ставится аппаратно при попытке записи с неверной конфигурацией регистра управления. Снимается записью 1.

— Бит 6 **PGPERR**: Ошибка параллелизма записи

Ставится аппаратно при несоответствии размера записи (байт, полуслово, слово) с конфигурацией PSIZE (x8, x16, x32, x64). Снимается записью 1.

— Бит 5 **PGAERR**: Ошибка выравнивания записи

Ставится аппаратно при нарушении 128-бит границы записи. Снимается записью 1.

— Бит 4 **WRPERR**: Нарушение защиты записи

Ставится аппаратно при записи в защищённую область. Снимается записью 1.

— Биты 3:2 Резерв, изменять нельзя.

— Бит 1 **OPERR**: Ошибка операции

Ставится аппаратно при появлении любой ошибки и разрешённом прерывании ошибки (ERRIE = 1).

— Бит 0 **EOP**: Конец операции

Ставится аппаратно при успешном завершении стирания/записи и разрешённом прерывании ошибки (ERRIE = 1). Снимается записью 1.

3.9.6. Регистр состояния (FLASH_SR) для STM32F42xxx и STM32F43xxx

Выдаёт состояние операций записи и стирания Flash.

Смещение: 0x0C

По сбросу: 0x0000 0000

Доступ: без состояний ожидания, слово, полуслово, байт.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															BSY
															r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							RDERR	PGSERR	PGPERR	PGAERR	WRPERR	Reserved		OPERR	EOP
							rc_w1	rc_w1	rc_w1	rc_w1	rc_w1			rc_w1	rc_w1

— Биты 31:17 Резерв, изменять нельзя.

— Бит 16 **BSY**: Занято

0: Свободно.

1: Занято.

— Биты 15:9 Резерв, изменять нельзя.

— Бит 8 **RDERR**: Нарушение защиты чтения (PCROP)

Ставится аппаратно при попытке чтения по шине D-bus защищённой памяти. Снимается записью 1.

— Бит 7 **PGSERR**: Ошибка последовательности записи
Ставится аппаратно при попытке записи с неверной конфигурацией регистра управления. Снимается записью 1.

— Бит 6 **PGPERR**: Ошибка параллелизма записи
Ставится аппаратно при несоответствии размера записи (байт, полуслово, слово) с конфигурацией PSIZE (x8, x16, x32, x64). Снимается записью 1.

— Бит 5 **PGAERR**: Ошибка выравнивания записи
Ставится аппаратно при нарушении 128-бит границы записи. Снимается записью 1.

— Бит 4 **WRPERR**: Нарушение защиты записи
Ставится аппаратно при записи в защищённую область. Снимается записью 1.

— Биты 3:2 Резерв, изменять нельзя.

— Бит 1 **OPERR**: Ошибка операции
Ставится аппаратно при появлении любой ошибки и разрешённом прерывании ошибки (ERRIE = 1).

— Бит 0 **EOP**: Конец операции
Ставится аппаратно при успешном завершении стирания/записи и разрешённом прерывании ошибки (ERRIE = 1). Снимается записью 1.

3.9.7. Регистр управления (FLASH_CR) для STM32F405xx/07xx и STM32F415xx/17xx

Конфигурация и запуск операций Flash.

Смещение: 0x10

По сбросу: 0x8000 0000

Доступ: без состояний ожидания для не занятой памяти, слово, полуслово, байт.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	Reserved					ERRIE	EOPIE	Reserved							STRT
rs						rw	rw								rs

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						PSIZE[1:0]		Res.	SNB[3:0]				MER	SER	PG
						rw	rw		rw	rw	rw	rw	rw	rw	rw

— Бит 31 **LOCK**: Блокировка регистра FLASH_CR
Ставится только записью 1. Снимается аппаратно при последовательности разблокировки.

— Биты 30:26 Резерв, изменять нельзя.

— Бит 25 **ERRIE**: Разрешение прерывания ошибки по OPERR
0: Нельзя.
1: Можно.

— Бит 24 **EOPIE**: Разрешение прерывания конца операции по EOP
0: Нельзя.
1: Можно.

— Биты 23:17 Резерв, изменять нельзя.

— Бит 16 **STRT**: Запуск операции стирания
Ставится программно. Снимается аппаратно при очистке бита BSY.

— Биты 15:10 Резерв, изменять нельзя.

— Биты 9:8 **PSIZE[1:0]**: Размер записи (параллелизм)
00: x8.
01: x16.
10: x32.
11: x64.

— Бит 7 Резерв, изменять нельзя.

— Биты 6:3 **SNB[3:0]**: Номер сектора (0 - 11), остальные нельзя.

— Бит 2 **MER**: Операция массового стирания всех секторов пользователя.

— Бит 1 **SER**: Операция стирания одного сектора.

— Бит 0 **PG**: Операция записи

3.9.8. Регистр управления (FLASH_CR) для STM32F42xxx и STM32F43xxx

Конфигурация и запуск операций Flash.

Смещение: 0x10

По сбросу: 0x8000 0000

Доступ: без состояний ожидания для не занятой памяти, слово, полуслово, байт.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOCK	Reserved					ERRIE	EOPIE	Reserved							STRT
rs						rw	rw								rs

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MER1	Reserved					PSIZE[1:0]		SNB[4:0]					MER	SER	PG
rw						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Бит 31 **LOCK**: Блокировка регистра FLASH_CR
Ставится только записью 1. Снимается аппаратно при последовательности разблокировки. При неудачной разблокировке остаётся стоять до следующего сброса.

— Биты 30:26 Резерв, изменять нельзя.

— Бит 25 **ERRIE**: Разрешение прерывания ошибки по OPERR
0: Нельзя.
1: Можно.

— Бит 24 **EOPIE**: Разрешение прерывания конца операции по EOP
0: Нельзя.
1: Можно.

— Биты 23:17 Резерв, изменять нельзя.

— Бит 16 **STRT**: Запуск операции стирания
Ставится программно. Снимается аппаратно при очистке бита BSY.

— Бит 15 **MER1**: Операция массового стирания секторов банка 2 (12 - 23).

— Биты 14:10 Резерв, изменять нельзя.

— Биты 9:8 **PSIZE[1:0]**: Размер записи (параллелизм)
00: x8.
01: x16.
10: x32.
11: x64.

— Биты 7:3 **SNB[4:0]**: Номер сектора.
00000 - 11011: 0 - 11.
10000 - 111011: 12 - 23.
Остальные нельзя.

— Бит 2 **MER**: Операция массового стирания секторов банка 1 (0 - 11).

— Бит 1 **SER**: Операция стирания одного сектора.

— Бит 0 **PG**: Операция записи

3.9.9. Регистр управления опциями (FLASH_OPTCR) для STM32F405xx/07xx и STM32F415xx/17xx

Модификация байтов опций пользователя.

Смещение: 0x14

По сбросу: 0x0FFF AAED Пишутся из Flash памяти по концу сброса.

Доступ: без состояний ожидания для не занятой памяти, слово, полуслово, байт.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				nWRP[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDP[7:0]								nRST_STDBY	nRST_STOP	WDG_SW	Reserved	BOR_LEV		OPTSTRT	OPTLOCK
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rs	rs

— Биты 31:28 Резерв, изменять нельзя.

— Биты 27:16 **nWRP[11:0]**: Не-защита записи соответствующих секторов.

Содержимое байтов опций защиты записи после сброса. Новые значения пишутся отсюда в Flash.

0: Защита есть.

1: Защиты нет.

— Биты 15:8 **RDP[7:0]**: Защита чтения.

Содержимое уровня защиты чтения после сброса. Новые значения пишутся отсюда в Flash.

0xAA: Уровень 0, защиты чтения нет.

0xCC: Уровень 2, защита чтения чипа есть.

Иное: Уровень 1, защита чтения памяти есть

USER[2:0]: Байты опций пользователя

Содержимое байтов опций защиты пользователя после сброса. Новые значения пишутся отсюда в Flash.

NB: Изменение режима WDG начинает действовать после сброса.

— Бит 7 **nRST_STDBY**

— Бит 6 **nRST_STOP**

— Бит 5 **WDG_SW**

— Бит 4 Резерв, изменять нельзя.

— Биты 3:2 **BOR_LEV[1:0]**: Уровень порога для сброса по падению питания.

Содержимое байтов опций защиты пользователя после сброса. Новые значения пишутся отсюда в Flash.

00: BOR Level 3 (VBOR3)

01: BOR Level 2 (VBOR2)

10: BOR Level 1 (VBOR1)

11: BOR выкл., POR/PDR (по умолчанию)

— Бит 1 **OPTSTRT**: Запуск операции опций.

Пишется программно, снимается аппаратно при очистке бита BSY.

— Бит 0 **OPTLOCK**: Блокировка опций.

Пишется только 1, снимается последовательностью разблокировки. После неудачной попытки остаётся стоять вплоть до сброса.

3.9.10. Регистр управления опциями (FLASH_OPTCR) для STM32F42xxx и STM32F43xxx

Модификация байтов опций пользователя.

Смещение: 0x14

По сбросу: 0x0FFF AAED Пишутся из Flash памяти по концу сброса.

Доступ: без состояний ожидания для не занятой памяти, слово, полуслово, байт.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPR MOD	DB1M	Reserved		nWRP[11:0]											
rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDP[7:0]								nRST_ STDBY	nRST_ STOP	WDG_ SW	BFB2	BOR_LEV		OPTST RT	OPTLO CK
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rs	rs

— Бит 31 **PRMOD**: Выбор режима работы битов nWPRi.

0: PCROP выкл., nWPRi - защита записи сектора i.

1: PCROP вкл., nWPRi - защита PCROP сектора i.

— Бит 30 **DB1M**: Два банка в 1 Мбайте Flash памяти.

0: 1 Мбайт одним банком (последовательные адреса банка 1).

1: Два банка по 512 Кбайт. Для стирания в банке 2 нужен правильный номер сектора.

— Биты 29:28 Резерв, изменять нельзя.

— Биты 27:16 **nWRP[11:0]**: Не-защита записи/чтения (PCROP) соответствующих секторов.

Содержимое байтов опций защиты записи после сброса. Новые значения пишутся отсюда в Flash.

Если SPRMOD снят:

0: Защита записи сектора i есть

1: Защиты записи сектора i нет

Если SPRMOD стоит:

0: PCROP защиты сектора i нет

1: PCROP защита сектора i есть

— Биты 15:8 **RDP[7:0]**: Защита чтения.

Содержимое уровня защиты чтения после сброса. Новые значения пишутся отсюда в Flash.

0xAA: Уровень 0, защиты чтения нет.

0xCC: Уровень 2, защита чтения чипа есть.

Иное: Уровень 1, защита чтения памяти есть

USER[2:0]: Байты опций пользователя

Содержимое байтов опций защиты пользователя после сброса. Новые значения пишутся отсюда в Flash.

NB: Изменение режима WDG начинает действовать после сброса.

— Бит 7 **nRST_STDBY**

— Бит 6 **nRST_STOP**

— Бит 5 **WDG_SW**

— Бит 4 **BFB2**: Загрузка из двух банков.

0: Выкл., грузят из банка 1 или системной памяти определено ножками (по умолчанию)

1: Вкл., грузят всегда из системной памяти.

NB: В STM32F42xxx и STM32F43xxx с 1MB номерами, должна быть нулевой при DB1M=0.

— Биты 3:2 **BOR_LEV[1:0]**: Уровень порога для сброса по падению питания.

Содержимое байтов опций защиты пользователя после сброса. Новые значения пишутся отсюда в Flash.

00: BOR Level 3 (VBOR3)

01: BOR Level 2 (VBOR2)

10: BOR Level 1 (VBOR1)

11: BOR выкл., POR/PDR (по умолчанию)

— Бит 1 **OPTSTRT**: Запуск операции опций.

Пишется программно, снимается аппаратно при очистке бита BSY.

— Бит 0 **OPTLOCK**: Блокировка опций.

Пишется только 1, снимается последовательностью разблокировки. После неудачной попытки остаётся стоять вплоть до сброса.

3.9.11. Регистр управления опциями (FLASH_OPTCR1) для STM32F42xxx и STM32F43xxx

Модификация байтов опций пользователя для банка 2.

Смещение: 0x18

По сбросу: 0x0FFF 0000 Пишутся из Flash памяти по концу сброса.

Доступ: без состояний ожидания для не занятой памяти, слово, полуслово, байт.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				nWRP[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															

— Биты 31:28 Резерв, изменять нельзя.

— Биты 27:16 **nWRP[11:0]**: Не-защита записи/чтения (PCROP) соответствующих секторов.

Содержимое байтов опций защиты записи после сброса. Новые значения пишутся отсюда в Flash.

Если SPRMOD снят:

0: Защита записи сектора i есть

1: Защиты записи сектора i нет

Если SPRMOD стоит:

0: PCROP защиты сектора i нет

1: PCROP защита сектора i есть

— Биты 15:0 Резерв, изменять нельзя.

3.9.12. Карта регистров Flash интерфейса

Table 19. Для STM32F405xx/07xx и STM32F415xx/17xx

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**FLASH_ACR**	Reserved																			DCRST	ICRST	DCEN	ICEN	PRFTEN	Reserved						LATENCY [2:0]		
	Reset value																				0	0	0	0	0							0	0	0
0x04	**FLASH_ KEYR**	KEY[31:16]																KEY[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**FLASH_OPT KEYR**	OPTKEYR[31:16]																OPTKEYR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**FLASH_SR**	Reserved															BSY	Reserved								PGSERR	PGPERR	PGAERR	WRPERR	Reserved		OPERR	EOP
	Reset value																0									0	0	0	0			0	0
0x10	**FLASH_CR**	LOCK	Reserved							EOPIE	Reserved						STRT	Reserved					PSIZE[1:0]		Reserved	SNB[3:0]				MER	SER	PG	
	Reset value	1								0							0						0	0		0	0	0	0		0	0	
0x14	**FLASH_ OPTCR**	Reserved								nWRP[11:0]								RDP[7:0]								nRST_STDBY	nRST_STOP	WDG_SW	Reserved	BOR_LEV[1:0]		OPTSTRT	OPTLOCK
	Reset value									1	1	1	1	1	1	1	1	1	0	1	0	1	0	1	0	1	1	1		1	1	0	1

Table 19. Для STM32F42xxx и STM32F43xxx

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**FLASH_ACR**	Reserved																			DCRST	ICRST	DCEN	ICEN	PRFTEN	Reserved				LATENCY[3:0]			
	Reset value																				0	0	0	0	0					0	0	0	0
0x04	**FLASH_KEYR**	KEY[31:16]																KEY[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**FLASH_ OPTKEYR**	OPTKEYR[31:16]																OPTKEYR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**FLASH_SR**	Reserved															BSY	Reserved							RDERR	PGSERR	PGPERR	PGAERR	WRPERR	Reserved		OPERR	EOP
	Reset value																0								0	0	0	0	0			0	0
0x10	**FLASH_CR**	LOCK	Reserved							EOPIE	Reserved						STRT	MER1	Reserved				PSIZE[1:0]		SNB[4:0]					MER	SER	PG	
	Reset value	1								0							0	0					0	0	0	0	0	0	0	0	0	0	
0x14	**FLASH_OPTCR**	SPRMOD	DB1M	Reserved	nWRP[11:0]													RDP[7:0]								nRST_STDBY	nRST_STOP	WDG_SW	BFB2	BOR_LEV[1:0]		OPTSTRT	OPTLOCK
	Reset value	0	0		1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1	0	1	0	1	1	1	0	1	1	0	1
0x18	**FLASH_ OPTCR1**	Reserved			nWRP[11:0]													Reserved																
	Reset value				1	1	1	1	1	1	1	1	1	1	1	1	1																

4. Блок вычисления CRC

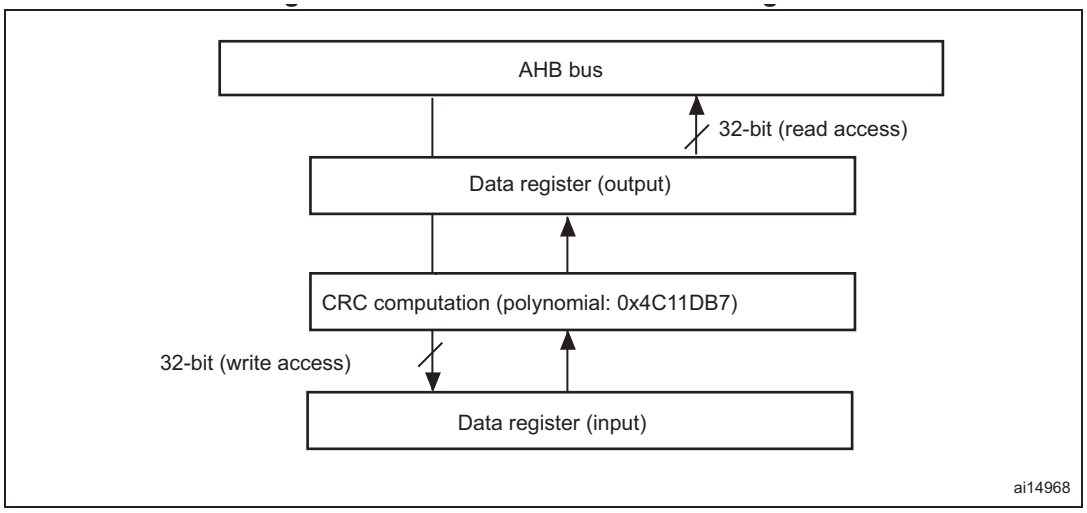
4.1. Введение в CRC

Как и другие алгоритмы стандарта EN/IEC 60335-1, техника CRC используется для проверки целостности передачи и хранения данных. Блок вычисления CRC помогает получить сигнатуру программы при выполнении и сравнить её с полученной при компоновке и хранящейся по определённому адресу.

4.2. Основные свойства CRC

- Использует CRC-32 (Ethernet) полином: 0x4C11DB7
  $-X^{32}+X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^{8}+X^{7}+X^{5}+X^{4}+X^{2}+X+1$
- Один 32-бит регистр ввода/вывода данных
- CRC вычисляется за 4 такта AHB (HCLK)
- Доступный 8-бит регистр общего назначения для временного хранения

Блок-схема.



4.3. Функциональное описание CRC

Блок вычисления CRC в основном состоит из одного 32-бит регистра данных, который:

- при записи получает новое данное для калькулятора CRC
- при чтении содержит результат предыдущего вычисления

Каждая запись в регистр данных создаёт новый результат с предыдущим CRC (CRC вычисляется 32-бит словами данных целиком, а не байт за байтом).

Операция записи задерживается до конца вычисления CRC, что позволяет и непрерывные операции записи и чередовать запись и чтение.

Калькулятор CRC сбрасывается в 0xFFFF FFFF битом `RESET` в регистре `CRC_CR`. Эта операция не влияет на содержимое регистра `CRC_IDR`.

4.4. Регистры CRC

Блок вычисления CRC содержит два регистра данных и регистр управления, доступных как 32-бит слова.

4.4.1. Регистр данных (`CRC_DR`)

При записи получает новое данное, при чтении содержит результат предыдущей операции.

Смещение адреса: 0x00

По сбросу: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DR [31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR [15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

4.4.2. Независимый регистр данных (CRC_IDR)

Смещение адреса: 0x04

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								IDR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Биты 31:8 Резерв, изменять нельзя.

Биты 7:0 8-бит регистр общего назначения. Сброс CRC битом RESET в регистре CRC_CR на него не влияет.

4.4.3. Регистр управления (CRC_CR)

Смещение адреса: 0x08

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															RESET
															w

— Биты 31:8 Резерв, изменять нельзя.

— Бит 7:0 **RESET** Запись 1 сбрасывает CRC и пишет в CRC_DR число 0xFFFF FFFF. Очищается автоматически.

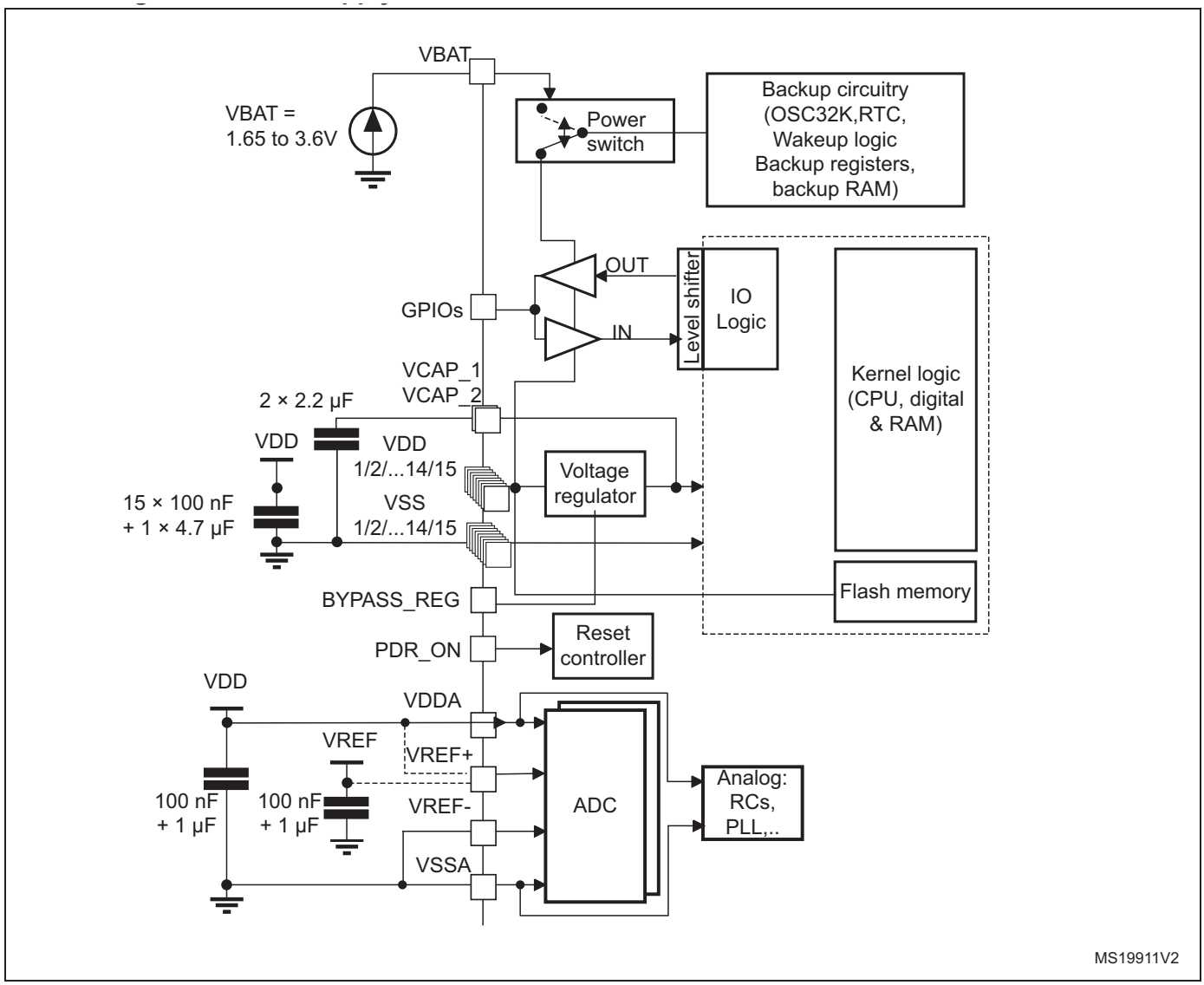
4.4.4. Карта регистров CRC

Offset	Register	31-24	23-16	15-8	7	6	5	4	3	2	1	0
0x00	**CRC_DR**	Data register										
	Reset value	0xFFFF FFFF										
0x04	**CRC_IDR**	Reserved			Independent data register							
	Reset value				0x00							
0x08	**CRC_CR**	Reserved										RESET
	Reset value											0

5. Управление питанием PWR

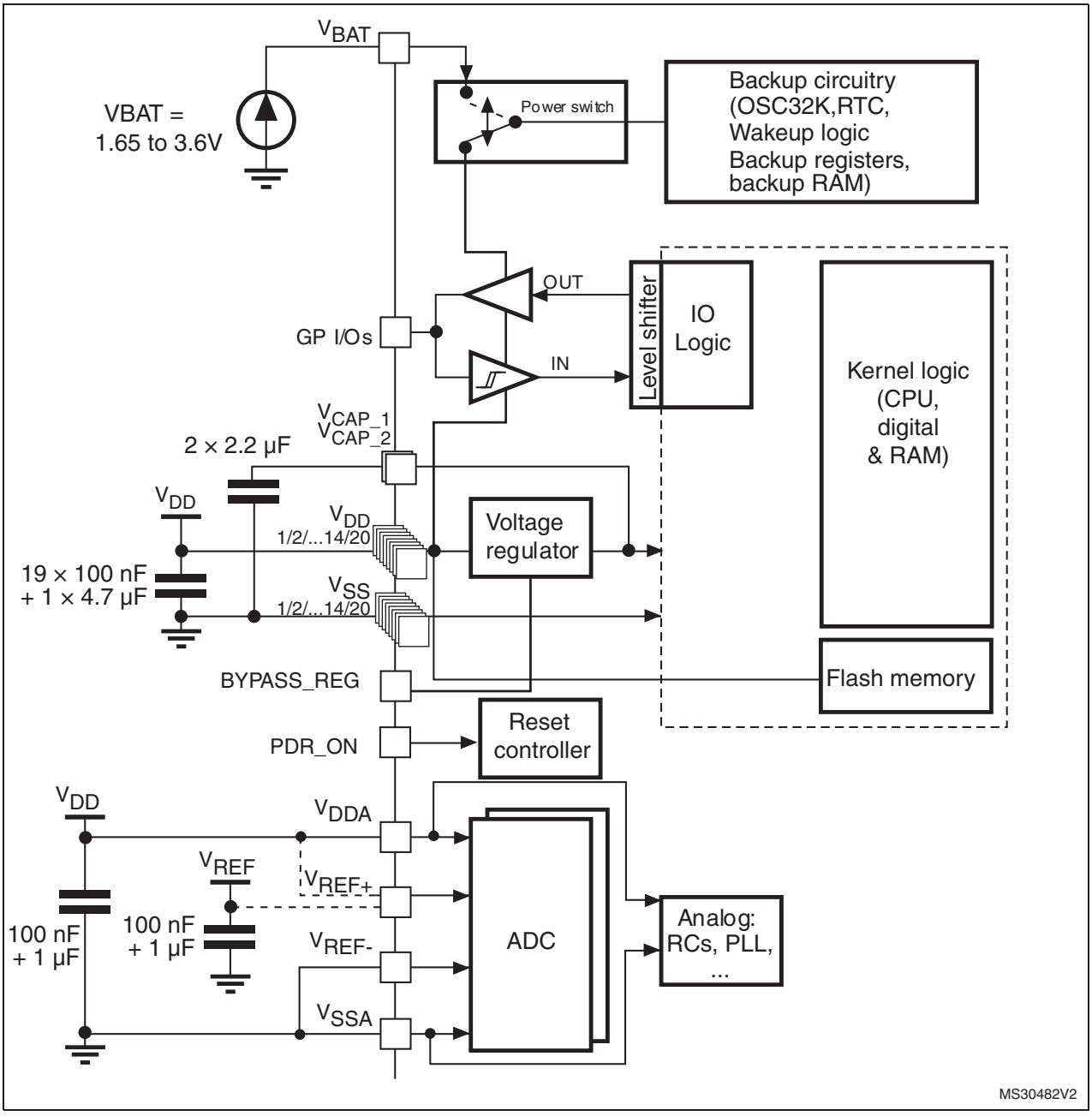
5.1. Источники питания

Устройство требует напряжения от 2.0 до 3.6 V (V_{DD}). Встроенный регулятор даёт 1.8 V питания. Часы реального времени (RTC) и регистры хранения могут питаться от V_{BAT} при выключенном V_{DD}.



1. V_{DDA} and V_{SSA} must be connected to V_{DD} and V_{SS}, respectively.

Рис. 9. Источник питания для STM32F405xx/07xx и STM32F415xx/17xx



1. V_{DDA} and V_{SSA} must be connected to V_{DD} and V_{SS}, respectively.

Рис. 10. Источник питания для STM32F42xxx и STM32F43xxx

5.1.1. Независимое питание ADC и опорное напряжение

Для повышения точности преобразования, ADC имеет независимое питание, которое можно отдельно фильтровать и защитить от шума PCB.

- Питание ADC и DAC подаётся на отдельную ножку V_{DDA}.
- Земля изолированного источника подключается к ножке V_{SSA}.

Для повышения точности преобразования малых сигналов на V_{REF} ADC можно подавать внешнее опорное напряжение от 1.8 V до V_{DDA}.

5.1.2. Резервное батарейное питание

Для сохранения содержимого регистров хранения и работы RTC при выключенном V_{DD} ножку V_{BAT} можно подключать к необязательному источнику питания от батареи или иному источнику.

Ножка V_{BAT} питает:

- блок RTC,
- генератор LSE
- резервную SRAM при включённом регуляторе режимов низкого потребления
- контакты от PC13 до PC15 IO и PI8 I/O (если есть).

Включение питания V_{BAT} управляется Power Down Reset в блоке Сброса.

Внимание:

Во время $t_{RSTTEMPO}$ (ожидание при запуске а V_{DD}) или после обнаружения PDR переключатель питания между V_{BAT} и V_{DD} остаётся подключённым к V_{BAT}.

Во время фазы запуска, если V_{DD} меньше чем $t_{RSTTEMPO}$ и $V_{DD} > V_{BAT} + 0.6$ V, ток в V_{BAT} может подаваться через внутренний диод между V_{DD} и переключателем питания (V_{BAT}).

Если подключённый к V_{BAT} источник/батарея не может обеспечить подачу этого тока, то весьма рекомендуется между источником и V_{BAT} подключить внешний диод с низким падением напряжения.

Если внешней батареи нет, то рекомендуется подключать V_{BAT} к V_{DD} через внешний керамический 100 nF конденсатор развязки в параллель.

При питании домена резервного хранения от V_{DD} (аналоговый ключ подключён к V_{DD}) доступны следующие функции:

- PC14 и PC15 можно использовать как GPIO или LSE
- PC13 можно использовать как GPIO, ножку RTC_AF1 см. Таблицу 37.

NB: Из-за того, что ключ пропускает только 3 mA тока, использование GPIO от PC13 до PC15 на вывод ограничено: скорость надо ограничить до 2 MHz с максимальной нагрузкой 30 pF и эти IO нельзя использовать как источник тока (например, управление светодиодами).

При питании домена резервного хранения от V_{BAT} (V_{DD} нету) доступны следующие функции:

- PC14 и PC15 можно использовать только как контакты LSE
- PC13 можно использовать как ножку RTC_AF1. См. *Таблицу 37*).
- PI8 можно использовать как RTC_AF2

Доступ к резервному домену

После сброса резервный домен (регистры RTC, резервный регистр RTC и резервная SRAM) защищены от записи. Для разрешения доступа надо:

- К регистрам RTC и резервному регистру RTC
  - Пускаем такты интерфейса установкой битов `PWREN` регистра `RCC_APB1ENR` (см. 7.3.13 и 6.3.13)
  - Ставим бит `DBP` регистра `PWR_CR` (см. 5.4.1) для разрешения доступа к домену
  - Выбираем источник тактов RTC (см. 7.2.8)
  - Пускаем такты RTC битом `RTCEN [15]` регистра `RCC_BDCR` (см. 7.3.20)
- К резервной SRAM
  - Пускаем такты интерфейса установкой битов `PWREN` регистра `RCC_APB1ENR` (см. 7.3.13 и 6.3.13)
  - Ставим бит `DBP` регистра `PWR_CR` для разрешения доступа к домену
  - Пускаем такты резервной SRAM установкой бита `BKPSRAMEN` регистра `RCC_AHB1ENR`.

Регистры RTC

Часы реального времени (RTC) это независимый BCD таймер/счётчик. RTC имеет счётчик времени суток/календарь, два программируемых прерывания будильника и периодический программируемый флаг побудки с прерыванием. RTC содержит 20 резервных регистров данных (80 байт), которые сбрасываются при обнаружении внешнего проникновения (см. раздел 26.).

Резервная SRAM

Она включает 4 Кбайта с 32-бит, 16-бит и 8-бит адресацией. Содержимое сохраняется в режиме Standby или V_{BAT} при включённом регуляторе напряжения. При включённом V_{BAT} её можно рассматривать как внутреннюю EEPROM.

При питании домена от V_{DD} (аналоговый ключ подключён к V_{DD}), резервная SRAM тоже питается от V_{DD} вместо V_{BAT}.

При питании домена от V_{BAT}, резервная SRAM питается через специальный регулятор напряжения, который можно отключать в случае ненадобности. Этим управляет бит `BRE` регистра `PWR_CSR`.

При шпионском внедрении резервная SRAM не стирается. От чтения она защищается вместе с Flash памятью. При изменении уровня защиты с уровня 1 на уровень 0 она стирается.

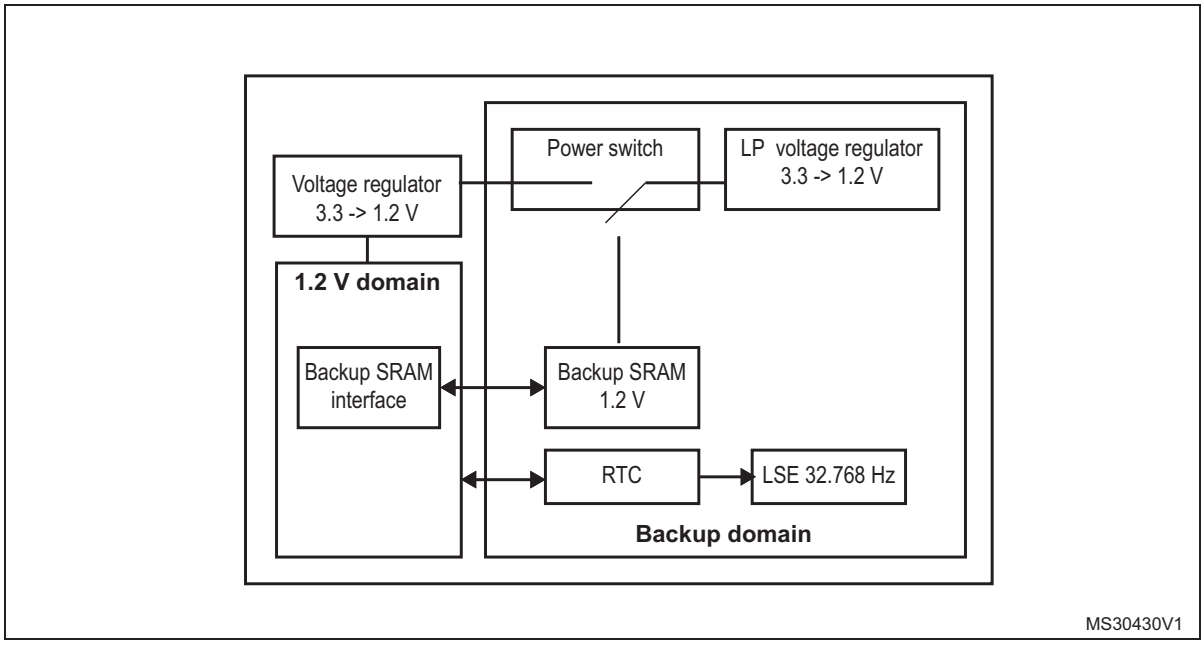


Рис. 11. Резервный домен

5.1.3. Регулятор напряжения для STM32F405xx/07xx and STM32F415xx/17xx

Встроенный линейный регулятор напряжения с выходным напряжением около 1.2 V питает все цифровые схемы, кроме резервного домена и схем Standby.

Ему нужны два внешних конденсатора, подключённых к контактам V_{CAP_1} и V_{CAP_2}. Для включения/выключения регулятора к V_{SS} или V_{DD} надо подключать специальные ножки. Они зависят от корпуса.

При программном включении, регулятор после сброса всегда включён. Он может работать в трёх режимах.

- В режиме **Run**, он подаёт полное питание 1.2 V на домен (ядро, память и цифровую периферию). В этом выходное напряжение (около 1.2 V) может иметь два значения, которые задаются на лету через `VOS` (бит 15 регистра `PWR_CR`). Это позволяет снижать потребление при пониженной частоте тактов.
- В режиме **Stop**, 1.2 V домен питают основной (MR) или экономичный регулятор (LPR), сохраняя регистры и внутреннюю SRAM. Значения напряжения остаются теми же (см. *5.4.1*).
- В режиме **Standby**, регулятор выключен. Регистры и SRAM теряются, кроме схем Standby и резервного домена.

5.1.4. Регулятор напряжения для STM32F42xxx и STM32F43xxx

Встроенный линейный регулятор напряжения с выходным напряжением около 1.2 V питает все цифровые схемы, кроме резервного домена и схем Standby.

Ему нужны два внешних конденсатора, подключённых к контактам V_{CAP_1} и V_{CAP_2}. Для включения/выключения регулятора к V_{SS} или V_{DD} надо подключать специальные ножки. Они зависят от корпуса.

При программном включении, регулятор после сброса всегда включён. Он может работать в трёх режимах.

- В режиме **Run**, он подаёт полное питание 1.2 V на домен (ядро, память и цифровую периферию). В этом выходное напряжение (около 1.2 V) может иметь три значения, которые задаются через биты `VOS[1:0]` регистра `PWR_CR`. Это позволяет снижать потребление при пониженной частоте тактов. Значение напряжения можно менять только при выключенном PLL и тактировании HSI или HSE системными тактами. Новое значение вступает в действие при включении PLL. При выключенном PLL и выходе из режима Stop действует выходное напряжение 3.
  Есть 2 режима работы:

  - **Нормальный**: CPU и ядро работают на максимальной частоте с заданным напряжением
  - **Усиленный**: CPU и ядро работают на повышенной частоте с напряжением 1 или 2.
- В режиме **Stop**: 1.2 V домен питают основной (MR) или экономичный регулятор (LPR), сохраняя регистры и внутреннюю SRAM. Значения напряжения остаются теми же.
  MR или LPR выбираются программно:
  - **Нормальный**: 1.2 V домен стоит в режиме номинальной утечки. Это режим по умолчанию при включённом MR или LPR.
  - **Ослабленный**: 1.2 V домен стоит в режиме уменьшенной утечки. Режим доступен только с MR или LPR (см. Таблицу 22).
- В режиме **Standby**: Регулятор выключен. Регистры и SRAM теряются, кроме схем Standby и резервного домена. Усиленный и ослабленный режимы недоступны.

Таблица 22. Регулятор напряжения и режим работы устройства(1)

Конфигурация	Run	Sleep	Stop	Standby
Нормальный	MR	MR	MR or LPR	-
Усиленный(2)	MR	MR	-	-
Ослабленный	-	-	MR or LPR	-
Выключено	-	-	-	Yes

1. '-' значит недоступно.
2. Усиленный режим недоступен при V_{DD} = 1.8 до 2.1 V.

Вход в усиленный режим

Рекомендуем переключаться в Усиленный режим при тактировании системы от HSI или HSE когда программа не выполняет критических задач. Включаться будет быстрее во время фазы блокировки PLL.

Последовательность:

1) Выбираем системные такты от HSI или HSE.
2) Конфигурируем RCC_PLLCFGR и ставим бит PLLON регистра RCC_CR.
3) Включаем Усиленный режим установкой бита ODEN регистра PWR_CR и ждём установки флага ODRDY в регистре PWR_CSR.
4) Переключаем регулятор из Нормального режима в Усиленный установкой бита ODSW в регистре PWR_CR. Во время переключения система будет стоять, а PLL тактируемая система будет работать.
5) Ждём установки флага ODSWRDY в регистре PWR_CSR.
6) Ставим нужную задержку Flash и предделители AHB и APB.
7) Ждём блокировки PLL.
8) Переключаем системные такты на PLL.
9) Включаем периферию, которая не сгенерирована Системной PLL (I2S clock, LCD-TFT clock, SAI1 clock, USB_48MHz clock....).

NB: PLLI2S и PLLSAI можно конфигурировать одновременно с системной PLL.

Во время переключения в Усиленный режим периферию тактировать нельзя, только после активации Усиленного режима.

Переход а режим Stop выключает Усиленный режим и PLL, так что при выходе из него их надо конфигурировать снова.

Выход из усиленного режима

Рекомендуем выходить из Усиленного режима при тактировании системы от HSI или HSE когда программа не выполняет критических задач. Есть две последовательности выхода:

- Одновременным снятием битов ODEN и ODSW в регистре PWR_CR (последовательность 1)
- Сначала снять бит ODSW для переключения регулятора напряжения в Нормальный режим и затем снятием бита ODEN выключить Усиленный режим (последовательность 2).

Пример последовательности 1:

1) Источником системных тактов выбираем HSI или HSE.
2) Выключаем такты периферии, которая не кормится от системной PLL (I2S clock, LCD-TFT clock, SAI1 clock, USB_48MHz clock,....)
3) Одновременно снимаем биты ODEN и ODSW в регистре PWR_CR.
4) Ждём снятия флага ODWRDY в PWR_CSR.

Пример последовательности 2:

1) Источником системных тактов выбираем HSI или HSE.
2) Выключаем такты периферии, которая не кормится от системной PLL (I2S clock, LCD-TFT clock, SAI1 clock, USB_48MHz clock,....).
3) Переключаем регулятор напряжения в Нормальный режим снятием бита ODSW в регистре PWR_CR. На время переключения напряжения системные такты останавливаются.
4) Ждём снятия флага ODWRDY в PWR_CSR.
5) Выключаем Усиленный режим снятием бита ODEN в регистре PWR_CR.

NB: Во время шага 3 бит ODEN ещё стоит, но Усиленный режим ещё разрешён, но неактивен (бит ODSW сброшен). Если сначала сбросить бит ODEN, то Усиленный режим отключается и регулятор напряжения возвращается в исходное состояние.

5.2. Супервизор питания

5.2.1. Сброс по включению (POR)/по выключению (PDR)

Встроенная схема POR/PDR обеспечивает правильную работу начиная с 1.8 V.

Устройство остаётся в режиме Сброса при V_{DD}/V_{DDA} ниже установленного порога, $V_{POR/PDR}$, без нужды во внешней схеме сброса. Подробности в описании устройств.

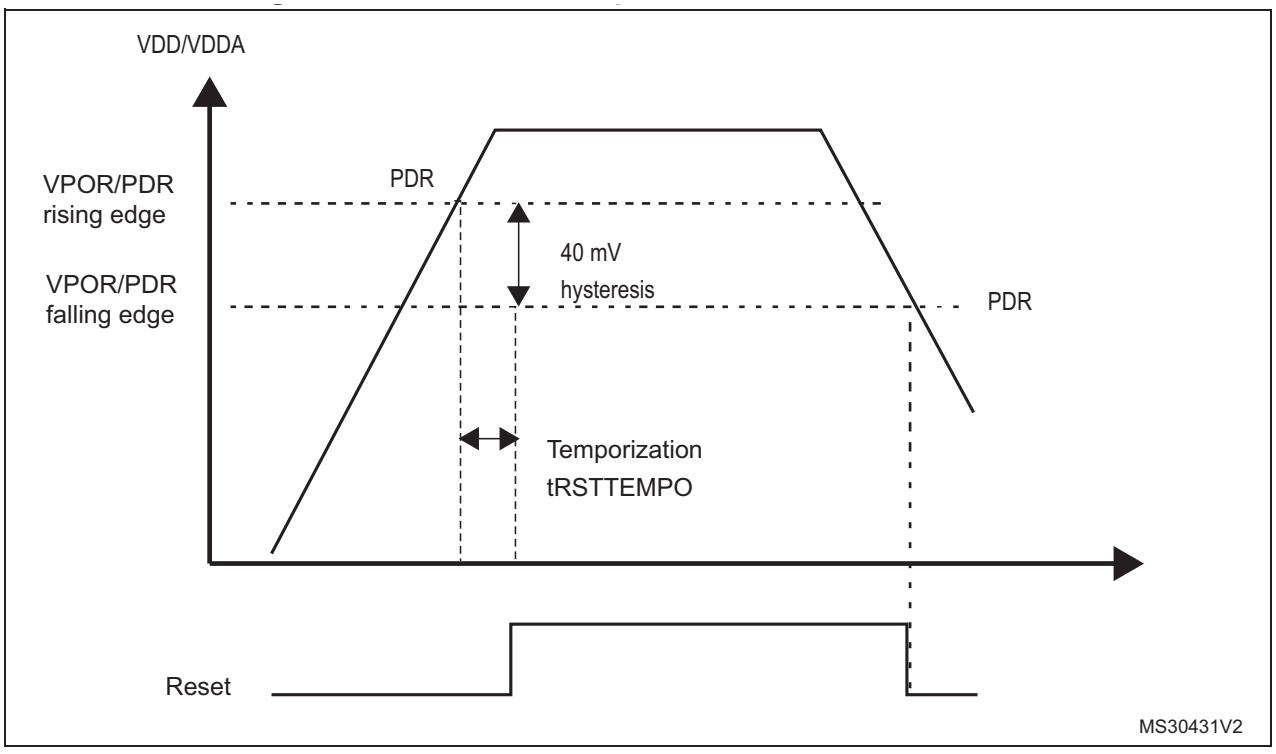


Рис. 12. Сброс по включению (POR)по выключению (PDR)

5.2.2. Сброс по падению напряжения (BOR)

Во время включения питания схема BOR удерживает сброс активным пока напряжение питания не достигнет порога V_{BOR}.

V_{BOR} пишется в байты опций устройства. По умолчанию BOR выключена. Можно выбрать три программируемых уровня V_{BOR}:

- BOR Level 3 (VBOR3).

- BOR Level 2 (VBOR2).
- BOR Level 1 (VBOR1).

Конкретные характеристики есть в описаниях устройств.

При падении напряжения питания (V_{DD}) ниже порога V_{BOR} выдаётся сброс устройства.

Выключается BOR в байтах опций и напряжение контролируется схемой POR/ PDR.

Гистерезис BOR равен ~100 mV (между передним и задним фронтом напряжения питания).

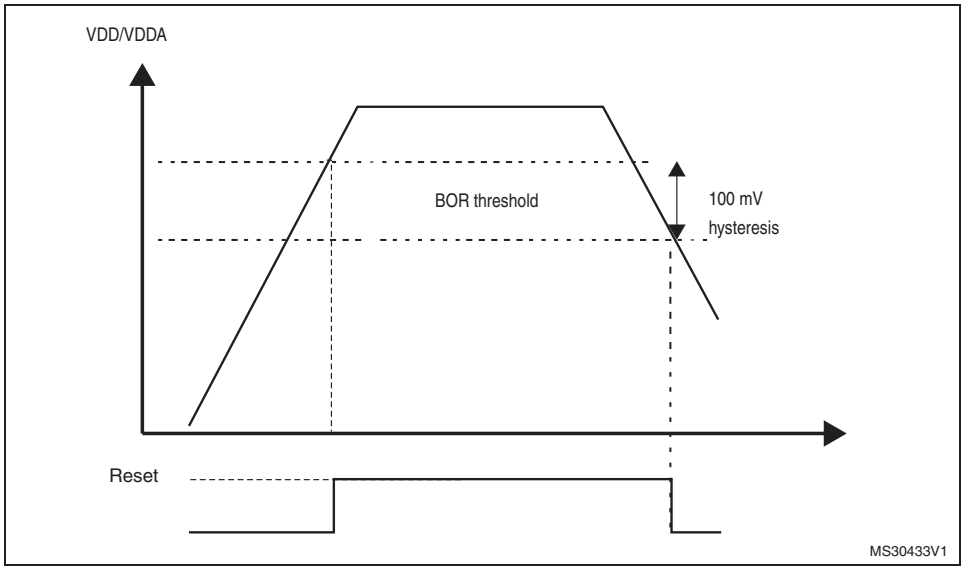


Рис. 13. Гистерезис BOR

5.2.3. Программируемый детектор напряжения (PVD)

PVD отслеживает уровень питания V_{DD}/V_{DDA} сравнивая его с порогом, указанным в битах `PLS[2:0]` регистра управления питанием (`PWR_CR`). Включается PVD установкой бита `PVDE`.

Флаг `PVDO` в регистре управления/состояния питания (`PWR_CSR`) показывает, что V_{DD}/V_{DDA} выше или ниже порога PVD. Это событие внутренне подключено к и может вызвать прерывание при его разрешении в регистрах `EXTI`. Направление перехода V_{DD}/V_{DDA} через порог PVD для генерации прерывания задаётся конфигурацией `EXTI[16]`.

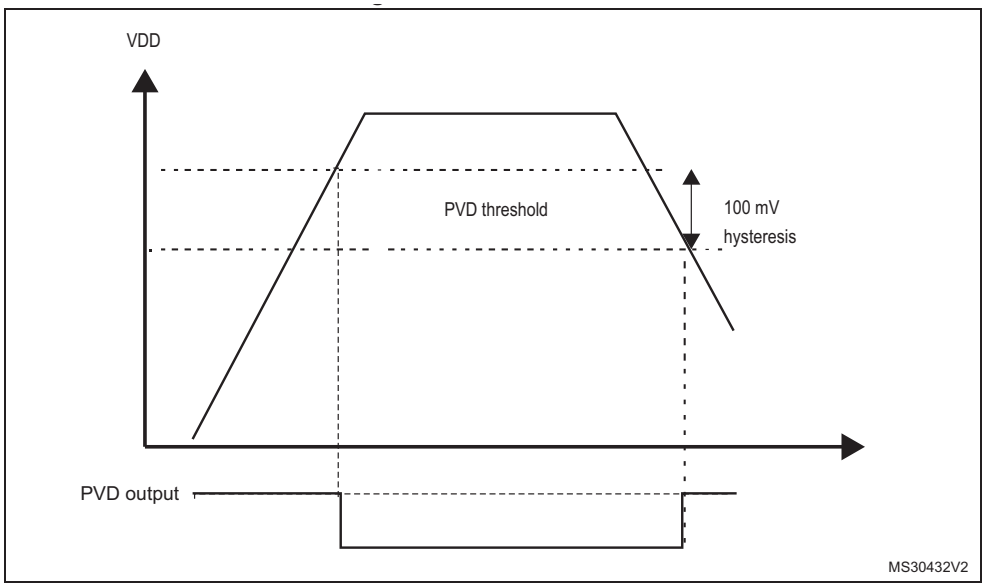


Рис. 14. Пороги PVD

5.3. Режимы пониженного потребления

По умолчанию после Сброса системы или питания микроконтроллер работает в режиме Run, CPU тактируется от HCLK и программа выполняется. Однако есть в наличии и режимы пониженного потребления, когда CPU совсем делать нечего. Это круто выбрать нужный компромисс между низким потреблением, быстрым запуском и ресурсами пробуждения.

В устройствах есть три экономичных режима:

- Сон (Sleep) — (тактов CPU нет, вся периферия, включая NVIC, SysTick и т.д., работает)
- Останов (Stop) (все тики остановлены)
- Дежурный (Standby) (1.2V домен выключен)

Кроме того, потребление можно понизить и в режиме Run:

- Замедляя системные тики
- Закрывая тики для ненужной периферии APBx и AHBx.

Вход в экономные режимы

Для этого MCU может может выполнять команды WFI (Wait For Interrupt), или WFE (Wait for Event), или если при выходе из прерывания в системном регистре управления стоит бит SLEEPONEXIT.

Если при исполнении команд WFI или WFE стоят запросы прерываний или событий, то экономный режим не включается.

Выход из экономного режима low-power mode

Выход MCU из режимов Sleep и Stop зависит от способа входа:

- Если был вход по команде WFI или возврат из прерывания, то пробуждает любое подтверждённое NVIC прерывание.
- Если был вход по команде WFE, то MCU пробуждается по событию. Оно может быть выдано:
  - Прерыванием NVIC IRQ:
    Если в системном регистре управления бит SEVONPEND = 0: разрешением прерывания в регистре управления периферией и NVIC. При выходе MCU из WFE надо чистить биты удержания прерывания в периферии, канале прерывания NVIC (в регистре очистки задержанных прерываний NVIC). Разбудят и прервут MCU только разрешённые прерывания NVIC с достаточным приоритетом.
    Если в системном регистре управления бит SEVONPEND = 1: разрешением прерывания в регистре управления периферией и необязательно NVIC. При выходе MCU из WFE надо чистить биты удержания прерывания в периферии, канале прерывания NVIC (в регистре очистки задержанных прерываний NVIC). All NVIC interrupts will wakeup the MCU разбудят все прерывания, даже неразрешённые. Разбудят и прервут MCU только разрешённые прерывания NVIC с достаточным приоритетом.
  - Событием
    Это делается конфигурацией линии EXTI в режим события. При выходе MCU из WFE чистить бит удержания прерывания периферии EXTI или канала NVIC IRQ не надо. Может понадобиться снять бит прерывания в периферии.

Из режима Standby MCU выходит по внешнему сбросу (ножка NRST), сброс IWDG, переднему фронту ножек WKUPx или событию RTC.

После выхода из режима Standby выполняется вся процедура , положенная после Сброса.

Разбудят и прервут MCU только разрешённые прерывания NVIC с достаточным приоритетом.

Таблица 23. Обзор режимов низкого потребления

Режим	Вход	Побудка	Действие на такты 1.2 V домена	Действие на такты V_{DD} домена	Регулятор напряжения
Sleep (Sleep now или Sleep-on exit)	WFI или возврат из ISR	Любое прерывание	Только выкл. тактов CPU	Нету	ВКЛ
	WFE	Событие побудки			
Stop	Биты PDDS и LPDS bits + бит SLEEPDEEP + WFI, возврат из ISR или WFE	Любая линия EXTI (заданная в регистрах EXTI, внутренние и внешние линии)			ВКЛ или в экономном режиме (зависит от регистра PWR_CR

Standby	Бит PDDS bit + бит SLEEPDEEP + WFI, возврат из ISR или WFE	Передний фронт ножки WKUP, RTC (Alarm A или Alarm B), события RTC: Wakeup, tamper, time stamp, или внешний сброс на ножке NRST, сброс IWDG	Все такты 1.2 V домена выключены	Генераторы HSI и HSE выключены	ВЫКЛ

5.3.1. Замедление системных тактов

В режиме Run частоту системных тиков (SYSCLK, HCLK, PCLK1, PCLK2) можно снизить регистрами предделителей. Ими также можно замедлить периферию перед входом в режим Sleep mode. Точнее см. в *Секции 7.3.3*.

5.3.2. Выключение тактов периферии

В режиме Run такты HCLKx и PCLKx для периферии и памяти можно выключать, а для режима Sleep их можно выключить перед командами `WFI` или `WFE`.

Тактирование периферии AHB1 управляется регистром (`RCC_AHB1ENR`), AHB2 - регистром (`RCC_AHB2ENR`) и AHB3 - регистром (`RCC_AHB3ENR`). См. *Секции 7.3.10, 7.3.11, 7.3.12, 6.3.10, 6.3.11, 6.3.12*.

Такты периферии в режиме Sleep можно выключить автоматически сбросом соответствующих битов в регистрах `RCC_AHBxLPENR` и `RCC_APBxLPENR`.

5.3.3. Режим Sleep

Вход

Он описан выше в Разделе 5.3. при чистом бите `SLEEPDEEP` в системном регистре управления. См. Таблицу 24 и Таблицу 25, там детали. Все биты удержания прерываний надо чистить до входа в режим сна.

Выход

Он описан выше в Разделе 5.3. при чистом бите `SLEEPDEEP` в системном регистре управления. См. Таблицу 24 и Таблицу 25, там детали.

Таблица 24. Sleep-now

Sleep-now	Описание
Вход	WFI (Wait for Interrupt) или WFE (Wait for Event) когда: – SLEEPDEEP = 0, и – Нет удерживаемого прерывания (для WFI) или события (для WFE).
	При возврате из ISR когда: – SLEEPDEEP = 0 и – SLEEPONEXIT = 1, – нет удерживаемого прерывания.
Выход	При входе по WFI или возврате из ISR: Прерывание: См. Таблицу 61. и Таблицу 62. При входе по WFE и SEVONPEND = 0 Событие побудки: См. Секцию 12.2.3. При входе по WFE и SEVONPEND = 1 Даже неразрешённое в NVIC прерывание. (См. Таблицу 61. и Таблицу 62.) или Событие побудки: См. Секцию 12.2.3..
Задержка пробуждения	НЕТУ

Таблица 25. Sleep-on-exit

Sleep-on-exit	Описание
	WFI (Wait for Interrupt) или WFE (Wait for Event) когда: – SLEEPDEEP = 0, и – Нет удерживаемого прерывания (для WFI) или события (для WFE).

Sleep-on-exit	Описание
Вход	При возврате из ISR когда: – SLEEPDEEP = 0 и – SLEEPONEXIT = 1, – нет удерживаемого прерывания.
Выход	Прерывание: См. Таблицу 61. и Таблицу 62.
Задержка пробуждения	НЕТУ

5.3.4. Режим Stop для (STM32F405xx/07xx и STM32F415xx/17xx)

Режим Stop основан на сочетании режима *deepsleep* с отключением тактирования периферии. Регулятор напряжения можно переключать в нормальный или экономный режим. В режиме Stop тактирование 1.2V останавливается, генераторы PLL, HSI и HSE выключаются. Внутренняя SRAM и регистры сохраняются.

При установленном бите FPDS в регистре PWR_CR питание Flash памяти отключается при переходе устройства в режим Stop. При этом появляется дополнительная задержка выхода из режима (см. *Таблицу 26. и Секцию 5.4.1)*.

Таблица 26. Режимы Stop (STM32F405xx/07xx и STM32F415xx/17xx)

Режим Stop	Бит LPDS	Бит FPDS	Задержка побудки
STOP MR (Main regulator)	0	0	Время старта HSI RC
STOP MR-FPD	0	1	Время старта HSI RC + время включения Flash
STOP LP	1	0	Время старта HSI RC + время перехода из LP режима
STOP LP-FPD	1	1	Время старта HSI RC + время включения Flash + время перехода питания из LP режима

Вход в режим Stop для (STM32F405xx/07xx и STM32F415xx/17xx)

Он описан выше в Разделе 5.3. при стоящем бите SLEEPDEEP в системном регистре управления.

См. *Таблицу 27*. Ради экономии питания можно битом LPDS в регистре PWR_CR переключить в бережливый режим и регулятор напряжения.

Во время программирования флэш памяти переход в режим Stop откладывается до завершения доступа к памяти.

Во время работы домена APB переход в режим Stop откладывается до завершения доступа APB.

В режиме Stop отдельными битами можно включить:

- Независимый сторожевой таймер (IWDG): IWDG запускается записью в его регистр Ключа или аппаратно. Запущенный однажды он останавливается только Сбросом. См. *Секцию 21.3*.
- Real-time clock (RTC): битом RTCEN в регистре RCC_BDCR
- Внутренний RC генератор (LSI RC): битом LSION в регистре RCC_CSR.
- Внешний 32.768 kHz генератор (LSE OSC): битом LSEON в регистре RCC_BDCR.

В режиме Stop блоки ADC и DAC могут кушать электричество, даже если отключить их. Во избежание этого нужно записать 0 в бит ADON регистра ADC_CR2 и в ENx регистра DAC_CR.

NB. Если перед остановкой нужно выключить внешний тактовый генератор, то его надо отключить битом HSEON и переключиться на внутренний HSI. Иначе, если бит HSEON остаётся включённым и внешние такты при остановке устраняются, то должна быть включена система безопасности тактирования (CSS) для обнаружения сбоев внешнего тактирования и предупреждения недостойного поведения системы.

Выход из режима Stop для (STM32F405xx/07xx и STM32F415xx/17xx)

Как выходить написано в Секции 5.3. См. Таблицу 27.

При выходе из режима Stop по прерыванию или событию в качестве системных тактов выбирается RC генератор HSI.

При выходе из режима Stop при регуляторе напряжения в экономном режиме появляется дополнительная задержка, но включённый регулятор напряжения в режиме Stop увеличивает потребление тока.

Таблица 27. Вход/выход в режим Stop для (STM32F405xx/07xx и STM32F415xx/17xx).

Режим Stop	Описание
Вход	По WFI (Wait for Interrupt) или WFE (Wait for Event) когда: – Нет удерживаемого прерывания (для WFI) или события (для WFE), – SLEEPDEEP =1, – PDDS=0 (PWR_CR), – Выбран регулятор напряжения битом LPDS в PWR_CR.
	По возврату из ISR: – Нет удерживаемого прерывания, – SLEEPDEEP =1, – SLEEPONEXIT = 1, – PDDS=0 (PWR_CR).
	NB: Для входа должны быть сняты биты удержания: всех EXTI Line, всех периферийных прерываний, RTC Alarm (Alarm A и Alarm B), побудка RTC, взлом от RTC и флаги меток времени RTCt. Иначе, игнор для Stop.
Выход	При входе по WFI или Возврате из ISR: —Любая линия EXTI в режиме прерываний (разрешённая в NVIC) от внешнего источника или внутреннего с возможностью побудки. См. Таблицу 61. и Таблицу 62. При входе по WFE с SEVONPEND = 0 —Любая линия EXTI в режиме события. См. Секцию 12.2.3. При входе по WFE с SEVONPEND = 1: —Любая линия EXTI в режиме прерываний (даже неразрешённая в NVIC) от внешнего источника или внутреннего с возможностью побудки. См. Таблицу 61. и Таблицу 62. —Событие побудки. См. Секцию 12.2.3.
Задержка	См. Таблицу 26.

5.3.5. Режим Stop для (STM32F42xxx и STM32F43xxx)

Режим Stop основан на сочетании режима *deepsleep* с отключением тактирования периферии. Регулятор напряжения можно переключать в нормальный или экономный режим. В режиме Stop тактирование 1.2V останавливается, генераторы PLL, HSI и HSE выключаются. Внутренняя SRAM и регистры сохраняются.

Потребление в режиме Stop можно уменьшить установками в регистре PWR_CR. При этом появляется дополнительная задержка выхода из режима (см. *Таблицу 28.*).

Таблица 28. Режимы для (STM32F42xxx и STM32F43xxx)

Режим регулятора		UDEN [1:0]	MRUDS	LPUDS	LPDS	FPDS	Задержка побудки
Норм.	STOP MR	-	0	-	0	0	Пуск HSI RC
	STOP MR- FPD	-	0	-	0	1	Пуск HSI RC + вкл. Flash
	STOP LP	-	0	0	1	0	Пуск HSI RC + выход питания из LP
	STOP LP-FPD	-	-	0	1	1	Пуск HSI RC + вкл. Flash + выход из LP
Слаб.	STOP UMR- FPD	3	1	-	0	-	Пуск HSI RC + вкл. Flash + выход основн. питания из LP + запуск логики ядра
	STOP ULP-FPD	3	-	1	1	-	Пуск HSI RC + вкл. Flash + выход питания из LP + запуск логики ядра

Вход в режим Stop для (STM32F42xxx и STM32F43xxx)

Он описан выше в Разделе 5.3. при стоящем бите SLEEPDEEP в системном регистре управления.

См. *Таблицу 29*. При этом автоматически выбирается шкала питания 3, Ради экономии внутренний источник питания можно переключить в бережливый режим битами LPDS, MRUDS, LPUDS, и UDEN в регистре PWR_CR.

Во время программирования флэш памяти переход в режим Stop откладывается до завершения доступа к памяти.

Во время работы домена APB переход в режим Stop откладывается до завершения доступа APB.

При переходе в режим Stop автоматически отключается усиленный режим питания.

В режиме Stop отдельными битами можно включить:

- Независимый сторожевой таймер (IWDG): IWDG запускается записью в его регистр Ключа или аппаратно. Запущенный однажды он останавливается только Сбросом. См. *Секцию 21.3*.
- Real-time clock (RTC): битом RTCEN в регистре RCC_BDCR
- Внутренний RC генератор (LSI RC): битом LSION в регистре RCC_CSR.
- Внешний 32.768 kHz генератор (LSE OSC): битом LSEON в регистре RCC_BDCR.

В режиме Stop блоки ADC и DAC могут кушать электричество, даже если отключить их. Во избежание этого нужно записать 0 в бит ADON регистра ADC_CR2 и в ENx регистра DAC_CR.

NB. Перед остановкой рекомендуется включить систему безопасности тактирования (CSS) и переключиться с внешнего тактового генератора на внутренний HSI.

Выход из режима Stop для (STM32F42xxx и STM32F43xxx)

Как выходить написано в Секции 5.3. См. Таблицу 29.

При выходе из режима Stop по прерыванию или событию в качестве системных тактов выбирается RC генератор HSI. После выхода ослабленный режим автоматически отключается.

При выходе из режима Stop при регуляторе напряжения в экономном режиме появляется дополнительная задержка, но включённый регулятор напряжения в режиме Stop увеличивает потребление тока.

Таблица 29. Вход/выход в режим Stop (STM32F42xxx и STM32F43xxx).

Stop mode	Description
Вход	По WFI (Wait for Interrupt) WFE (Wait for Event) когда: – Нет задержанных прерываний и событий, – SLEEPDEEP = 0, – Выбор режима регулятора битами LPDS, MRUDS, LPUDS и UDEN в PWR_CR (см. Таблицу 28.).
	По возврату из ISR когда: —Нет задержанных прерываний, —SLEEPDEEP=1, и —SLEEPONEXIT = 1, и —PDDS=0 в PWR_CR1.
	NB: Для входа должны быть сняты биты удержания: всех EXTI Line, всех периферийных прерываний, RTC Alarm (Alarm A и Alarm B), побудка RTC, взлом от RTC и флаги меток времени RTCt. Иначе, игнор для Stop.
Выход	При входе по WFI или Возврате из ISR: —Любая линия EXTI в режиме прерываний (разрешённая в NVIC) от внешнего источника или внутреннего с возможностью побудки. См. Таблицу 61. и Таблицу 62. При входе по WFE с SEVONPEND = 0 —Любая линия EXTI в режиме события. См. Секцию 12.2.3. При входе по WFE с SEVONPEND = 1: —Любая линия EXTI в режиме прерываний (даже неразрешённая в NVIC) от внешнего источника или внутреннего с возможностью побудки. См. Таблицу 61. и Таблицу 62. —Событие побудки. См. Секцию 12.2.3.
Задержка	См. Таблицу 28.

5.3.6. Дежурный режим Standby

Самый экономный режим Standby это сочетание режима *deepsleep* с выключением регулятора напряжения. Выключаются: домен 1.2V, PLL, генераторы HSI и HSE. SRAM и регистры теряются, кроме домена резервного хранения и схематики Standby.

Вход

Как входить написано в Секции 5.3. с битом SLEEPDEEP=1. См. Таблицу 30.

В режиме Standby отдельными битами можно включить:

- Независимый сторожевой таймер (IWDG): IWDG запускается записью в его регистр Ключа или аппаратно. Запущенный однажды он останавливается только Сбросом. См. *Секцию 19.3*.
- Real-time clock (RTC): битом RTCEN в регистре RCC_BDCR

- Внутренний RC генератор (LSI RC): битом LSION в регистре RCC_CSR.
- Внешний 32.768 kHz генератор (LSE OSC): битом LSEON в регистре RCC_BDCR.

Выход

Выход из Standby аналогичен выходу из экономного режима. Флаг SBF в регистре PWR_CSR указывает на выход из Standby. Все регистры, кроме PWR_CSR, сбрасываются.

Таблица 30. Вход и выход из Standby.

Режим Standby	Описание
Вход	WFI (Wait for Interrupt) или WFE (Wait for Event) когда: —SLEEPDEEP=1, —PDDS=1, —Нет задержанных прерываний (для WFI) или событий (для WFE), —WUF=0 в регистре (PWR_CR), —Флаг RTC события источника побудки (RTC Alarm A, RTC Alarm B, RTC wakeup, Tamper or Timestamp flags) чист
	По возврату из ISR когда: —SLEEPDEEP=1, —SLEEPONEXIT = 1, —PDDS=1, —Нет задержанных прерываний, —WUF=0 в регистре (PWR_CR), —Флаг RTC события источника побудки (RTC Alarm A, RTC Alarm B, RTC wakeup, Tamper or Timestamp flags) чист
Выход	Передний фронт ножки WKUP, будильник RTC (Alarm A и Alarm B), побудка RTC, взлом, метка времени, внешний сброс на ножке NRST, сброс IWDG.
Wakeup latency	Фаза Сброса.

Состояние ножек I/O в режиме Standby

Все ножки находятся в третьем состоянии, кроме:

- Reset pad (доступно)
- Ножка RTC_AF1 (PC13) если сконфигурирована как взлом, метка времени, будильник RTC или выход калибровки RTC
- Ножка WKUP, если разрешена

Режим отладки

По умолчанию из-за остановки ядра подключение отладки в режимах Stop и Standby теряется.

Но в установкой битов конфигурации в регистре DBGMCU_CR отлаживать можно и в экономных режимах. Детали в *Секции 38.16.1*.

5.3.7. Альтернативные функции RTC для выхода из Stop и Standby

Это будильники RTC (Alarm A и Alarm B), побудка RTC, событие взлома от RTC и метка времени от RTC. Будильником или побудкой RTC систему можно выводить из экономных режимов без внешнего прерывания. Для этого у RTC есть своя база времени.

Битами RTCSEL[1:0] в регистре RCC_BDCR можно выбрать два из трёх источников RTC:

- Внешний 32.768 kHz кварцевый генератор (LSE OSC). Это самый точный и экономный режим (менее 1μA добавки)
- Внутренний RC генератор (LSI RC). Экономия на стоимости кварцевого резонатора.

Альтернативные функции RTC для выхода из Stop

- Для выхода из Stop по событию RTC alarm надо:
  a) Поставить EXTI Line 17 чувствительность к переднему фронту (Прерывание или Событие)
  b) Разрешить прерывание RTC Alarm в регистре RTC_CR
  c) Включить на RTC выдачу RTC alarm
- Для выхода из Stop по событию RTC tamper или time stamp надо:
  a) Поставить EXTI Line 21 чувствительность к переднему фронту (Прерывание или Событие)
  b) Разрешить в регистре RTC_CR прерывание RTC time stamp или RTC tamper interrupt в регистре RTC_TAFCR

c) Включить на RTC обнаружение событий взлома или меток времени

- Для выхода из Stop по событию побудки RTC надо:
  a) Поставить EXTI Line 22 чувствительность к переднему фронту (Прерывание или Событие)
  b) Разрешить прерывание RTC wakeup в регистре RTC_CR
  c) Включить на RTC выдачу события RTC Wakeup

Альтернативные функции RTC для выхода из Standby

- Для выхода из Standby по событию RTC alarm надо:
  a) Разрешить прерывание RTC Alarm в регистре RTC_CR
  b) Включить на RTC выдачу RTC alarm
- Для выхода из Standby по событию RTC tamper или time stamp надо:
  a) Разрешить в регистре RTC_CR прерывание RTC time stamp или RTC tamper interrupt в регистре RTC_TAFCR
  b) Включить на RTC обнаружение событий взлома или меток времени
- Для выхода из Standby по событию побудки RTC надо:
  a) Разрешить прерывание RTC wakeup в регистре RTC_CR
  b) Включить на RTC выдачу события RTC Wakeup

Последовательность безопасного снятия флага побудки RTC

Если альтернативную функцию RTC поставить до снятия флага побудки (WUTF), то она не сработает по следующему переднему фронту.

Во избежание дребезга на ножках альтернативных функций RTC и правильного выхода из режимов Stop и Standby рекомендуется перед входом в режим Standby:

- При использовании для побудки сигнала RTC alarm:
  a) Запретить прерывание RTC alarm (биты ALRAIE или ALRBIE в RTC_CR)
  b) Снять флаг RTC alarm (ALRAF/ALRBF)
  c) Снять флаг PWR Wakeup (WUF)
  d) Разрешить прерывание RTC alarm
  e) Повторно войти в экономный режим
- При использовании для побудки сигнала RTC:
  a) Запретить прерывание RTC Wakeup (бит WUTIE в RTC_CR)
  b) Снять флаг Wakeup (WUTF)
  c) Снять флаг PWR Wakeup (WUF)
  d) Разрешить прерывание RTC Wakeup
  e) Повторно войти в экономный режим
- При использовании для побудки сигнала RTC tamper:
  a) Запретить прерывание RTC tamper (бит TAMPIE в RTC_TAFCR)
  b) Снять флаг Tamper (TAMP1F/TSF)
  c) Снять флаг PWR Wakeup (WUF)
  d) Разрешить прерывание RTC tamper
  e) Повторно войти в экономный режим
- When using RTC time stamp to wake up the device from the low-power modes:
  a) Запретить прерывание RTC time stamp (бит TSIE в RTC_CR)
  b) Снять флаг RTC time stamp (TSF)
  c) Снять флаг PWR Wakeup (WUF)
  d) Разрешить прерывание RTC TimeStamp
  e) Повторно войти в экономный режим.

5.4. Регистры управления питанием для STM32F405xx/07xx и STM32F415xx/17xx

5.4.1. Регистр управления (PWR_CR) для STM32F405xx/07xx и STM32F415xx/17xx

Смещение адреса: 0x00

По сбросу: 0x0000 4000 (сброс выходом из Standby)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	VOS	Reserved				FPDS	DBP	PLS[2:0]			PVDE	CSBF	CWUF	PDDS	LPDS
	rw					rw	rw	rw	rw	rw	rw	w	w	rw	rw

— Биты 31:15 Резерв, не трогать.

— Бит 14 **VOS**: Выбор шкалы выхода регулятора напряжения

0: Шкала 2

1: Шкала 1 (по умолчанию)

— Биты 13:10 Резерв, не трогать.

— Бит 9 **FPDS**: Выключение питания Flash при входе в режим Stop

0: Не выключать

1: Выключить

— Бит 8 **DBP**: Отключить защиту записи резервного домена.

В состоянии сброса регистр RCC_BDCR, регистры RTC, резервные регистры и бит BRE регистра PWR_CSR защищены от паразитной записи, а потом её нужно разрешать.

0: Доступа нет

1: Запись разрешена.

NB: Если такты RTC это HSE/128, то этот бит должен оставаться в 1.

— Биты 7:5 **PLS[2:0]**: Выбор уровня детектора питания.

000: 2.0V

001: 2.1V

010: 2.3V

011: 2.5V

100: 2.6V

101: 2.7V

110: 2.8V

111: 2.9V

NB: См. описание устройства.

— Бит 4 **PVDE**: Включение детектора питания, изменяется программно.

0: PVD выключен

1: PVD включён

— Бит 3 **CSBF**: Очистка флага Standby, читается как 0.

0: Не влияет

1: Чистит SBF Standby Flag.

— Бит 2 **CWUF:** Очистка флага побудки, читается как 0.

0: Не влияет

1: Чистит WUF Wakeup Flag **после 2 System clock циклов**.

— Бит 1 **PDDS**: Режим *deepsleep*.

Изменяется программно, работает вместе с битом LPDS.

0: Включает режим Stop при входе CPU в Deepsleep. Регулятор зависит от бита LPDS.

1: Включает режим Standby при входе CPU в Deepsleep.

— Бит 0 **LPDS**: Экономный *deepsleep*.

Изменяется программно, работает вместе с битом PDDS.

0: В режиме Stop регулятор напряжения включён

1: В режиме Stop регулятор напряжения экономит электричество.

5.4.2. Регистр состояния и управления (PWR_CSR) для STM32F405xx/07xx и STM32F415xx/17xx

Смещение адреса: 0x04

По сбросу: 0x0000 0000 (выходом из Standby не сбрасывается)

Для чтения нужны дополнительные циклы APB.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	VOS RDY	Reserved				BRE	EWUP	Reserved				BRR	PVDO	SBF	WUF
	r					rw	rw					r	r	r	r

— Биты 31:9 Резерв, не трогать.

— Бит 14 **VOSRDY**: Регулятор напряжения готов

0: Не готов

1: Готов

— Биты 13:10 Резерв, не трогать.

— Бит 9 **BRE**: Включение регулятора питания резервного домена

0: Выключен, содержимое SRAM в режимах Standby и V_{BAT} теряется

1: Включение, для перехода надо дождаться флага BRR

NB: При выходе из режима Standby, системным сбросом и сбросом питания не снимается.

— Бит 8 **EWUP**: Разрешение ножки WKUP.

Изменяется программно.

0: Ножка WKUP используется как I/O и из Standby не будит

1: Ножка WKUP выводит из Standby по переднему фронту сигнала.

NB: Очищается системным Сбросом.

— Биты 7:4 Резерв, не трогать.

— Бит 3 **BRR**: Готовность регулятора питания резервного домена

0: Не готов

1: Готов

NB: При выходе из режима Standby, системным сбросом и сбросом питания не снимается.

— Бит 2 **PVDO:** Выход детектора питания.

Изменяется аппаратно. Достоверен только при включённом PVD (бит PVDE)

0: V_{DD}/V_{DDA} больше порога PVD в битах PLS[2:0]

1: V_{DD}/V_{DDA} меньше порога PVD в битах PLS[2:0].

— Бит 1 **SBF**: Флаг Standby.

Ставится аппаратно, снимается только POR/PDR или установкой бита CSBF в регистре PWR_CR.

0: Standby не было.

1: Standby был.

— Бит 0 **WUF**: Флаг побудки.

Изменяется аппаратно, системным сбросом или битом CWUF в регистре PWR_CR.

0: Побудки не было

1: Разбудили ножкой WKUP или сигналом RTC alarm.

NB: Если ножку WKUP разрешают битом EWUP, а уровень WKUP уже высокий, то появляется дополнительное событие побудки.

5.5. Регистры управления питанием для STM32F42xxx и STM32F43xxx

5.5.1. Регистр управления (PWR_CR) для STM32F42xxx и STM32F43xxx

Смещение адреса: 0x00

По сбросу: 0x0000 0000 (сброс выходом из Standby)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												UDEN[1:0]		ODSWEN	ODEN
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VOS[1:0]		ADCDC1	Res.	MRUDS	LPUDS	FPDS	DBP	PLS[2:0]			PVDE	CSBF	CWUF	PDDS	LPDS
rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rc_w1	rc_w1	rw	rw

— Биты 31:20 Резерв, не трогать.

— Биты 19:18 **UDEN[1:0]**: Разрешение ослабленного питания в режиме Stop

Ставятся программно.

00: Нельзя

01: Резерв

10: Резерв

11: Можно

— Бит 17 **ODSWEN**: Разрешение ключа подачи усиленного питания.

Ставятся программно. Снимается аппаратно при выходе из режима Stop или при снятии бита ODEN. Системные такты должны быть от HSI или HSE. Ставить можно при стоящем бите ODRDY.

0: Нельзя

1: Можно

NB: При изменении состояния ключа системные такты должны быть остановлены до установки напряжения.

— Бит 16 **ODEN**: Разрешение усиленного питания.

Ставятся программно. Снимается аппаратно при выходе из режима Stop. Системные такты должны быть от HSI или HSE. При стоящем ODEN перед включением ODSWEN надо дождаться флага ODRDY.

0: Выключено

1: Включено

— Биты 15:14 **VOS[1:0]**: Выбор шкалы выхода регулятора напряжения

Менять можно только при выключенном PLL (выбирается шкала 3). Новое значение действует при включении PLL.

00: Резерв (выбрана шкала 3)

01: Шкала 2

10: Шкала 2

11: Шкала 1 (по сбросу)

— Бит 13 **ADCDC1**:

0: Ничего.

1: См. документ AN4073, там сказано.

NB: Ставить можно при напряжении от 2.7 до 3.6V и выключенной предвыборкой.

— Бит 12 Резерв, не трогать.

— Бит 11 **MRUDS**: Главный регулятор в ослабленном режиме Stop

Ставится и снимается программно.

0: Главный регулятор ВКЛ в режиме Stop

1: Главный регулятор в ослабленном режиме и Flash память выключена в режиме Stop.

— Бит 10 **LPRUDS**: Слабый регулятор в режиме deepsleep

Ставится и снимается программно.

0: В режиме Stop при стоящем бите LPDS слабый регулятор включён

1: В режиме Stop при стоящем бите LPDS слабый регулятор включён и Flash память выключена.

— Бит 9 **FPDS**: Выключение питания Flash при входе в режим Stop

0: Не выключать

1: Выключить

— Бит 8 **DBP**: Отключить защиту записи резервного домена.

В состоянии сброса регистр RCC_BDCR, регистры RTC, резервные регистры и бит BRE регистра PWR_CSR защищены от паразитной записи, а потом её нужно разрешать.

0: Доступа нет

1: Запись разрешена.

NB: Если такты RTC это HSE/128, то этот бит должен оставаться в 1.

— Биты 7:5 **PLS[2:0]**: Выбор уровня детектора питания.

000: 2.0V

001: 2.1V

010: 2.3V

011: 2.5V

100: 2.6V

101: 2.7V

110: 2.8V

111: 2.9V

NB: См. описание устройства.

— Бит 4 **PVDE**: Включение детектора питания, изменяется программно.
0: PVD выключен
1: PVD включён
— Бит 3 **CSBF**: Очистка флага Standby, читается как 0.
0: Не влияет
1: Чистит SBF Standby Flag.
— Бит 2 **CWUF:** Очистка флага побудки, читается как 0.
0: Не влияет
1: Чистит WUF Wakeup Flag **после 2 System clock циклов**.
— Бит 1 **PDDS**: Режим *deepsleep*.
Изменяется программно, работает вместе с битом LPDS.
0: Включает режим Stop при входе CPU в Deepsleep. Регулятор зависит от бита LPDS.
1: Включает режим Standby при входе CPU в Deepsleep.
— Бит 0 **LPDS**: Экономный *deepsleep*.
Изменяется программно, работает вместе с битом PDDS.
0: В режиме Stop регулятор напряжения включён
1: В режиме Stop регулятор напряжения экономит электричество.

5.5.2. Регистр состояния и управления (PWR_CSR) для STM32F42xxx и STM32F43xxx

Смещение адреса: 0x04
По сбросу: 0x0000 0000 (выходом из Standby не сбрасывается)
Для чтения нужны дополнительные циклы APB.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												UDRDY[1:0]		ODSWRDY	ODRDY
												rc_w1	rc_w1	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	VOS RDY	Reserved				BRE	EWUP	Reserved.				BRR	PVDO	SBF	WUF
	r					rw	rw					r	r	r	r

— Биты 31:20 Резерв, не трогать.
— Биты 19:18 **UDRDY[1:0]**: Ослабленный режим готов.
Ставится аппаратно при изменении питания MCU. Снимается записью 1.
00: Выключен
01: Резерв
10: Резерв
11: В режиме Stop ослабленный режим работает.
— Бит 17 **ODSWRDY**: Ключ усиленного питания готов
0: Ключ не активен.
1: В домене 1.2 V питание усилено.
— Бит 16 **ODRDY**: Усиленный режим готов.
0: Не готово.
1: Готово.
— Бит 15 Резерв, не трогать.
— Бит 14 **VOSRDY**: Регулятор напряжения готов
0: Не готов
1: Готов
— Биты 13:10 Резерв, не трогать.
— Бит 9 **BRE**: Включение регулятора питания резервного домена
0: Выключен, содержимое SRAM в режимах Standby и V_{BAT} теряется
1: Включение, для перехода надо дождаться флага BRR
NB: При выходе из режима Standby, системным сбросом и сбросом питания не снимается.
— Бит 8 **EWUP**: Разрешение ножки WKUP.
Изменяется программно.
0: Ножка WKUP используется как I/O и из Standby не будит

1: Ножка WKUP выводит из Standby по переднему фронту сигнала.

NB: Очищается системным Сбросом.

— Биты 7:4 Резерв, не трогать.

— Бит 3 **BRR**: Готовность регулятора питания резервного домена

0: Не готов

1: Готов

NB: При выходе из режима Standby, системным сбросом и сбросом питания не снимается.

— Бит 2 **PVDO:** Выход детектора питания.

Изменяется аппаратно. Достоверен только при включённом PVD (бит PVDE)

0: V_{DD}/V_{DDA} больше порога PVD в битах PLS[2:0]

1: V_{DD}/V_{DDA} меньше порога PVD в битах PLS[2:0].

— Бит 1 **SBF**: Флаг Standby.

Ставится аппаратно, снимается только POR/PDR или установкой бита CSBF в регистре PWR_CR.

0: Standby не было.

1: Standby был.

— Бит 0 **WUF**: Флаг побудки.

Изменяется аппаратно, системным сбросом или битом CWUF в регистре PWR_CR.

0: Побудки не было

1: Разбудили ножкой WKUP или сигналом RTC alarm.

NB: Если ножку WKUP разрешают битом EWUP, а уровень WKUP уже высокий, то появляется дополнительное событие побудки.

5.5.3. Карта регистров PWR

Таблица 31. Для STM32F405xx/07xx STM32F415xx/17xx

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**PWR_CR**	Reserved																	VOS	Reserved				FPDS	DBP	PLS[2:0]			PVDE	CSBF	CWUF	PDDS	LPDS
	Reset value																		1					0	0	0	0	0	0	0	0	0	0
0x004	**PWR_CSR**	Reserved																	VOSRDY	Reserved				BRE	EWUP	Reserved				BRR	PVDO	SBF	WUF
	Reset value																		0					0	0					0	0	0	0

Таблица 32. Для STM32F42xxx и STM32F43xxx

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	PWR_CR	Reserved												UDEN[1:0]		ODSWEN	ODEN	VOS[1:0]		ADCDC1	Reserved	MRUDS	LPUDS	FPDS	DBP	PLS[2:0]			PVDE	CSBF	CWUF	PDDS	LPDS
	Reset value													1	1	1	1	1	1	0		1	1	0	0	0	0	0	0	0	0	0	0
0x004	PWR_CSR	Reserved												UDRDY[1:0]		ODSWRDY	ODRDY	Reserved	VOSRDY	Reserved				BRE	EWUP	Reserved				BRR	PVDO	SBF	WUF
	Reset value													0	0	0	0		0					0	0					0	0	0	0

6. Сброс и такты в STM32F42xxx и STM32F43xxx (RCC)

6.1. Сброс

Есть три типа сброса: системный сброс, сброс питания и сброс домена резервного хранения.

6.1.1. Системный сброс

Сбрасывает все регистры в предписанное состояние кроме флагов сброса в регистре CSR и регистров домена резервного хранения.

Системный сброс генерируется в следующих случаях:

1. Низкий уровень на ножке NRST (внешний сброс)
2. Конец счёта Оконного сторожевого таймера (сброс WWDG)
3. Конец счёта Независимого сторожевого таймера (сброс IWDG reset)
4. Программный сброс (SW сброс)
5. Сброс Управления питанием

Источники идентифицируются флагами в регистре RCC_CSR (см. *Секцию 7.3.10*).

Программный сброс

Для запуска программного сброса надо поставить бит SYSRESETREQ в регистре управления системой.

Сброс управления питанием

Есть два вида сброса:

1. Сброс по входу в режим Standby:

   Он разрешается очисткой бита nRST_STDBY в байтах Опций Пользователя. В этом случае при успешном выполнении последовательности входа в режим Standby устройство сбрасывается вместо входа в режим Standby.

2. Сброс по входу в режим Stop:

   Он разрешается очисткой бита nRST_STOP в байтах Опций Пользователя. В этом случае при успешном выполнении последовательности входа в режим Stop устройство сбрасывается вместо входа в режим Stop.

6.1.2. Сброс питания

Генерируется в следующих случаях:

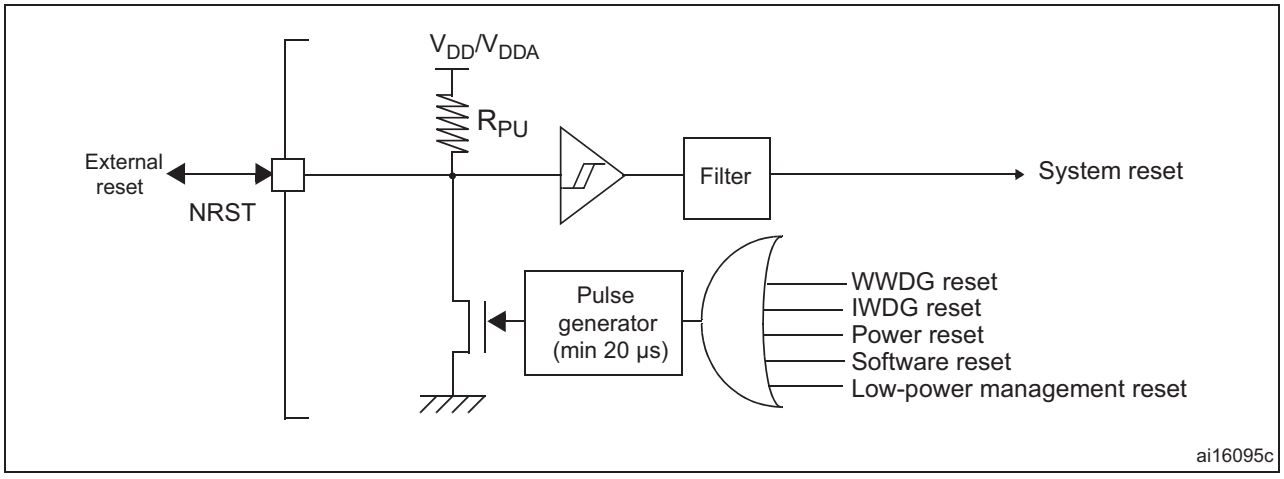
1. Вкл./Выкл. питания (сброс POR/PDR)
2. Выход из режима Standby

Устанавливает все регистры в предписанное состояние, кроме домена резервного хранения.

Эти источники работают как ножка NRST, которая удерживается в низком состоянии на время фазы задержки. Вектор программы сброса расположен по адресу 0x0000_0004 в карте памяти.

Сигнал системного сброса выдаётся на ножку NRST. Генератор импульса гарантирует минимальную длительность импульса 20 µs для обоих источников (внешнего или внутренного). В случае внешнего сброса импульс генерируется на всё время удержания ножки NRST низкой.

Рисунок 15. Упрощённая схема сброса.



6.1.3. Сброс резервного домена

Этот сброс ставит все регистры RTC и регистр RCC_BDCR в их исходное состояние. BKPSRAM не изменяется, его можно стереть только переходом уровня защиты из 1 в 0.

Здесь есть два способа запуска сброса:

1. Программный сброс установкой бита BDRST в регистре RCC_BDCR.
2. Включение одного из обоих выключенных V_{DD} или V_{BAT}.

6.2. Тактирование

Для системных тактов (SYSCLK) можно использовать три источника:

- такты генератора HSI
- такты генератора HSE
- такты основного PLL

Устройства имеют два вторичных источника тактов:

- 40 kHz низкоскоростной внутренний RC (LSI RC), для независимого сторожевого таймера и необязательно RTC для Автопробуждения из режима Stop/Standby.
- 32.768 kHz низкоскоростной внешний кварцевый генератор (LSE crystal) для возможного тактирования часов реального времени (RTCCLK).

Все источники можно включать и выключать независимо во имя экономии электронов.

Максимальная частота домена AHB равна 180 MHz, домена APB2 — 90 MHz, а APB1 — 45 MHz

Все свои такты периферия получает из SYSCLK, кроме:

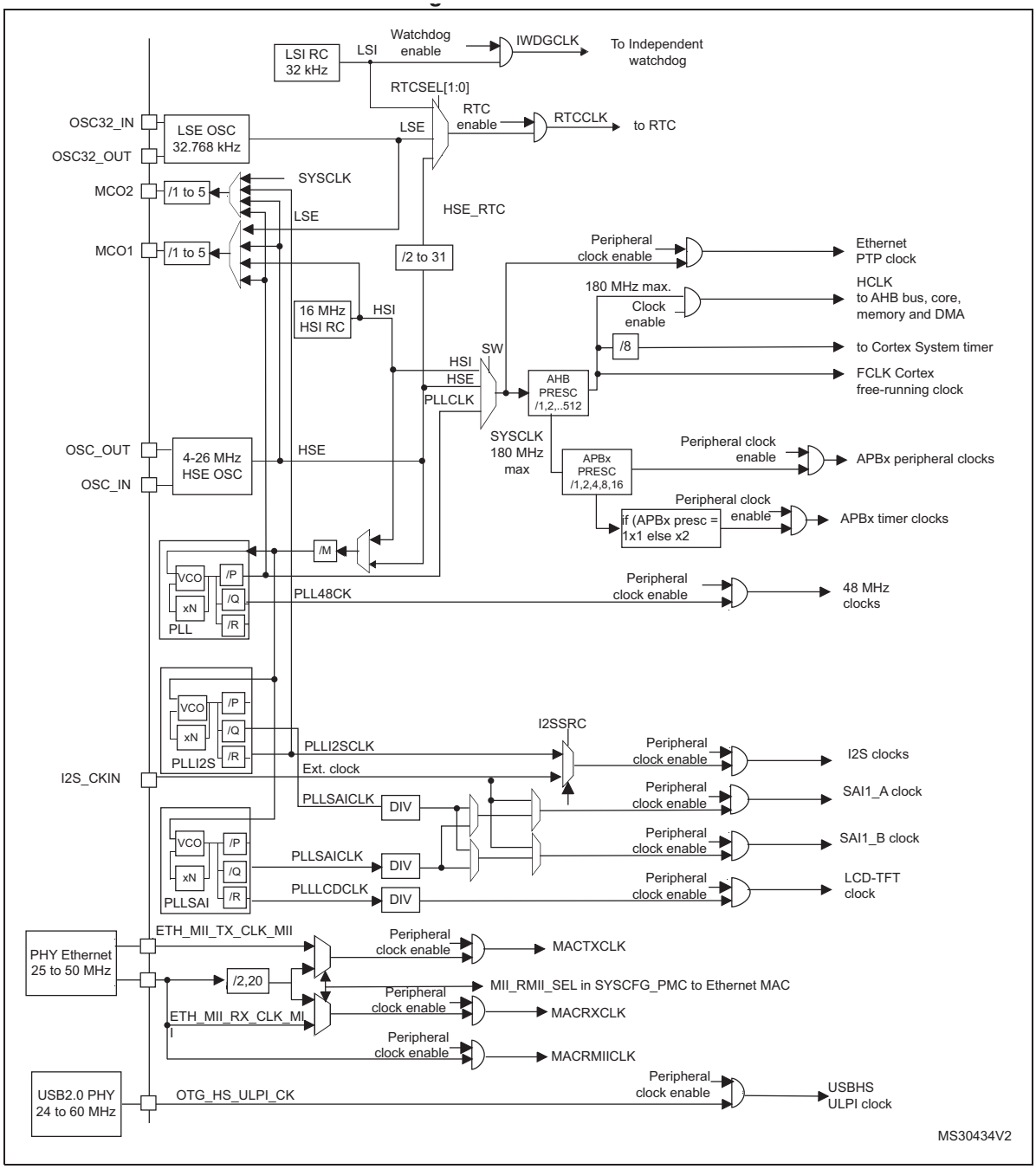
- USB OTG FS (48 MHz), случайный аналоговый генератор (RNG) (≤48 MHz) и SDIO (≤48 MHz) получают такты от отдельного выхода PLL (PLL48CLK)
- I2S
  Качества звука для, такты I2S можно получить от PLLI2S или снаружи по ножке I2S_CKIN. См. Секцию 28.4.4.
- SAI1
  Такты SAI1 получают от PLLSAI или PLLI2S, или снаружи по ножке I2S_CKIN.
  Если PLLI2S отведён для получения частоты выборки звука (49.152 MHz или 11.2896 MHz), то для тактов SAI1 можно использовать PLLSAI.
- LTDC
  Такты LTDC получают от PLLSAI.
- USB OTG HS (60 MHz) получает такты от внешнего PHY
- Ethernet MAC (TX, RX и RMII) получает такты от внешнего PHY. См. Секцию 33.4.4. При рабочем Ethernet частота тактов AHB должна быть не меньше 25 MHz.

RCC кормит SysTick тактами AHB (HCLK) / 8. Ещё SysTick может кушать такты HCLK.

Частоты тактов таймера задаются аппаратно. Два варианта зависят от бита TIMPRE в RCC_CFGR:

- При снятом бите TIMPRE в RCC_DKCFGR:
  Если предделитель APB равен 1, то TIMxCLK равен PCLKx. Иначе, TIMxCLK = 2xPCLKx.
- При стоящем бите TIMPRE в RCC_DKCFGR:
  Если предделитель APB равен 1, 2 или 4, то TIMxCLK = HCLK. Иначе, TIMxCLK = 4xPCLKx.

FCLK это свободно бегущие такты.



1. Детали есть в описании устройств.
2. Если бит TIMPRE регистра RCC_DCKCFGR сброшен и предделитель APBx равен 1, то TIMxCLK = PCLKx, иначе TIMxCLK = 2x PCLKx.
3. Если бит TIMPRE регистра RCC_DCKCFGR стоит и предделитель APBx равен 1,2 или 4, то TIMxCLK = HCLK, иначе TIMxCLK = 4x PCLKx.

Рисунок 16. Дерево тактирования

6.2.1. Такты HSE

Высокоскоростной внешний тактовый сигнал (HSE) можно получить из двух источников:

- внешний кварцевый/керамический резонатор HSE
- внешние такты HSE пользователя

Резонатор и конденсаторы нагрузки надо размещать как можно ближе к ножкам генератора для снижения помех и стабилизации времени запуска. Ёмкости нагрузки должны соответствовать выбранному генератору.

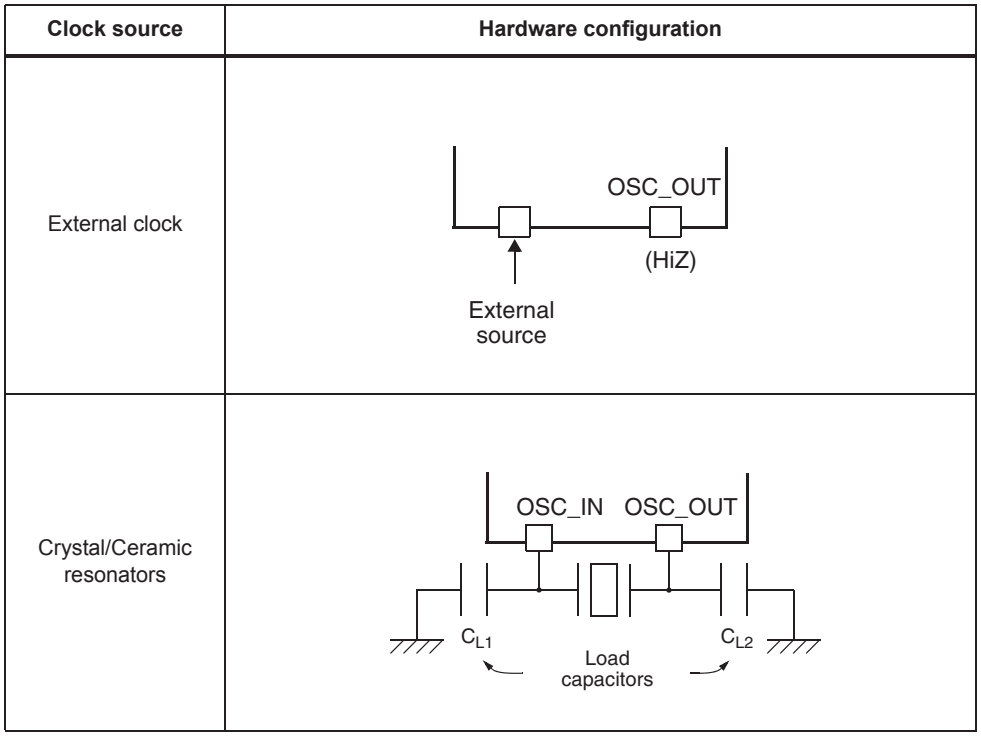


Рис. 17. Источники HSE/LSE

Внешний источник (HSE bypass)

В этом режиме нужен внешний источник. Режим включается битами `HSEBYP` и `HSEON` в регистре `RCC_CR`. Внешний сигнал (меандр, синус или треугольник) с ~50% заполнением подаётся на ножку OSC_IN, ножка OSC_OUT должна оставаться в третьем (hi-Z) состоянии. См. Рис. 17.

Внешний кварцевый/керамический резонатор (HSE crystal)

Выгода от внешнего генератора в очень точных основных тактах.

Стабильность внешнего генератора показывается флагом `HSERDY` в регистре `RCC_CR`. При включении такты не выпускаются на волю пока аппаратура не поставит этот бит. При этом может быть вызвано прерывание, если разрешено в регистре `RCC_CIR`.

HSE Crystal включается/выключается битом `HSEON` в регистре `RCC_CR`.

6.2.2. Такты HSI

Тактовый сигнал HSI выдаётся внутренним 16 MHz RC генератором и может использоваться как системные такты или подаваться на вход PLL (ФАПЧ).

Выгода HSI RC генератора в низкой стоимости. Да и запускается быстрее, чем HSE crystal генератор, но даже после калибровки точность всё равно ниже кварцевого или керамического.

Калибровка

Частоты RC генератора каждого чипа калибруются фирмой ST до 1% точности при TA=25°C.

После сброса, заводская калибровка загружается в биты `HSICAL[7:0]` регистра `RCC_CR`.

Если ваша система работает при колебаниях напряжения и температуры, то частота RC генератора может меняться. Подстроить частоту HSI можно битами `HSITRIM[4:0]` в регистре `RCC_CR`.

Флаг `HSIRDY` в регистре `RCC_CR` показывает стабильность HSI RC. При включении такты HSI RC не выпускаются на волю пока аппаратура не поставит этот бит.

HSI RC включается/выключается битом `HSION` в регистре `RCC_CR`.

Сигнал HSI можно использовать как резервный (Auxiliary clock) при сбое HSE. См. *Секцию 6.2.7*.

6.2.3. PLL (ФАПЧ)

MCU имеют три PLL:

- Основной PLL (PLL) тактируется от HSE или HSI и имеет два выхода:
  - Первый выход выдаёт системные такты высокой частоты (до 180 MHz)
  - Второй выдаёт такты для USB OTG FS (48 MHz), случайного генератора (≤48 MHz) и SDIO (≤48 MHz).

- Два специализированных PLL (PLLI2S и PLLSAI) выдают точную частоту для обработки звука на интерфейсах I2S и SAI1. PLLSAI также используется для тактов LCD-TFT.

Параметры основного PLL (выбор источника тактов HSI или HSE конфигурация делителей, M, N, P и Q) надо ставить до его включения.

PLLI2S и PLLSAI используют один вход тактов (биты PLLM[5:0] и PLLSRC), но раздельное разрешение и делители. (N and R). У включённых PLLI2S и PLLSAI параметры конфигурации менять нельзя.

При входе в режимы Stop и Standby или при отказе HSE или PLL (тактируемого от HSE), используемого для системных тактов, все три PLL аппаратно выключаются. Конфигурируются PLL, PLLI2S и PLLSAI в регистрах RCC_PLLCFGR, RCC_CFGR и RCC_DCKCFGR.

6.2.4. Такты LSE

LSE crystal это 32.768 kHz низкоскоростной внешний кварцевый или керамический резонатор. Его выгода в высокой точности сигнала для RTC и другой периферии.

Включается/выключается LSE битом LSEON в регистре резервного домена RCC_BDCR.

Флаг LSERDY в регистре RCC_BDCR показывает стабильность LSE crystal. При включении такты LSE crystal не выпускаются на волю пока аппаратура не поставит этот бит. При этом может быть вызвано прерывание, если разрешено в регистре RCC_CIR.

Внешний источник (LSE bypass)

Внешний источник частотой до 1 MHz выбирается установкой битов LSEBYP и LSEON в регистре RCC_BDCR. Внешний сигнал (меандр, синус или треугольник) с ~50% заполнением подаётся на ножку OSC32_IN, ножка OSC32_OUT должна оставаться в третьем (hi-Z) состоянии.

6.2.5. Такты LSI

Генератор LSI RC с частотой около 32 kHz продолжает работать в режимах Stop и Standby ради независимого сторожевого таймера (IWDG) и блока автопобудки (AWU).

Включается/выключается LSI RC битом LSION в регистре RCC_CSR.

Флаг LSIRDY в регистре RCC_CSR показывает стабильность LSE crystal. При включении такты LSE crystal не выпускаются на волю пока аппаратура не поставит этот бит. При этом может быть вызвано прерывание, если разрешено в регистре RCC_CIR.

6.2.6. Выбор SysClk

После системного сброса такты системы берутся от HSI. Источник системных тактов, подключенный напрямую, или через PLL остановить нельзя.

Переключение на новый источник тактов возможно только если он готов (стабилен после запуска или PLL зафиксирован). Если выбран неготовый источник, то он подключится только после готовности. Биты состояния в регистре RCC_CR показывают готовые генераторы и кто выбран системным.

6.2.7. Система безопасности тактирования (CSS)

CSS активируется программно и включается после задержки запуска HSE, выключается после остановки генератора.

Если в тактах HSE обнаруживается сбой, то он автоматически останавливается, таймерам (TIM1 и TIM8) посылается событие отключения входа и генерируется прерывание (Clock Security System Interrupt CSSI), подключённое к вектору исключения NMI.

NB: Если при включённой CSS возникает сбой HSE, прерывание NMI происходит независимо от того, очищен ли бит удержания CSS. Следовательно в NMI ISR нужно очистить прерывание CSS установкой бита CSSC в регистре RCC_CIR.

Если генератор HSE служит источником системных тактов напрямую или через PLL, то обнаруженный сбой переключает тактирование на HSI и выключает HSE, PLL (в случае его использования) и включённый PLLI2S.

6.2.8. Такты RTC/AWU

Источником тактов RTCCLK могут быть HSE 1 МГц после предделителя, LSE или LSI. Он выбирается битами RTCSEL[1:0] в регистре RCC_BDCR и битами RTCPRE[4:0] в регистре RCC_CFGR. Этот выбор нельзя изменить без сброса домена резервного хранения.

Если RTC тактируется от LSE, то он работает нормально при выключении системного или резервного питания. Если AWU тактируется от LSI, то при отключении системного питания его состояние не гарантируется. Если RTC тактируется от HSE с предделителем от 2 до 31, то его состояние при выключении системного или резервного питания не гарантируется.

Генератор LSE расположен в резервном домене, а HSE и LSI нет. Следовательно:

- Если RTC тактируется от LSE:
  - RTC продолжает работать при отключении V_{DD}, питаясь от V_{BAT}.
- Если блок авто-побудки (AWU) тактируется от LSI:
  - Состояние AWU при отключении V_{DD} не гарантируется. См. *Секцию 7.2.5*.
- Если RTC тактируется от HSE:
  - При отключении V_{DD} или регулятора напряжения (выключения 1.2 V домена) состояние RTC не гарантируется.

NB: Если частота тактов APB1 меньше семикратной частоты RTC ($f_{APB1} < 7xf_{RTCLCK}$), то читать регистр календаря надо читать дважды. А если не совпало, то и и трижды.

6.2.9. Такты сторожевого таймера

При аппаратном или программном запуске независимого сторожевого таймера (IWDG) генератор LSI включается и не выключается. Тактирование на IWDG пропускается после задержки запуска LSI.

6.2.10. Вывод тактового сигнала

Есть две ножки вывода тактового сигнала микроконтроллера (MCO):

- MCO1 (PA8)
  Можно вывести четыре сигнала с предделителем от 1 до 5:
  - HSI clock
  - LSE clock
  - HSE clock
  - PLL clock

  Это выбирается битами MCO1PRE[2:0] и MCO1[1:0] в регистре RCC_CFGR.
- MCO2 (PC9)
  Можно вывести четыре сигнала с предделителем от 1 до 5:
  - HSE clock
  - PLL clock
  - System clock (SYSCLK)
  - PLLI2S clock

  Это выбирается битами MCO2PRE[2:0] и MCO2[1:0] в регистре RCC_CFGR.

Порт GPIO для ножек MCO надо ставить в режим альтернативных функций,.

Выходная частота на MCO не должна превышать 100 MHz.

6.2.11. Измерение частоты тактов с помощью TIM5/TIM11

Частоту всех генераторов системы можно косвенно замерить с помощью захвата входа на TIM5 канал 4 и TIM11 канал 1. См. Рис. 18. и Рис. 19.

Измерение частоты тактов с помощью TIM5 канал 4

Входной мультиплексор TIM5 позволяет захватывать вход по сигналу I/O или по внутренним тактам. Это выбирается битами TI4_RMP [1:0] в регистре TIM5_OR.

LSE подключён к захвату входа канала 4 для точного измерения частоты HSI (когда он питает системные такты). Измеряется число тактов HSI между фронтами кварцевого LSE. Теперь можно вводить компенсацию отклонений по температуре и напряжению. Для этого у HSI есть биты калибровки.

А можно померить частоту сверх-маломощного, но не точного LSI в тактах HSI процедурой:

1. Включаем TIM5 и ставим канал 4 на захват входа.
2. Записью в биты TI4_RMP регистра TIM5_OR числа 0x01 подключаем такты LSI к захвату входа TIM5 канал 4.
3. Измеряем частоту тактов LSI по событию или прерыванию TIM5 захват/сравнение 4.
4. Обновляем предделитель RTC точности повышения для.

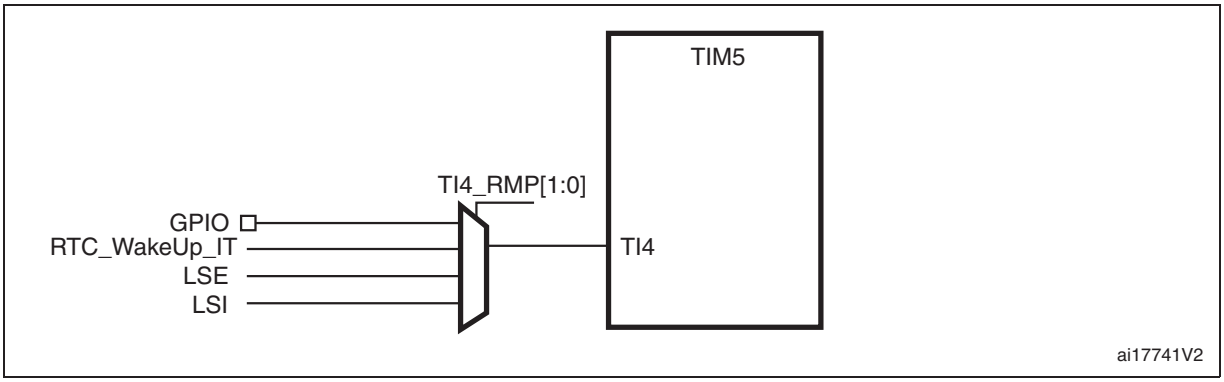


Рис. 18. Измерение частоты с помощью захвата входа TIM5

Измерение частоты тактов с помощью TIM11 канал 1

Входной мультиплексор TIM11 позволяет захватывать вход по сигналу I/O или по внутренним тактам. Это выбирается битами TI1_RMP [1:0] в регистре TIM11_OR.

Такты HSE_RTC (HSE/предделитель) подключаются к захвату входа канала 1 для грубого определения частоты внешнего кварца. Системные такты должно кормиться от HSI. Это полезно для соблюдения стандарта IEC 60730/IEC 61335, требующего возможности определения гармонических и субгармонических частот (с девиацией –50/+100%).

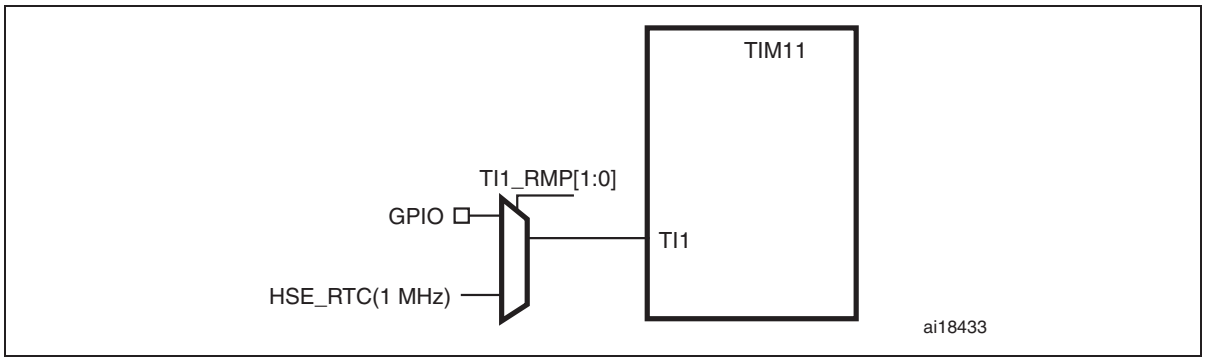


Рис. 19. Измерение частоты с помощью захвата входа TIM11

6.3. Регистры RCC

6.3.1. Регистр управления (RCC_CR)

Смещение адреса: 0x00

По сбросу: 0x0000 XX83 где X не определено.

Доступ: без ожидания, слово, полуслово и байт.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		PLLSAI RDY	PLLSAI ON	PLLI2S RDY	PLLI2S ON	PLLRD Y	PLLON	Reserved				CSS ON	HSE BYP	HSE RDY	HSE ON
		r	rw	r	rw	r	rw					rw	rw	r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HSICAL[7:0]								HSITRIM[4:0]					Res.	HSI RDY	HSION
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw		r	rw

— Биты 31:30 Резерв, не трогать.

— Бит 29 **PLLSAIRDY**: Флаг фиксации PLLSAI

Ставится аппаратно.

0: PLLSAI unlocked

1: PLLSAI locked

— Бит 28 **PLLSAION**: Включение PLLSAI.

Ставится и снимается программно. Снимается аппаратно при входе в Stop и Standby.

0: PLLSAI OFF

1: PLLSAI ON

— Бит 27 **PLLI2SRDY**: Флаг фиксации PLLI2S

Ставится аппаратно.

0: PLLI2S unlocked

1: PLLI2S locked

— Бит 26 **PLLI2SON**: Включение PLLI2S.

Ставится и снимается программно. Снимается аппаратно при входе в Stop и Standby.

0: PLLI2S OFF

1: PLLI2S ON

— Бит 25 **PLLRDY**: Флаг фиксации PLL.

Ставится аппаратно.

0: PLL unlocked

1: PLL locked

— Бит 24 **PLLON:** Включение PLL.

Ставится и снимается программно. Снимается аппаратно при входе в Stop и Standby. Если PLL используется как системные такты, то снять его невозможно.

0: PLL OFF

1: PLL ON

— Биты 23:20 Резерв, не трогать.

— Бит 19 **CSSON:** Включение системы безопасности.

Управляется программно. При стоящем CSSON детектор тактов аппаратно включается при готовом HSE и выключается при отказе HSE

0: Детектор OFF

1: Детектор ON (ON если HSE готов , OFF если нет).

— Бит 18 **HSEBYP:** Шунт (обход) **HSE**.

Управляется программно. Бит HSEON должен стоять. Бит HSEBYP можно писать только при выключенном HSE.

0: Внешний генератор не обойдён

1: Внешний генератор обойдён

— Бит 17 **HSERDY:** Флаг готовности HSE.

Ставится аппаратно при стабильном HSE. Снимается через 6 тактов HSE после снятия HSEON.

0: HSE не готов

1: HSE готов

— Бит 16 **HSEON:** Включение HSE.

Управляется программно. Снимается аппаратно при входе в режим Stop или Standby. Если HSE прямо или непрямо используется как источник системных тиков, то снять его невозможно.

0: HSE генератор OFF

1: HSE генератор ON

— Биты 15:8 **HSICAL[7:0]:** Калибровка HSI.

Установки инициализируются при запуске.

— Биты 7:3 **HSITRIM[4:0]:** Подстройка HSI.

Программная подстройка частоты внутреннего HSI RC при колебаниях напряжения и температуры.

Добавляется к значению HSICAL, подстраивает HSI RC.

— Бит 2 Резерв, не трогать.

— Бит 1 **HSIRDY:** Флаг готовности HSI.

Ставится аппаратно при стабильности внутреннего RC генератора. Снимается через 6 тактов HSI после выключения HSI.

0: HSI не готов

1: HSI готов

— Бит 0 **HSION:** Включение HSI.

Управляется программно. Ставится аппаратно при выходе из режимов Stop или Standby и в случае отказа внешнего генератора, используемого для системных тактов. Этот бит невозможно снять при прямом или непрямом использовании для системных тактов или выборе стать таковым.

0: HSI OFF

1: HSI ON

6.3.2. Регистр конфигурации PLL (RCC_PLLCFR)

Смещение адреса: 0x04

По сбросу: 0x2400 3010.

Доступ: без ожидания, слово, полуслово или байт.

Конфигурация выходных тактов PLL по формулам:

- $f_{(VCO\ clock)} = f_{(PLL\ clock\ input)} \times (PLLN / PLLM)$
- $f_{(PLL\ general\ clock\ output)} = f_{(VCO\ clock)} / PLLP$
- $f_{(USB\ OTG\ FS, SDIO, RNG\ clock\ output)} = f_{(VCO\ clock)} / PLLQ$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				PLLQ3	PLLQ2	PLLQ1	PLLQ0	Reserved	PLLSRC	Reserved				PLLP1	PLLP0
				rw	rw	rw	rw		rw					rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	PLLN									PLLM5	PLLM4	PLLM3	PLLM2	PLLM1	PLLM0
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:28 Резерв, не трогать.

— Биты 27:24 **PLLQ:** Делитель тактов PLL для USB OTG FS, SDIO и генератора случайных чисел

Ставится и снимается программно при выключенном PLL.

Внимание: USB OTG FS требует правильных тактов 48MHz. SDIO и RNG требуют правильных тактов ≤48MHz.

Частота тактов USB OTG FS = Частота VCO / PLLQ с 2 ≤PLLQ ≤15

0000: PLLQ = 0, плохой выбор

0001: PLLQ = 1, плохой выбор

0010: PLLQ = 2

0011: PLLQ = 3

0100: PLLQ = 4

…

1111: PLLQ = 15

— Бит 23 Резерв, не трогать.

— Бит 22 **PLLSRC:** Источник тактов PLL и PLLI2S

Ставится и снимается программно при выключенных PLL и PLLI2S.

0: Такты от HSI

1: Такты от HSE

— Биты 21:18 Резерв, не трогать.

— Биты 17:16 **PLLP:** Делитель PLL для системных тактов

Ставится и снимается программно при выключенном PLL.

Внимание: Выходная частота PLL не должна превышать 180 MHz для этого домена.

Частота PLL = Частота VCO / PLLP с PLLP = 2, 4, 6, или 8

00: PLLP = 2

01: PLLP = 4

10: PLLP = 6

11: PLLP = 8

— Бит 15 Резерв, не трогать.

— Биты 14:6 **PLLN**: Множитель PLL для VCO

Ставится и снимается программно при выключенном PLL. Доступ словом и полусловом

Внимание: Выходная частота VCO должна быть между 100 и 432 MHz.

Выходные такты VCO = Входные такты VCO × PLLN с 50 ≤PLLN ≤432

000000000: PLLN = 0, плохой выбор

000000001: PLLN = 1, плохой выбор

...

000110010: PLLN = 50

…

001100011: PLLN = 99

001100100: PLLN = 100

...

110110000: PLLN = 432

110110001: PLLN = 433, плохой выбор

…

111111111: PLLN = 511, плохой выбор

NB: Множитель от 50 и 99 возможен при входной частоте VCO больше 1 MHz.

— Биты 5:0 **PLLSM**: Делитель входной частоты для PLL и PLLI2S перед VCO

Ставится и снимается программно при выключенных PLL и PLLI2S.

Внимание: Входная частота VCO должна быть между 1 и 2 MHz. Рекомендуется 2 MHz для уменьшения дребезга PLL.

Входные такты VCO = Входные такты PLL × PLLM с 2 ≤PLLM ≤63

000000: PLLM = 0, wrong configuration

000001: PLLM = 1, wrong configuration

000010: PLLM = 2

000011: PLLM = 3

000100: PLLM = 4

…

111110: PLLM = 62

111111: PLLM = 63

6.3.3. Регистр конфигурации (RCC_CFR)

Смещение адреса: 0x08

По сбросу: 0x0000 0000.

Доступ: 0 ≤ такты ожидания ≤ 2, слово, полуслово или байт. 1 или 2 тактов ожидания появляются только при переключении источника тактов.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCO2		MCO2 PRE[2:0]			MCO1 PRE[2:0]			I2SSCR	MCO1		RTCPRE[4:0]				
rw		rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPRE2[2:0]			PPRE1[2:0]			Reserved		HPRE[3:0]				SWS1	SWS0	SW1	SW0
rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	r	r	rw	rw

— Биты 31:30 **MCO2**: Выход 2 тактов MCU

Ставится и снимается программно. Переключение источника может выдать глитчи на MCO2, так что надо ставить биты после сброса и перед включением внешних генераторов и PLL.

00: System clock (SYSCLK)

01: PLLI2S clock

10: HSE oscillator clock

11: PLL clock

— Биты 29:27 **MCO2 PRE[2:0]**: Предделитель MCO2

Ставится и снимается программно. Переключение предделителя может выдать глитчи на MCO2, так что надо ставить биты после сброса и перед включением внешних генераторов и PLL.

0xx: без деления

100: / 2

101: / 3

110: / 4

111: / 5

— Биты 26:24 **MCO1 PRE[2:0]**: Предделитель MCO1

Ставится и снимается программно. Переключение предделителя может выдать глитчи на MCO1, так что надо ставить биты после сброса и перед включением внешних генераторов и PLL.

0xx: без деления

100: / 2

101: / 3

110: / 4

111: / 5

— Бит 23 **I2SSRC**: Выбор источника тактов I2S

Ставится и снимается программно. Рекомендуется ставить биты после сброса и перед включением модуля I2S.

0: Такты от PLLI2S

1: Внешние такты, приведённые на ножку I2S_CKIN.

— Биты 22:21 **MCO1**: Выход 1 тактов MCU

Ставится и снимается программно. Переключение источника может выдать глитчи на MCO2, так что надо ставить биты после сброса и перед включением внешних генераторов и PLL.

00: HSI

01: LSE

10: HSE

11: PLL

— Биты 20:16 **RTC PRE[4:0]:** Предделитель HSE для получения тактов RTC 1 MHz

Ставится и снимается программно. Рекомендуется ставить биты до выбора источника тактов с RTC.

00000: тактов нет

00001: тактов нет

00010: HSE/2

00011: HSE/3

00100: HSE/4

...

11110: HSE/30

11111: HSE/31.

— Биты 15:13 **PPRE2**: Высокоскоростной предделитель APB2.

Пишется программно.

Внимание: Частота не должна превышать 90 MHz. Новое значение вступает в действие после от 1 до 19 тактов AHB.

0xx: AHB как есть

100: AHB / 2

101: AHB / 4

110: AHB / 8

111: AHB / 16

— Биты 12:10 **PPRE1**: Низкоскоростной предделитель APB1.

Пишется программно.

Внимание: Частота не должна превышать 45 MHz. Новое значение вступает в действие после от 1 до 19 тактов AHB.

0xx: AHB как есть

100: AHB / 2

101: AHB / 4

110: AHB / 8

111: AHB / 16

— Биты 9:8 Резерв, не трогать.

— Биты 7:4 **HPRE[3:0]**: Предделитель AHB.

Пишется программно.

Внимание: При использовании Ethernet частота должна быть не ниже 25 MHz. Новое значение вступает в действие после от 1 до 19 тактов AHB.

0xxx: SYSCLK как есть

1000: SYSCLK / 2

1001: SYSCLK / 4

1010: SYSCLK / 8

1011: SYSCLK / 16

1100: SYSCLK / 64

1101: SYSCLK / 128

1110: SYSCLK / 256

1111: SYSCLK / 512

— Биты 3:2 **SWS**: Состояние переключателя системных тактов.

Ставится аппаратно.

00: HSI

01: HSE
10: PLL
11: нету такого

— Биты 1:0 **SW**: Переключатель системных тактов.
Программно ставится для выбора источника SYSCLK.
Аппаратно переключается на HSI при выходе из режима Stop и Standby или при отказе генератора HSE при включённой системе контроля тактирования CSS.
00: HSI
01: HSE
10: PLL
11: не дозволено.

6.3.4. Регистр прерываний (RCC_CIR)

Смещение адреса: 0x0C
По сбросу: 0x0000 0000.
Доступ: без ожидания, слово, полуслово и байт.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								CSSC	PLLSAI RDYC	PLLI2S RDYC	PLL RDYC	HSE RDYC	HSI RDYC	LSE RDYC	LSI RDYC
								w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserv ed	PLLSAI RDYIE	PLLI2S RDYIE	PLL RDYIE	HSE RDYIE	HSI RDYIE	LSE RDYIE	LSI RDYIE	CSSF	PLLSAI RDYF	PLLI2S RDYF	PLL RDYF	HSE RDYF	HSI RDYF	LSE RDYF	LSI RDYF
	rw	rw	rw	rw	rw	rw	rw	r	r	r	r	r	r	r	r

— Биты 31:24 Резерв, не трогать.
— Бит 23 **CSSC**: Очистка флага прерывания CSS.
Пишется программно. 0: ничего,1: чистит
— Бит 22 **PLLSAIRDYC:** Очистка флага прерывания готовности PLLSAI.
Пишется программно. 0: ничего 1: чистит
— Бит 21 **PLLI2SRDYC:** Очистка флага прерывания готовности PLLI2S.
Пишется программно. 0: ничего 1: чистит
— Бит 20 **PLLRDYC:** Очистка флага прерывания готовности PLL.
Пишется программно. 0: ничего 1: чистит
— Бит 19 **HSERDYC:** Очистка флага прерывания готовности HSE.
Пишется программно. 0: ничего 1: чистит
— Бит 18 **HSIRDYC:** Очистка флага прерывания готовности HSI.
Пишется программно. 0: ничего 1: чистит
— Бит 17 **LSERDYC:** Очистка флага прерывания готовности LSE.
Пишется программно. 0: ничего 1: чистит
— Бит 16 **LSIRDYC:** Очистка флага прерывания готовности LSI.
Пишется программно. 0: ничего 1: чистит
— Бит 15 Резерв, не трогать.
— Бит 14 **PLLSAIRDYIE:** Разрешение прерывания готовности PLLSAI.
Ставится и снимается программно. 0: Выключено 1: Включено
— Бит 13 **PLLI2SRDYIE:** Разрешение прерывания готовности PLLI2S.
Ставится и снимается программно. 0: Выключено 1: Включено
— Бит 12 **PLLRDYIE:** Разрешение прерывания готовности PLL.
Ставится и снимается программно. 0: Выключено 1: Включено
— Бит 11 **HSERDYIE:** Разрешение прерывания готовности HSE.
Ставится и снимается программно. 0: Выключено 1: Включено
— Бит 10 **HSIRDYIE:** Разрешение прерывания готовности HSI.
Ставится и снимается программно. 0: Выключено 1: Включено
— Бит 9 **LSERDYIE:** Разрешение прерывания готовности LSE.
Ставится и снимается программно. 0: Выключено 1: Включено
— Бит 8 **LSIRDYIE:** Разрешение прерывания готовности LSI.
Ставится и снимается программно. 0: Выключено 1: Включено
— Бит 7 **CSSF:** Флаг CSS.

Ставится аппаратно при отказе HSE. Снимается записью бита CSSC.
0: Отказа нет 1: Отказ есть

— Бит 6 **PLLSAIRDYF:** Флаг прерывания готовности PLLSAI.
Ставится аппаратно при фиксации PLLSAI и стоящем PLLSAIRDYIE. Чистится записью бита PLLSAIRDYC.
0: Прерывания нет 1: Прерывание есть

— Бит 5 **PLLI2SRDYF:** Флаг прерывания готовности PLLI2S.
Ставится аппаратно при фиксации PLL и стоящем PLLI2SRDYIE. Чистится записью бита PLLI2SRDYC.
0: Прерывания нет 1: Прерывание есть

— Бит 4 **PLLRDYF:** Флаг прерывания готовности PLL.
Ставится аппаратно при фиксации PLL и стоящем PLLRDYDIE. Чистится записью бита PLLRDYC.
0: Прерывания нет 1: Прерывание есть

— Бит 3 **HSERDYF:** Флаг прерывания готовности HSE.
Ставится аппаратно при готовности HSE и стоящем HSERDYDIE. Чистится записью бита HSERDYC.
0: Прерывания нет 1: Прерывание есть

— Бит 2 **HSIRDYF:** Флаг прерывания готовности HSI.
Ставится аппаратно при готовности HSI и стоящем HSIRDYDIE. Чистится записью бита HSIRDYC.
0: Прерывания нет 1: Прерывание есть

— Бит 1 **LSERDYF:** Флаг прерывания готовности LSE.
Ставится аппаратно при готовности LSE и стоящем LSERDYDIE. Чистится записью бита LSERDYC.
0: Прерывания нет 1: Прерывание есть

— Бит 0 **LSIRDYF:** Флаг прерывания готовности LSI.
Ставится аппаратно при готовности LSI и стоящем LSIRDYDIE. Чистится записью бита LSIRDYC.
0: Прерывания нет 1: Прерывание есть

6.3.5. Регистр сброса периферии AHB1 (RCC_AHB1RSTR)

Смещение адреса: 0x10

По сбросу: 0x0000 0000.

Доступ: без ожидания, слово, полуслово и байт. Сброс записью "1".

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		OTGHS RST	Reserved			ETHMAC RST	Res.	DMA2D RST	DMA2 RST	DMA1 RST	Reserved				
		rw				rw		rw	rw	rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CRCRST	Res.	GPIOKRST	GPIOJRST	GPIOIRST	GPIOHRST	GPIOGGRST	GPIOFRST	GPIOERST	GPIODRST	GPIOCRST	GPIOBRST	GPIOARST
			rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:30 Резерв, не трогать.
— Бит 29 **OTGHSRST**: Сброс OTG HS.
— Биты 28:26 Резерв, не трогать.
— Бит 25 **ETHMACRST**: Сброс Ethernet MAC.
— Бит 24 Резерв, не трогать.
— Бит 23 **DMA2DRST**: Сброс DMA2D.
— Бит 22 **DMA2RST**: Сброс DMA2.
— Бит 21 **DMA1RST**: Сброс DMA1.
— Биты 20:16 Резерв, не трогать.
— Бит 12 **CRCRST:** Сброс CRC.
— Бит 11 Резерв, не трогать.
— Бит 10 **GPIOKRST:** Сброс GPIOK.
— Бит 9 **GPIOJRST:** Сброс GPIOJ.
— Бит 8 **GPIOIRST:** Сброс GPIOI.
— Бит 7 **GPIOHRST:** Сброс GPIOH.
— Бит 6 **GPIOGRST:** Сброс GPIOG.
— Бит 5 **GPIOFRST:** Сброс GPIOF.
— Бит 4 **GPIOERST:** Сброс GPIOE.

— Бит 3 **GPIODRST:** Сброс IGPIOD.
— Бит 2 **GPIOCRST:** Сброс GPIOC.
— Бит 1 **GPIOBRST:** Сброс GPIOB.
— Бит 0 **GPIOARST:** Сброс GPIOA.

6.3.6. Регистр сброса периферии AHB2 (RCC_AHB2RSTR)

Смещение адреса: 0x14

По сбросу: 0x0000 0000.

Доступ: без ожидания, слово, полуслово и байт. Сброс записью "1" в бит.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								OTGFS RST	RNG RST	HASH RST	CRYP RST	Reserved			DCMI RST
								rw	rw	rw	rw				rw

— Биты 31:8 Резерв, не трогать.
— Бит 7 **OTGFSRST:** Сброс OTG FS.
— Бит 6 **RNGRST:** Сброс RNG.
— Бит 5 **HASHRST:** Сброс HASH.
— Бит 4 **CRYPRST:** Сброс CRYP.
— Биты 3:1 Резерв, не трогать.
— Бит 0 **DCMIRST:** Сброс DCMI.

6.3.7. Регистр сброса периферии AHB3 (RCC_AHB3RSTR)

Смещение адреса: 0x18

По сбросу: 0x0000 0000.

Доступ: без ожидания, слово, полуслово и байт. Сброс записью "1" в бит.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															FMCRST
															rw

— Бит 0 **FMCRST:** Сброс FMC.

6.3.8. Регистр сброса периферии APB1 (RCC_APB1RSTR)

Смещение адреса: 0x20

По сбросу: 0x0000 0000.

Доступ: без ожидания, слово, полуслово и байт. Сброс записью "1".

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UART8RST	UART7RST	DACRST	PWR RST	Reser-ved	CAN2 RST	CAN1 RST	Reser-ved	I2C3 RST	I2C2 RST	I2C1 RST	UART5 RST	UART4 RST	UART3 RST	UART2 RST	Reser-ved
rw	rw	rw	rw			rw		rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 RST	SPI2 RST	Reserved		WWDG RST	Reserved		TIM14 RST	TIM13 RST	TIM12 RST	TIM7 RST	TIM6 RST	TIM5 RST	TIM4 RST	TIM3 RST	TIM2 RST
rw	rw			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

— Бит 31 **UART8RST**: Сброс UART8.
— Бит 30 **UART7RST**: Сброс UART7.
— Бит 29 **DACRST**: Сброс DAC.
— Бит 28 **PWRRST**: Сброс PWR.

— Бит 27 Резерв, не трогать.
— Бит 26 **CAN2RST**: Сброс CAN2.
— Бит 25 **CAN1RST**: Сброс CAN1.
— Бит 24 Резерв, не трогать.
— Бит 23 **I2C3RST**: Сброс I2C3.
— Бит 22 **I2C2RST**: Сброс I2C2.
— Бит 21 **I2C1RST**: Сброс I2C1.
— Бит 20 **UART5RST**: Сброс UART5.
— Бит 19 **UART4RST**: Сброс UART4.
— Бит 18 **UART3RST**: Сброс UART3.
— Бит 17 **UART2RST**: Сброс UART2.
— Бит 16 Резерв, не трогать.
— Бит 15 **SPI3RST**: Сброс SPI3.
— Бит 14 **SPI2RST**: Сброс SPI2.
— Биты 13:12 Резерв, не трогать.
— Бит 11 **WWDGRST:** Сброс WWDG.
— Биты 10:9 Резерв, не трогать.
— Бит 8 **TIM14RST:** Сброс TIM14.
— Бит 7 **TIM13RST:** Сброс TIM13.
— Бит 6 **TIM12RST:** Сброс TIM12.
— Бит 5 **TIM7RST:** Сброс TIM7.
— Бит 4 **TIM6RST:** Сброс TIM6.
— Бит 3 **TIM5RST:** Сброс TIM5.
— Бит 2 **TIM4RST:** Сброс TIM4.
— Бит 1 **TIM3RST:** Сброс TIM3.
— Бит 0 **TIM2RST:** Сброс TIM2.

6.3.9. Регистр сброса периферии APB2 (RCC_APB2RSTR)

Смещение адреса: 0x24

По сбросу: 0x0000 0000.

Доступ: без ожидания, слово, полуслово и байт. Сброс записью "1".

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					LTD CRST	Reserved			SAI1 RST	SPI6 RST	SPI5 RST	Res.	TIM11 RST	TIM10 RST	TIM9 RST
					rw				rw	rw	rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reser-ved	SYSCFG RST	SPI4 RST	SPI1 RST	SDIO RST	Reserved		ADC RST	Reserved		USART 6 RST	USART 1 RST	Reserved		TIM8 RST	TIM1 RST
	rw	rw	rw	rw			rw			rw	rw			rw	rw

— Биты 31:27 Резерв, не трогать.
— Бит 26 **LTDCRST**: Сброс LTDC.
— Биты 25:23 Резерв, не трогать.
— Бит 22 **SAI1RST**: Сброс SAI1.
— Бит 21 **SPI6RST**: Сброс SPI6.
— Бит 20 **SPI5RST**: Сброс SPI5.
— Бит 19 Резерв, не трогать.
— Бит 18 **TIM11RST**: Сброс TIM11.
— Бит 17 **TIM10RST**: Сброс TIM10.
— Бит 16 **TIM9RST**: Сброс TIM9.
— Бит 15 Резерв, не трогать.
— Бит 14 **SYSCFGRST**: Сброс SYSCFG.
— Бит 13 **SPI4RST**: Сброс SPI4.
— Бит 12 **SPI1RST**: Сброс SPI1.
— Бит 11 **SDIORST**: Сброс SDIO.

— Биты 10:9 Резерв, не трогать.
— Бит 8 **ADCRST**: Сброс ADC.
— Биты 7:6 Резерв, не трогать.
— Бит 5 **USART6RST**: Сброс USART6.
— Бит 4 **USART1RST**: Сброс USART1.
— Биты 3:2 Резерв, не трогать.
— Бит 1 **TIM8RST:** Сброс TIM8.
— Бит 0 **TIM1RST:** Сброс TIM1.

6.3.10. Регистр разрешения тактов периферии AHB1 (RCC_AHB1ENR)

Смещение адреса: 0x30

По сбросу: 0x0010 0000.

Доступ: без ожидания, слово, полуслово и байт.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reser-ved	OTGHS ULPIEN	OTGHSEN	ETHMACPTPEN	ETHMACRXEN	ETHMACTXEN	ETHMACEN	Res.	**DMA2DEN**	DMA2EN	DMA1EN	CCMDATARAMEN	Res.	BKPSRAMEN	Reserved	
	rw	rw	rw	rw	rw	rw		rw	rw	rw			rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CRCEN	Res.	GPIOKEN	GPIOJEN	GPIOIEN	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
			rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Читается и пишется программно.
0: Выключено
1: Включено

— Бит 31 Резерв, не трогать.
— Бит 30 **OTGHSULPIEN**: Тактирование OTG HSULPI.
— Бит 29 **OTGHSEN**: Тактирование OTG HS.
— Бит 28 **ETHMACPTPEN**: Тактирование Ethernet PTP.
— Бит 27 **ETHMACRXEN**: Тактирование Ethernet RX.
— Бит 26 **ETHMACTXEN**: Тактирование Ethernet TX.
— Бит 25 **ETHMACEN**: Тактирование Ethernet MAC.
— Бит 24 Резерв, не трогать.
— Бит 23 **DMA2DEN**: Тактирование DMA2D.
— Бит 22 **DMA2EN**: Тактирование DMA2.
— Бит 21 **DMA1EN**: Тактирование DMA1.
— Бит 20 **CCMDATARAMEN**: Тактирование CCM RAM.
— Бит 19 Резерв, не трогать.
— Бит 18 **BKPSRAMEN**: Тактирование Резервной SRAM.
— Бит 17:13 Резерв, не трогать.
— Бит 12 **CCREN**: Тактирование CRC.
— Бит 11 Резерв, не трогать.
— Бит 10 **GPIOKEN**: Тактирование GPIOK.
— Бит 9 **GPIOJEN**: Тактирование GPIOJ.
— Бит 8 **GPIOIEN**: Тактирование GPIOI.
— Бит 7 **GPIOHEN**: Тактирование GPIOH.
— Бит 6 **GPIOGEN**: Тактирование GPIOG.
— Бит 5 **GPIOFEN**: Тактирование GPIOF.
— Бит 4 **GPIOEEN**: Тактирование GPIOE.
— Бит 3 **GPIODEN**: Тактирование GPIOD.
— Бит 2 **GPIOCEN**: Тактирование GPIOC.
— Бит 1 **GPIOBEN**: Тактирование GPIOB.
— Бит 0 **GPIOAEN**: Тактирование GPIOA.

6.3.11. Регистр разрешения тактов периферии AHB2 (RCC_AHB2ENR)

Смещение адреса: 0x34

По сбросу: 0x0000 0000.

Доступ: без ожидания, слово, полуслово и байт.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								OTGFS EN	RNG EN	HASH EN	CRYP EN	Reserved			DCMI EN
								rw	rw	rw	rw				rw

Читается и пишется программно.

0: Выключено

1: Включено

— Биты 31:8 Резерв, не трогать.

— Бит 7 **OTGFSEN:** Тактирование OTG FS.

— Бит 6 **RNGEN:** Тактирование RNG.

— Бит 5 **HASHEN:** Тактирование HASH.

— Бит 4 **CRYPEN:** Тактирование CRYP.

— Биты 3:1 Резерв, не трогать.

— Бит 0 **DCMIEN:** Тактирование DCMI.

6.3.12. Регистр разрешения тактов периферии AHB3 (RCC_AHB3ENR)

Смещение адреса: 0x38

По сбросу: 0x0000 0000.

Доступ: без ожидания, слово, полуслово и байт.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															FMCEN
															rw

Читается и пишется программно.

0: Выключено

1: Включено

— Биты 31:1 Резерв, не трогать.

— Бит 0 **FMCEN:** Тактирование FMC.

6.3.13. Регистр разрешения тактов периферии APB1 (RCC_APB1ENR)

Смещение адреса: 0x40

По сбросу: 0x0000 0000.

Доступ: Без ожиданий, слово, полуслово и байт.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UART8 EN	UART7 EN	DAC EN	PWR EN	Reser-ved	CAN2 EN	CAN1 EN	Reser-ved	I2C3 EN	I2C2 EN	I2C1 EN	UART5 EN	UART4 EN	USART 3 EN	USART 2 EN	Reser-ved
rw	rw	rw	rw		rw	rw		rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 EN	SPI2 EN	Reserved		WWDG EN	Reserved		TIM14 EN	TIM13 EN	TIM12 EN	TIM7 EN	TIM6 EN	TIM5 EN	TIM4 EN	TIM3 EN	TIM2 EN
rw	rw			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

Читается и пишется программно.

0: Выключено

1: Включено

— Бит 31 **UART8EN**: Тактирование UART8.
— Бит 30 **UART7EN**: Тактирование UART7.
— Бит 29 **DACEN**: Тактирование DAC.
— Бит 28 **PWREN**: Тактирование PWR.
— Бит 27 Резерв, не трогать.
— Бит 26 **CAN2EN**: Тактирование CAN2.
— Бит 25 **CAN1EN:** Тактирование CAN1.
— Бит 24 Резерв, не трогать.
— Бит 23 **I2C3EN:** Тактирование I2C3.
— Бит 22 **I2C2EN:** Тактирование I2C2.
— Бит 21 **I2C1EN:** Тактирование I2C1.
— Бит 20 **UART5EN:** Тактирование UART5.
— Бит 19 **UART4EN:** Тактирование UART4.
— Бит 18 **USART3EN:** Тактирование USART3.
— Бит 17 **USART2EN:** Тактирование USART2.
— Бит 16 Резерв, не трогать.
— Бит 15 **SPI3EN:** Тактирование SPI3.
— Бит 14 **SPI2EN:** Тактирование SPI2.
— Биты 13:12 Резерв, не трогать.
— Бит 11 **WWDGEN:** Тактирование WWDG.
— Биты 10:9 Резерв, не трогать.
— Бит 8 **TIM14EN:** Тактирование TIM14.
— Бит 7 **TIM13EN:** Тактирование TIM13.
— Бит 6 **TIM12EN:** Тактирование TIM12.
— Бит 5 **TIM7EN:** Тактирование TIM7.
— Бит 4 **TIM6EN:** Тактирование TIM6.
— Бит 3 **TIM5EN:** Тактирование TIM5.
— Бит 2 **TIM4EN:** Тактирование TIM4.
— Бит 1 **TIM3EN:** Тактирование TIM3.
— Бит 0 **TIM2EN:** Тактирование TIM2.

6.3.14. Регистр разрешения тактов периферии APB2 (RCC_APB2ENR)

Смещение адреса: 0x44

По сбросу: 0x0000 0000.

Доступ: Без ожиданий, слово, полуслово и байт.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					LTDC EN	Reserved			SAI1EN	SPI6EN	SPI5EN	Res.	TIM11 EN	TIM10 EN	TIM9 EN
					rw				rw	rw	rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reser-ved	SYSCFG EN	SPI4EN	SPI1 EN	SDIO EN	ADC3 EN	ADC2 EN	ADC1 EN	Reserved		USART 6 EN	USART 1 EN	Reserved		TIM8 EN	TIM1 EN
	rw	rw	rw	rw	rw	rw	rw			rw	rw			rw	rw

Читается и пишется программно.
0: Выключено
1: Включено

— Биты 31:27 Резерв, не трогать.
— Бит 26 **LTDCEN**: Тактирование LTDC.
— Биты 25:23 Резерв, не трогать.
— Бит 22 **SAI1EN**: Тактирование SAI1.
— Бит 21 **SPI6EN:** Тактирование SPI6.
— Бит 21 **SPI5EN:** Тактирование SPI5.
— Бит 19 Резерв, не трогать.
— Бит 18 **TIM11EN:** Тактирование TIM11.
— Бит 17 **TIM10EN:** Тактирование TIM10.

— Бит 16 **TIM9EN:** Тактирование TIM9.
— Бит 15 Резерв, не трогать.
— Бит 14 **SYSCFGEN:** Тактирование SYSCFG.
— Бит 13 **SPI4EN:** Тактирование SPI4.
— Бит 12 **SPI1EN:** Тактирование SPI1.
— Бит 11 **SDIOEN:** Тактирование SDIO.
— Бит 10 **ADC3EN:** Тактирование ADC3.
— Бит 9 **ADC2EN:** Тактирование ADC2.
— Бит 8 **ADC1EN:** Тактирование ADC1.
— Биты 7:6 Резерв, не трогать.
— Бит 5 **USART6EN:** Тактирование USART6.
— Бит 4 **USART1EN:** Тактирование USART1.
— Биты 3:2 Резерв, не трогать.
— Бит 1 **TIM8EN:** Тактирование TIM8.
— Бит 0 **TIM1EN:** Тактирование TIM1.

6.3.15. Регистр разрешения тактов периферии AHB1 в экономном режиме (RCC_AHB1LPENR)

Смещение адреса: 0x50

По сбросу: 0x7EEF 97FF.

Доступ: без ожидания, слово, полуслово и байт.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	OTGHS ULPILPEN	OTGHS LPEN	ETHPTP LPEN	ETHRX LPEN	ETHTX LPEN	ETHMAC LPEN	Res.	DMA2D LPEN	DMA2 LPEN	DMA1 LPEN	Res.	SRAM3 LPEN	BKPSRAM LPEN	SRAM2 LPEN	SRAM1 LPEN
	rw	rw	rw	rw	rw	rw		rw	rw	rw		rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLITF LPEN	Reserved		CRC LPEN	Res.	GPIOK LPEN	GPIOIJ LPEN	GPIOI LPEN	GPIOH LPEN	GPIOG G LPEN	GPIO F LPEN	GPIOE LPEN	GPIOD LPEN	GPIOC LPEN	GPIOB LPEN	GPIOA LPEN
rw			rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Читается и пишется программно.
0: Выключено
1: Включено

— Бит 31 Резерв, не трогать.
— Бит 30 **OTGHSULPIEN**: Тактирование OTG HSULPI.
— Бит 29 **OTGHSLPEN**: Тактирование OTG HS.
— Бит 28 **ETHMACPTPLPEN**: Тактирование Ethernet PTP.
— Бит 27 **ETHMACRXLPEN**: Тактирование Ethernet RX.
— Бит 26 **ETHMACTXLPEN**: Тактирование Ethernet TX.
— Бит 25 **ETHMACLPEN**: Тактирование Ethernet MAC.
— Бит 24 Резерв, не трогать.
— Бит 23 **DMA2DLPEN**: Тактирование DMA2D.
— Бит 22 **DMA2LPEN**: Тактирование DMA2.
— Бит 21 **DMA1LPEN**: Тактирование DMA1.
— Бит 20 Резерв, не трогать.
— Бит 19 **SRAM3LPEN**: Тактирование SRAM3.
— Бит 18 **BKPSRAMLPEN**: Тактирование Резервной SRAM.
— Бит 17 **SRAM2LPEN**: Тактирование SRAM2.
— Бит 16 **SRAM1LPEN**: Тактирование SRAM1.
— Бит 15 **FLITFLPEN**: Тактирование FLITF.
— Бит 17:13 Резерв, не трогать.
— Бит 12 **CCRLPEN**: Тактирование CRC.
— Бит 11 Резерв, не трогать.
— Бит 10 **GPIOKLPEN**: Тактирование GPIOK.
— Бит 9 **GPIOJLPEN**: Тактирование GPIOJ.
— Бит 8 **GPIOILPEN**: Тактирование GPIOI.

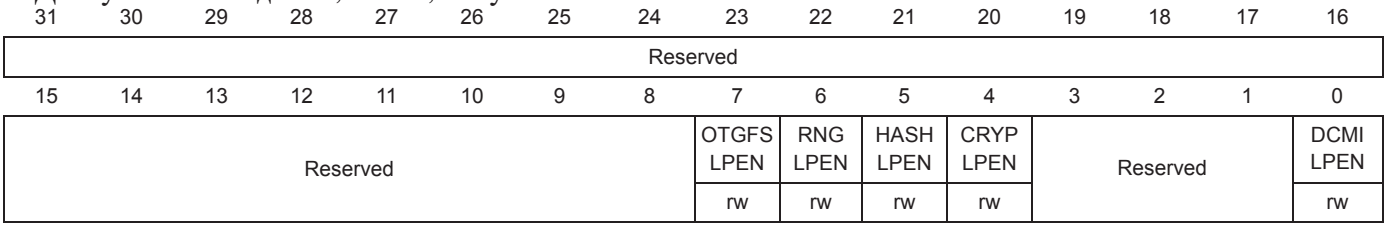
— Бит 7 **GPIOHLPEN**: Тактирование GPIOH.
— Бит 6 **GPIOGLPEN**: Тактирование GPIOG.
— Бит 5 **GPIOFLPEN**: Тактирование GPIOF.
— Бит 4 **GPIOELPEN**: Тактирование GPIOE.
— Бит 3 **GPIODLPEN**: Тактирование GPIOD.
— Бит 2 **GPIOCLPEN**: Тактирование GPIOC.
— Бит 1 **GPIOBLPEN**: Тактирование GPIOB.
— Бит 0 **GPIOALPEN**: Тактирование GPIOA.

6.3.16. Регистр разрешения тактов периферии AHB2 в экономном режиме (RCC_AHB2LPENR)

Смещение адреса: 0x54

По сбросу: 0x0000 00F1.

Доступ: без ожидания, слово, полуслово и байт.



Читается и пишется программно.
0: Выключено
1: Включено

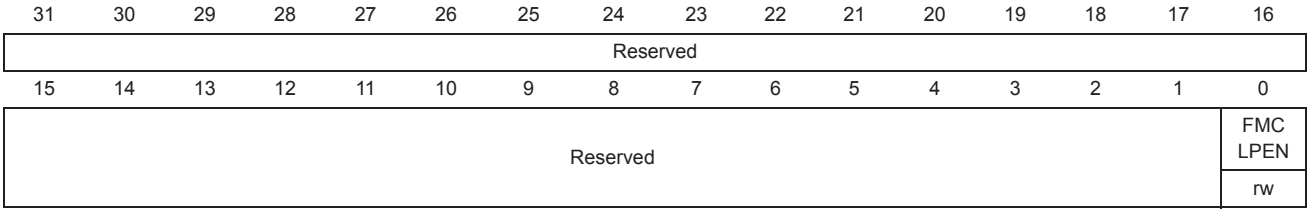
— Биты 31:8 Резерв, не трогать.
— Бит 7 **OTGFSLPEN:** Тактирование OTG FS.
— Бит 6 **RNGLPEN:** Тактирование RNG.
— Бит 5 **HASHLPEN:** Тактирование HASH.
— Бит 4 **CRYPLPEN:** Тактирование CRYP.
— Биты 3:1 Резерв, не трогать.
— Бит 0 **DCMILPEN:** Тактирование DCMI.

6.3.17. Регистр разрешения тактов периферии AHB3 в экономном режиме (RCC_AHB3LPENR)

Смещение адреса: 0x58

По сбросу: 0x0000 0001.

Доступ: без ожидания, слово, полуслово и байт.



Читается и пишется программно.
0: Выключено
1: Включено

— Биты 31:1 Резерв, не трогать.
— Бит 0 **FMCLPEN:** Тактирование FMC.

6.3.18. Регистр разрешения тактов периферии APB1 в экономном режиме (RCC_APB1LPENR)

Смещение адреса: 0x60

По сбросу: 0xF6FE C9FF.

Доступ: Без ожиданий, слово, полуслово и байт.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UART8 LPEN	UART7 LPEN	DAC LPEN	PWR LPEN	RESERVED	CAN2 LPEN	CAN1 LPEN	Reser-ved	I2C3 LPEN	I2C2 LPEN	I2C1 LPEN	UART5 LPEN	UART4 LPEN	USART 3 LPEN	USART 2 LPEN	Reser-ved
rw	rw	rw	rw		rw	rw		rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 LPEN	SPI2 LPEN	Reserved		WWDG LPEN	Reserved		TIM14 LPEN	TIM13 LPEN	TIM12 LPEN	TIM7 LPEN	TIM6 LPEN	TIM5 LPEN	TIM4 LPEN	TIM3 LPEN	TIM2 LPEN
rw	rw			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

Читается и пишется программно.

0: Выключено

1: Включено

— Бит 31 **UART8LPEN**: Тактирование UART8.
— Бит 30 **UART7LPEN**: Тактирование UART7.
— Бит 29 **DACLPEN**: Тактирование DAC.
— Бит 28 **PWRLPEN**: Тактирование PWR.
— Бит 27 Резерв, не трогать.
— Бит 26 **CAN2LPEN**: Тактирование CAN2.
— Бит 25 **CAN1LPEN:** Тактирование CAN1.
— Бит 24 Резерв, не трогать.
— Бит 23 **I2C3LPEN:** Тактирование I2C3.
— Бит 22 **I2C2LPEN:** Тактирование I2C2.
— Бит 21 **I2C1LPEN:** Тактирование I2C1.
— Бит 20 **UART5LPEN:** Тактирование UART5.
— Бит 19 **UART4LPEN:** Тактирование UART4.
— Бит 18 **USART3LPEN:** Тактирование USART3.
— Бит 17 **USART2LPEN:** Тактирование USART2.
— Бит 16 Резерв, не трогать.
— Бит 15 **SPI3LPEN:** Тактирование SPI3.
— Бит 14 **SPI2LPEN:** Тактирование SPI2.
— Биты 13:12 Резерв, не трогать.
— Бит 11 **WWDGLPEN:** Тактирование WWDG.
— Биты 10:9 Резерв, не трогать.
— Бит 8 **TIM14LPEN:** Тактирование TIM14.
— Бит 7 **TIM13LPEN:** Тактирование TIM13.
— Бит 6 **TIM12LPEN:** Тактирование TIM12.
— Бит 5 **TIM7LPEN:** Тактирование TIM7.
— Бит 4 **TIM6LPEN:** Тактирование TIM6.
— Бит 3 **TIM5LPEN:** Тактирование TIM5.
— Бит 2 **TIM4LPEN:** Тактирование TIM4.
— Бит 1 **TIM3LPEN:** Тактирование TIM3.
— Бит 0 **TIM2LPEN:** Тактирование TIM2.

6.3.19. Регистр разрешения тактов периферии APB2 в экономном режиме (RCC_APBLP2ENR)

Смещение адреса: 0x64

По сбросу: 0x0477 7F33.

Доступ: Без ожиданий, слово, полуслово и байт.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					LTDC LPEN	Reserved			SAI1 LPEN	SPI6 LPEN	SPI5 LPEN	Reser-ved	TIM11 LPEN	TIM10 LPEN	TIM9 LPEN
					rw				rw	rw	rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reser-ved	SYSCFG LPEN	SPI4 LPEN	SPI1 LPEN	SDIO LPEN	ADC3 LPEN	ADC2 LPEN	ADC1 LPEN	Reserved		USART 6 LPEN	USART 1 LPEN	Reserved		TIM8 LPEN	TIM1 LPEN
	rw	rw	rw	rw	rw	rw	rw			rw	rw			rw	rw

Читается и пишется программно.

0: Выключено

1: Включено

— Биты 31:27 Резерв, не трогать.

— Бит 26 **LTDCLPEN**: Тактирование LTDC.

— Биты 25:23 Резерв, не трогать.

— Бит 22 **SAI1LPEN**: Тактирование SAI1.

— Бит 21 **SPI6LPEN:** Тактирование SPI6.

— Бит 21 **SPI5LPEN:** Тактирование SPI5.

— Бит 19 Резерв, не трогать.

— Бит 18 **TIM11LPEN:** Тактирование TIM11.

— Бит 17 **TIM10LPEN:** Тактирование TIM10.

— Бит 16 **TIM9LPEN:** Тактирование TIM9.

— Бит 15 Резерв, не трогать.

— Бит 14 **SYSCFGLPEN:** Тактирование SYSCFG.

— Бит 13 **SPI4LPEN:** Тактирование SPI4.

— Бит 12 **SPI1LPEN:** Тактирование SPI1.

— Бит 11 **SDIOLPEN:** Тактирование SDIO.

— Бит 10 **ADC3LPEN:** Тактирование ADC3.

— Бит 9 **ADC2LPEN:** Тактирование ADC2.

— Бит 8 **ADC1LPEN:** Тактирование ADC1.

— Биты 7:6 Резерв, не трогать.

— Бит 5 **USART6LPEN:** Тактирование USART6.

— Бит 4 **USART1LPEN:** Тактирование USART1.

— Биты 3:2 Резерв, не трогать.

— Бит 1 **TIM8LPEN:** Тактирование TIM8.

— Бит 0 **TIM1LPEN:** Тактирование TIM1.

6.3.20. Регистр управления резервным доменом (RCC_BDCR)

Смещение адреса: 0x70

По сбросу Резервного Домена: 0x0000 0000.

Доступ: 0 ≤ тактов ожидания ≤ 3, слово, полуслово и байт. Такты ожидания добавляются при успешном доступе к регистру.

NB: Биты LSEON, LSEBYP, RTCSEL и RTCEN регистра RCC_BDCR лежат в резервном домене. Значит после Сброса они защищены от записи и перед их изменением надо установить бит DBP в регистре PWR_CR. См. *Секцию 6.1.1*. Обнуляются они только Сбросом резервного домена и ничем

иным (см. *Секцию 6.1.3*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															BDRST
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTCEN	Reserved					RTCSEL[1:0]		Reserved					LSEBYP	LSERDY	LSEON
rw						rw	rw						rw	r	rw

— Биты 31:17 Резерв, не трогать.

— Бит 16 **BDRST:** Программный сброс резервного домена.

Читается и пишется программно.

0: Ничего

1: Сбрасывает весь домен

NB: Этот сброс на BKPSRAM не влияет, нужен переход уровня защиты с уровня 1 на 0.

— Бит 15 **RTCEN:** Тактирование RTC.

Читается и пишется программно.

0: Выключено

1: Включено

— Биты 14:10 Резерв, не трогать.

— Биты 9:8 **RTCSEL[1:0]:** Выбор источника тактов RTC.

Пишется программно однократно. Сбрасывается только Сбросом всего домена битом BDRST.

00: Нету

01: LSE

10: LSI

11: HSE с предделителем (выбор битами RTCPRE[4:0] в регистре RCC_CFGR

— Биты 7:3 Резерв, не трогать.

— Бит 2 **LSEBYP:** Шунт (обход) LSE.

Читается и пишется программно для обхода LSE в режиме отладки. Пишется только при выключенном LSE.

0: LSE не шунтирован

1: LSE шунтирован

— Бит 1 **LSERDY:** Готовность LSE.

Читается и пишется аппаратно при готовности внешнего 32 КГц LSE. После снятия бита LSEON бит LSERDY снимется через 6 тактов генератора LSE.

0: Не готово

1: Готово

— Бит 0 **LSEON:** Разрешение LSE.

Читается и пишется программно.

0: Выключено

1: Включено

6.3.21. Регистр состояния/управления (RCC_CSR)

Смещение адреса: 0x74

По сбросу: 0x0E00 0000. Сброс системным сбросом, флаги только сбросом питания.

Доступ: 0 ≤ тактов ожидания ≤ 3, слово, полуслово и байт. Такты ожидания добавляются при успешном доступе к регистру.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPWRRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	PORRSTF	PINRSTF	BORRSTF	RMVF	Reserved							
r	r	r	r	r	r	r	rt_w								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														LSIRDY	LSION
														r	rw

Биты флагов ставятся аппаратно. Чистятся записью бита RMVF.

0: Сброса не было

1: Сброс был

— Бит 31 **LPWRRSTF:** Флаг сброса экономного питания.

— Бит 30 **WWDGRSTF:** Флаг сброса WWDG.

— Бит 29 **IWDGRSTF:** Флаг сброса IWDG.

— Бит 28 **SFTRSTF:** Флаг программного сброса.

— Бит 27 **PORRSTF:** Флаг сброса POR/PDR.

— Бит 26 **PINRSTF:** Флаг сброса PIN.

— Бит 25 **BORRSTF:** Флаг сброса BOR.

— Бит 24 **RMVF:** Снятие флагов сброса.

Пишется программно для снятия флагов сброса.

0: Ничего

1: Флаги сброса снимаются

— Биты 23:2 Резерв, не трогать.

— Бит 1 **LSIRDY:** Готовность LSI.

Читается и пишется аппаратно при готовности LSI. После снятия бита LSION бит LSIRDY снимется через 3 такта генератора.

0: Не готово

1: Готово

— Бит 0 **LSION:** Разрешение LSI.

Читается и пишется программно.

0: Выключено

1: Включено

6.3.22. Регистр генератора тактов широкого спектра (RCC_SSCGR)

Смещение адреса: 0x80

По сбросу: 0x0E00 0000.

Доступ: без тактов ожидания, слово, полуслово и байт.

Генерация тактов широкого спектра есть только на главном PLL. Регистр RCC_SSCGR надо писать перед включением или после выключения главного PLL.

NB: Подробности как всегда в описаниях устройств.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SSCGEN	SPREADSEL	Reserved		INCSTEP											
rw	rw			rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INCSTEP				MODPER											
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Бит 31 **SSCGEN:** Включение модуляции широкого спектра

Читается и пишется программно.

0: ВЫКЛ. (Писать после очистки CR[24]=PLLON бит)

1: ВКЛ. (Писать перед установкой CR[24]=PLLON бит)

— Бит 30 **SPREADSEL:** Выбор ширины

Читается и пишется программно.

Писать перед установкой CR[24]=PLLON бит.

0: Централь

1: Спад

— Биты 29:28 Резерв, не трогать.

— Биты 27:13 **INCSTEP:** Шаг инкремента

Читается и пишется программно. Писать перед установкой CR[24]=PLLON бит. Амплитуда профиля модуляции.

— Биты 12:0 **MODPER:** Период модуляции.

Читается и пишется программно. Писать перед установкой CR[24]=PLLON бит. Период профиля модуляции.

6.3.23. Регистр конфигурации PLLI2S (RCC_PLLI2SCFGR)

Смещение адреса: 0x84

По сбросу: 0x2400 3000.

Доступ: без тактов ожидания, слово, полуслово и байт.

Конфигурация выходов PLLI2S по формулам:

$f_{(VCO\ clock)} = f_{(PLLI2S\ clock\ input)} \times (PLLI2SN / PLLM)$

$f_{(PLL\ I2S\ clock\ output)} = f_{(VCO\ clock)} / PLLI2SR$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	PLLI2SR2	PLLI2SR1	PLLI2SR0	PLLI2SQ				Reserved							
	rw	rw	rw	rw	rw	rw	rw								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	PLLI2SN8	PLLI2SN7	PLLI2SN6	PLLI2SN5	PLLI2SN4	PLLI2SN3	PLLI2SN2	PLLI2SN1	PLLI2SN0	Reserved					
	rw	rw	rw	rw	rw	rw	rw	rw	rw						

— Бит 31 Резерв, не трогать.

— Биты 30:28 **PLLI2SR:** Делитель тактов для I2S

Ставится и снимается программно при выключенном PLLI2S. Вместе с предделителем периферии I2S должен дать ошибку 0.3% со стандартным кварцем и 0% с звуковым кварцем. См. *Секцию 28.4.4*.

Внимание: I2S требует частот ≤ 192MHz.

Частота тактов I2S = Частота VCO / PLLR с 2 ≤PLLR ≤7

000: PLLR = 0, Не то

001: PLLR = 1, Не то

010: PLLR = 2

…

111: PLLR = 7

— Биты 27:24 **PLLI2SQ:** Делитель тактов для SAI1

Ставится и снимается программно при стоящем PLLI2S.

Частота тактов I2S = Частота VCO / PLLI2SQ с 2 ≤PLLI2SQ ≤15

0000: PLLI2SQ = 0, Не можно

0001: PLLI2SQ = 1, Не можно

0010: PLLI2SQ = 2

0011: PLLI2SQ = 3

0100: PLLI2SQ = 4

0101: PLLI2SQ = 5

…

1111: PLLI2SQ = 15

— Биты 23:15 Резерв, не трогать.

— Биты 14:6 **PLLI2SN**: Множитель PLLI2S для VCO

Пишется словами и полусловами программно при выключенном PLLI2S.

Внимание: Выходная частота VCO должна быть от 100 до 432 MHz.

Выходная частота VCO = Входная частота VCO × PLLI2SN при 50 ≤PLLI2SN ≤432

000000000: PLLI2SN = 0, Нельзя

000000001: PLLI2SN = 1, Нельзя

...

000110010: PLLI2SN = 50

…

001100011: PLLI2SN = 99

001100100: PLLI2SN = 100

001100101: PLLI2SN = 101

001100110: PLLI2SN = 102

…

110110000: PLLI2SN = 432

110110001: PLLI2SN = 433, Нельзя

…

111111111: PLLI2SN = 511, Нельзя

NB: Множитель от 50 до 99 возможен для входной частоты VCO больше 1 MHz.

— Биты 5:0 Резерв, не трогать.

6.3.24. Регистр конфигурации PLL (RCC_PLLCFGR)

Смещение адреса: 0x88

По сбросу: 0x2400 3000.

Доступ: без тактов ожидания, слово, полуслово и байт.

Конфигурация выходов PLLSAI по формулам:

$f_{(VCO\ clock)} = f_{(PLLI2S\ clock\ input)} \times (PLLSAIN / PLLM)$

$f_{(PLL\ SAI1\ clock\ output)} = f_{(VCO\ clock)} / PLLSAIQ$

$f_{(PLL\ LCD\ clock\ output)} = f_{(VCO\ clock)} / PLLSAIR$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	PLLSAIR			PLLSAIQ				Reserved							
	rw	rw	rw	rw	rw	rw	rw								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	PLLSAIN									Reserved					
	rw	rw	rw	rw	rw	rw	rw	rw	rw						

— Бит 31 Резерв, не трогать.

— Биты 30:28 **PLLSAIR:** Делитель тактов для LCD

Ставится и снимается программно при выключенном PLLSAI.

Частота тактов LCD = Частота VCO / PLLSAIR с 2 ≤PLLSAIR ≤7

000: PLLSAIR = 0, Не то

001: PLLSAIR = 1, Не то

010: PLLSAIR = 2

…

111: PLLSAIR = 7

— Биты 27:24 **PLLSAIQ:** Делитель тактов для SAI1

Ставится и снимается программно при стоящем PLLSAI.

Частота тактов SAI = Частота VCO / PLLSAIQ с 2 ≤ PLLSAIQ ≤15

0000: PLLSAIQ = 0, Не можно

0001: PLLSAIQ = 1, Не можно

0010: PLLSAIQ = 2

0011: PLLSAIQ = 3

0100: PLLSAIQ = 4

0101: PLLSAIQ = 5

…

1111: PLLSAIQ = 15

— Биты 23:15 Резерв, не трогать.

— Биты 14:6 **PLLI2SN**: Множитель PLLSAI для VCO

Пишется словами и полусловами программно при выключенном PLLSAI.

Внимание: Выходная частота VCO должна быть от 100 до 432 MHz.

Выходная частота VCO = Входная частота VCO × PLLSAIN при 50 ≤ PLLSAIN ≤432

000000000: PLLSAIN = 0, Нельзя

000000001: PLLSAIN = 1, Нельзя

...

000110010: PLLSAIN = 50

…

001100011: PLLSAIN = 99

001100100: PLLSAIN = 100

001100101: PLLSAIN = 101

001100110: PLLSAIN = 102

…

110110000: PLLSAIN = 432

110110001: PLLSAIN = 433, Нельзя

…

111111111: PLLSAIN = 511, Нельзя

NB: Множитель от 50 до 99 возможен для входной частоты VCO больше 1 MHz.

— Биты 5:0 Резерв, не трогать.

6.3.25. Отдельный регистр конфигурации RCC (RCC_DCKCFGR)

Смещение адреса: 0x8C

По сбросу: 0x0000 0000.

Доступ: без тактов ожидания, слово, полуслово и байт.

Конфигурация предделителей тактов таймеров и делителей выходных тактов PLLSAI и PLLI2S для SAI1и LTDC по формулам:

$f_{(PLLSAIDIVQ\ clock\ output)} = f_{(PLLSAI\_Q)}$ / PLLSAIDIVQ

$f_{(PLLSAIDIVR\ clock\ output)} = f_{(PLLSAI\_R)}$ / PLLSAIDIVR

$f_{(PLLI2SDIVQ\ clock\ output)} = f_{(PLLI2S\_Q)}$ / PLLI2SDIVQ

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved							TIMPRE	SAI1BSRC		SAI1ASRC		Reserved		PLLSAIDIVR	
							rw	rw	rw	rw	rw			rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			PLLSAIDIVQ					Reserved			PLLS2DIVQ				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

— Биты 31:25 Резерв, не трогать.

— Бит 24 **TIMPRE:** Выбор предделителей таймеров домена APB1 и APB2.

Ставится и снимается программно.

0: Если предделитель APB (PPRE1, PPRE2 в регистре RCC_CFGR) равен 1, то TIMxCLK = PCLKx. Иначе, TIMxCLK = 2xPCLKx.

1: Если предделитель APB (PPRE1, PPRE2 в регистре RCC_CFGR) равен 1, 2 или 4, TIMxCLK = HCLK. Иначе, TIMxCLK = 4xPCLKx.

— Биты 23:22 **SAI1BSRC**: Выбор источника SAI1-B

Ставится и снимается программно при выключенных PLLSAI и PLLI2S.

00: Частота SAI1-B = $f_{(PLLSAI\_Q)}$ / PLLSAIDIVQ

01: Частота SAI1-B = $f_{(PLLI2S\_Q)}$ / PLLI2SDIVQ

10: Частота SAI1-B = Частота входа альтернативной функции

11: Лажа

— Биты 21:19 **SAI1ASRC**: Выбор источника SAI1-A

Ставится и снимается программно при выключенных PLLSAI и PLLI2S.

00: Частота SAI1-A = $f_{(PLLSAI\_Q)}$ / PLLSAIDIVQ

01: Частота SAI1-A = $f_{(PLLI2S\_Q)}$ / PLLI2SDIVQ

10: Частота SAI1-A = Частота входа альтернативной функции

11: Лажа

— Биты 19:18 Резерв, не трогать.

— Биты 17:16 **PLLSAIDIVR**: Делитель для LCD_CLK

Ставится и снимается программно при выключенном PLLSAI.

Частота LCD_CLK = $f_{(PLLSAI\_R)}$ / PLLSAIDIVR с 2 ≤PLLSAIDIVR ≤16

00: PLLSAIDIVR = /2

01: PLLSAIDIVR = /4

10: PLLSAIDIVR = /8

11: PLLSAIDIVR = /16

— Биты 15:13 Резерв, не трогать.

— Биты 12:8 **PLLSAIDIVQ**: Делитель тактов PLLSAI для SAI1

Ставится и снимается программно при выключенном PLLSAI.

Частота SAI1 = $f_{(PLLSAI\_Q)}$ / PLLSAIDIVQ с 1 ≤PLLSAIDIVQ ≤31

00000: PLLSAIDIVQ = /1

00001: PLLSAIDIVQ = /2

00010: PLLSAIDIVQ = /3

00011: PLLSAIDIVQ = /4

00100: PLLSAIDIVQ = /5
…
11111: PLLSAIDIVQ = /32

— Биты 7:5 Резерв, не трогать.

— Биты 4:0 **PLLS2DIVQ**: Делитель PLLI2S для тактов SAI1

Ставится и снимается программно при выключенном PLLI2S.

Частота SAI1 = $f_{(PLLI2S\_Q)}$ / PLLI2SDIVQ с 1 <= PLLI2SDIVQ <= 31

00000: PLLI2SDIVQ = /1
00001: PLLI2SDIVQ = /2
00010: PLLI2SDIVQ = /3
00011: PLLI2SDIVQ = /4
00100: PLLI2SDIVQ = /5
…
11111: PLLI2SDIVQ = /32

6.3.26. Карта регистров RCC

Addr. offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	RCC_CR	Reserved		PLL SAIRDY	PLL SAION	PLL I2SRDY	PLL I2SON	PLL RDY	PLL ON	Reserved				CSSON	HSEBYP	HSERDY	HSEON	HSICAL 7	HSICAL 6	HSICAL 5	HSICAL 4	HSICAL 3	HSICAL 2	HSICAL 1	HSICAL 0	HSITRIM 4	HSITRIM 3	HSITRIM 2	HSITRIM 1	HSITRIM 0	Reserved	HSIRDY	HSION
0x04	RCC_PLLCFGR	Reserved				PLLQ 3	PLLQ 2	PLLQ 1	PLLQ 0	Reserved	PLLSRC	Reserved				PLLP 1	PLLP 0	Reserved	PLLN 8	PLLN 7	PLLN 6	PLLN 5	PLLN 4	PLLN 3	PLLN 2	PLLN 1	PLLN 0	PLLM 5	PLLM 4	PLLM 3	PLLM 2	PLLM 1	PLLM 0
0x08	RCC_CFGR	MCO2 1	MCO2 0	MCO2PRE2	MCO2PRE1	MCO2PRE0	MCO1PRE2	MCO1PRE1	MCO1PRE0	I2SSRC	MCO1 1	MCO1 0	RTCPRE 4	RTCPRE 3	RTCPRE 2	RTCPRE 1	RTCPRE 0	PPRE2 2	PPRE2 1	PPRE2 0	PPRE1 2	PPRE1 1	PPRE1 0	Reserved		HPRE 3	HPRE 2	HPRE 1	HPRE 0	SWS 1	SWS 0	SW 1	SW 0	
0x0C	RCC_CIR	Reserved								CSSC	PLLSAIRDYC	PLLI2SRDYC	PLLRDYC	HSERDYC	HSIRDYC	LSERDYC	LSIRDYC	Reserved	PLLSAIRDYIE	PLLI2SRDYIE	PLLRDYIE	HSERDYIE	HSIRDYIE	LSERDYIE	LSIRDYIE	CSSF	PLLSAIRDYF	PLLI2SRDYF	PLLRDYF	HSERDYF	HSIRDYF	LSERDYF	LSIRDYF
0x10	RCC_AHB1RSTR	Reserved		OTGHSRST	Reserved			ETHMACRST	Reserved	DMA2DRST	DMA2RST	DMA1RST	Reserved								CRCRST	Reserved	GPIOKRST	GPIOJRST	GPIOIRST	GPIOHRST	GPIOGRST	GPIOFRST	GPIOERST	GPIODRST	GPIOCRST	GPIOBRST	GPIOARST
0x14	RCC_AHB2RSTR	Reserved																								OTGFSRST	RNGRST	HSAHRST	CRYPRST	Reserved			DCMIRST
0x18	RCC_AHB3RSTR	Reserved																															FMCRST
0x1C	Reserved	Reserved																															
0x20	RCC_APB1RSTR	UART8RST	UART7RST	DACRST	PWRRST	Reserved	CAN2RST	CAN1RST	Reserved	I2C3RST	I2C2RST	I2C1RST	UART5RST	UART4RST	UART3RST	UART2RST	Reserved	SPI3RST	SPI2RST	Reserved			WWDGRST	Reserved		TIM14RST	TIM13RST	TIM12RST	TIM7RST	TIM6RST	TIM5RST	TIM4RST	TIM3RST	TIM2RST
0x24	RCC_APB2RSTR	Reserved					LTDCRST	Reserved		SAI1RST	SPI6RST	SPI5RST	Reserved	TIM11RST	TIM10RST	TIM9RST	Reserved	SYSCFGRST	SP45RST	SPI1RST	SDIORST	Reserved		ADCRST	Reserved		USART6RST	USART1RST	Reserved		TIM8RST	TIM1RST
0x28	Reserved	Reserved																															
0x2C	Reserved	Reserved																															
0x30	RCC_AHB1ENR	Reserved	OTGHSULPIEN	OTGHSEN	ETHMACPTPEN	ETHMACRXEN	ETHMACTXEN	ETHMACEN	Reserved	DMA2DEN	DMA2EN	DMA1EN	CCMDATARAMEN	Reserved	BKPSRAMEN	Reserved					CRCEN	Reserved	GPIOKEN	GPIOJEN	GPIOIEN	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
0x34	RCC_AHB2ENR	Reserved																								OTGFSEN	RNGEN	HASHEN	CRYPEN	Reserved			DCMIEN
0x38	RCC_AHB3ENR	Reserved																															FMCEN
0x3C	Reserved	Reserved																															

Addr. offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x40	RCC_APB1ENR	UART8EN	UART7EN	DACEN	PWREN	Reserved	CAN2EN	CAN1EN	Reserved	I2C3EN	I2C2EN	I2C1EN	UART5EN	UART4EN	USART3EN	USART2EN	Reserved	SPI3EN	SPI2EN	Reserved		WWDGEN	Reserved		TIM14EN	TIM13EN	TIM12EN		TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN
0x44	RCC_APB2ENR	Reserved					LTDCEN	Reserved			SAI1EN	SPI6EN	SPI5EN	Reserved	TIM11EN	TIM10EN	TIM9EN	Reserved	SYSCFGEN	SPI4EN	SPI1EN	SDIOEN	ADC3EN	ADC2EN	ADC1EN	Reserved		USART6EN	USART1EN	Reserved		TIM8EN	TIM1EN
0x48	Reserved	Reserved																															
0x4C	Reserved	Reserved																															
0x50	RCC_AHB1LPENR	Reserved	OTGHSULPILPEN	OTGHSLPEN	ETHMACPTPLPEN	ETHMACRXLPEN	ETHMACTXLPEN	ETHMACLPEN	Reserved	DMA2DLPEN	DMA2LPEN	DMA1LPEN	Reserved	SRAM3LPEN	BKPSRAMLPEN	SRAM2LPEN	SRAM1LPEN	FLITFLPEN	Reserved		CRCLPEN	Reserved	GPIOKLPEN	GPIOJLPEN	GPIOILPEN	GPIOHLPEN	GPIOGLPEN	GPIOFLPEN	GPIOELPEN	GPIODLPEN	GPIOCLPEN	GPIOBLPEN	GPIOALPEN
0x54	RCC_AHB2LPENR	Reserved																								OTGFSLPEN	RNGLPEN	HASHLPEN	CRYPLPEN	Reserved			DCMILPEN
0x58	RCC_AHB3LPENR	Reserved																															FMCLPEN
0x5C	Reserved	Reserved																															
0x60	RCC_APB1LPENR	UART8LPEN	UART7LPEN	DACLPEN	PWRLPEN	Reserved	CAN2LPEN	CAN1LPEN	Reserved	I2C3LPEN	I2C2LPEN	I2C1LPEN	UART5LPEN	UART4LPEN	USART3LPEN	USART2LPEN	Reserved	SPI3LPEN	SPI2LPEN	Reserved		WWDGLPEN	Reserved		TIM14LPEN	TIM13LPEN	TIM12LPEN		TIM7LPEN	TIM6LPEN	TIM5LPEN	TIM4LPEN	TIM3LPEN	TIM2LPEN
0x64	RCC_APB2LPENR	Reserved					LTDCLPEN	Reserved			SAI1LPEN	SPI6LPEN	SPI5LPEN	Reserved	TIM11LPEN	TIM10LPEN	TIM9LPEN	Reserved	SYSCFGLPEN	SPI4LPEN	SPI1LPEN	SDIOLPEN	ADC3LPEN	ADC2LPEN	ADC1LPEN	Reserved		USART6LPEN	USART1LPEN	Reserved		TIM8LPEN	TIM1LPEN
0x68	Reserved	Reserved																															
0x6C	Reserved	Reserved																															
0x70	RCC_BDCR	Reserved															BDRST	RTCEN	Reserved				RTCSEL 1	RTCSEL 0	Reserved							LSEBYP	LSERDY	LSEON
0x74	RCC_CSR	LPWRRSTF	WWDGRSTF	WDGRSTF	SFTRSTF	PORRSTF	PADRSTF	BORRSTF	RMVF	Reserved																						LSIRDY	LSION
0x78	Reserved	Reserved																															
0x7C	Reserved	Reserved																															
0x80	RCC_SSCGR	SSCGEN	SPREADSEL	Reserved							INCSTEP												MODPER										
0x84	RCC_PLLI2SCFGR	Reserved	PLLI2SRx				PLLI2SQ				Reserved								PLLI2SNx										Reserved				
0x88	RCC_PLLSAICFGR		PLLSAIR				PLLSAIQ				Reserved								PLLSAIN										Reserved				
0x8C	RCC_DCKCFGR	Reserved							TIMPRE	SAI1BSCR	SAI1ASCR	Reserved	PLLSAIDIVR		Reserved		PLLSAIDIVQ			Reserved		PLLI2SDIVQ										

7. Сброс и тактирование STM32F405xx/07xx и STM32F415xx/17xx(RCC)

7.1. Сброс

Есть три типа сброса: системный, питания и резервного домена.

7.1.1. Системный сброс

Сбрасываются все регистры, кроме флагов сброса в регистре RCC_CSR и регистров резервного домена.

Системный сброс выполняется в следующих случаях:

1. Низкий уровень на ножке NRST (внешний сброс)
2. Конец счёта WWDG
3. Конец счёта IWDG reset
4. Программный сброс (SW reset)
5. Сброс управления экономным питанием.

Источник сброса отмечается в регистре RCC_CSR.

Программный сброс

Для этого нужно установить бит SYSRESETREQ в регистре Application Interrupt and Reset Control Register. Источник сброса есть регистре RCC_CSR.

Сброс управления пониженным питанием

Есть два способа:

1. Вход в режим Standby:
   Разрешается сбросом бита nRST_STDBY в Байтах Опций Пользователя. В этом случае при успешном выполнении последовательности входа в режим Standby устройство просто сбрасывается и всё.
2. Вход в режим Stop:
   Разрешается сбросом бита nRST_STOP в Байтах Опций Пользователя. В этом случае при успешном выполнении последовательности входа в режим Stop устройство просто сбрасывается.

7.1.2. Сброс питания

Генерируется в следующих случаях:

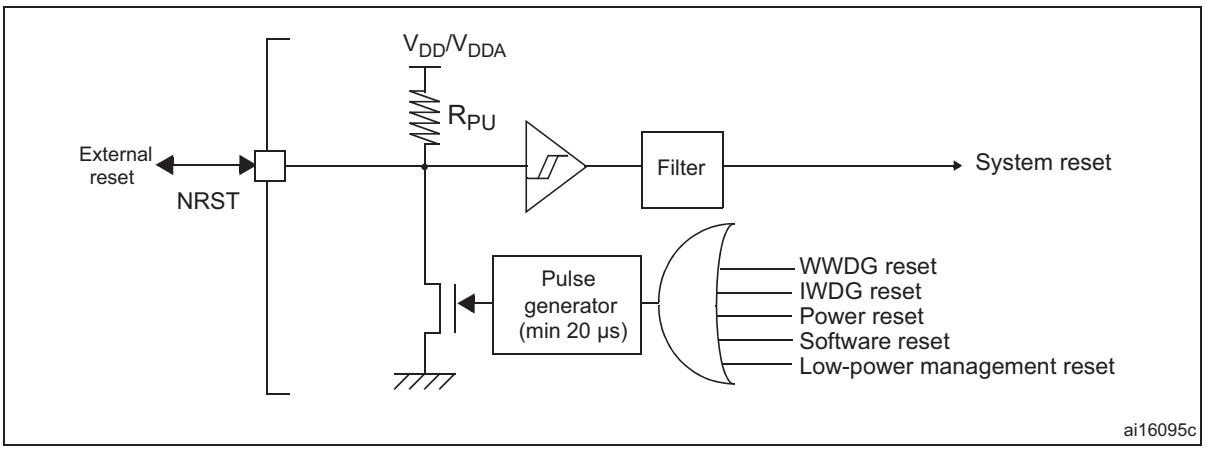
1. Вкл./Выкл. питания (сброс POR/PDR)
2. Выход из режима Standby

Устанавливает все регистры в предписанное состояние, кроме домена резервного хранения.

Эти источники работают как ножка NRST, которая удерживается в низком состоянии на время фазы задержки. Вектор программы сброса расположен по адресу 0x0000_0004 в карте памяти.

Сигнал системного сброса подаётся на ножку NRST. Генератор импульса гарантирует минимальную длительность импульса 20 µs для обоих источников (внешнего или внутренного). В случае внешнего сброса импульс генерируется на всё время удержания ножки NRST низкой.

Упрощённая схема сброса.



7.1.3. Сброс резервного домена

Ставит все регистры RTC и регистр RCC_BDCR в предписанное состояние. BKPSRAM не сбрасывается, его сбрасывают изменением уровня защиты Flash с 1 на 0.

Здесь есть два типа сброса:

1. Программный сброс установкой бита BDRST в регистре RCC_BDCR.
2. Включение одного из обоих выключенных V_{DD} или V_{BAT}.

7.2. Тактирование

Для системных тактов (SYSCLK) можно использовать три источника:

- такты генератора HSI
- такты генератора HSE
- такты PLL

Устройства имеют два вторичных источника тактов:

- 32 kHz низкоскоростной внутренний RC (LSI RC), для независимого сторожевого таймера и необязательно RTC для Автопробуждения из режима Stop/Standby.
- 32.768 kHz низкоскоростной внешний кварцевый генератор (LSE crystal) для возможного тактирования часов реального времени (RTCCLK).

Во имя экономии электронов все источники можно включать и выключать независимо.

Контроллер имеет гибкий выбор между внешним и внутренним генераторами для тактов ядра и периферии с наивысшей частотой и гарантии нужных частот для Ethernet, USB OTG FS и HS, I2S и SDIO.

Несколько предделителей допускают конфигурацию частот доменов AHB, высокоскоростного APB (APB2) и низкоскоростного APB (APB1). Максимальная частота домена AHB равна 168 MHz, домена APB2 равна 84 MHz. Максимальная допустимая частота домена APB1 равна 42 MHz.

Вся периферия тактируется от SYSCLK кроме:

- Тактов USB OTG FS (48 MHz), генератора случайного (RNG) (≤48 MHz) и SDIO (≤48 MHz), что поступают от своего выхода PLL (PLL48CLK)
- Тактов I2S
  Их можно получить от своего PLL (PLLI2S) или внешнего источника, приведённого на ножку I2S_CKIN. См. Секцию 28.4.4.
- Тактов USB OTG HS (60 MHz) от внешнего PHY
- Тактов Ethernet MAC (TX, RX и RMII) от внешнего PHY. См. Секцию 33.4.4. При использовании Ethernet частота AHB должна быть не меньше 25 MHz.

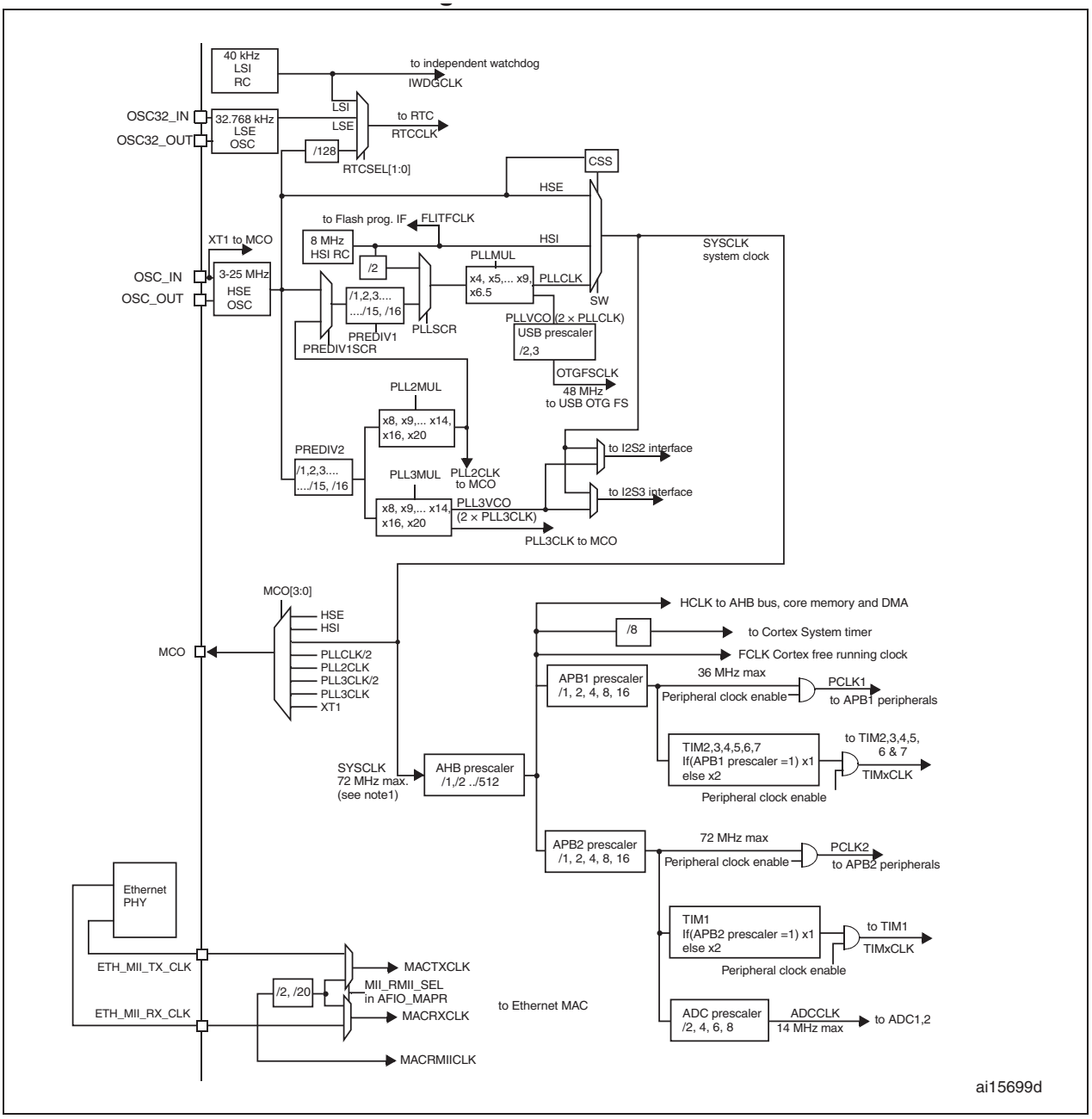
RCC кормит системный таймер (SysTick) внешними тактами AHB HCLK/8. SysTick может работать от них или от HCLK, что выбирается в регистре управления и состояния SysTick. ADC тактируются частотой высокоскоростного домена (APB2) делённой by 2, 4, 6 или 8.

Частоты тактов таймера фиксируются аппаратно:

1. если предделитель APB равен 1, то они равны частоте своего домена APB.
2. иначе они равны удвоенной (×2) частоте своего домена APB.

FCLK работает как независимый тактовый генератор.

Рис. 21. Дерево тактирования



1. При питании PLL от HSI максимальная частота системных тиков может достичь 36 MHz.

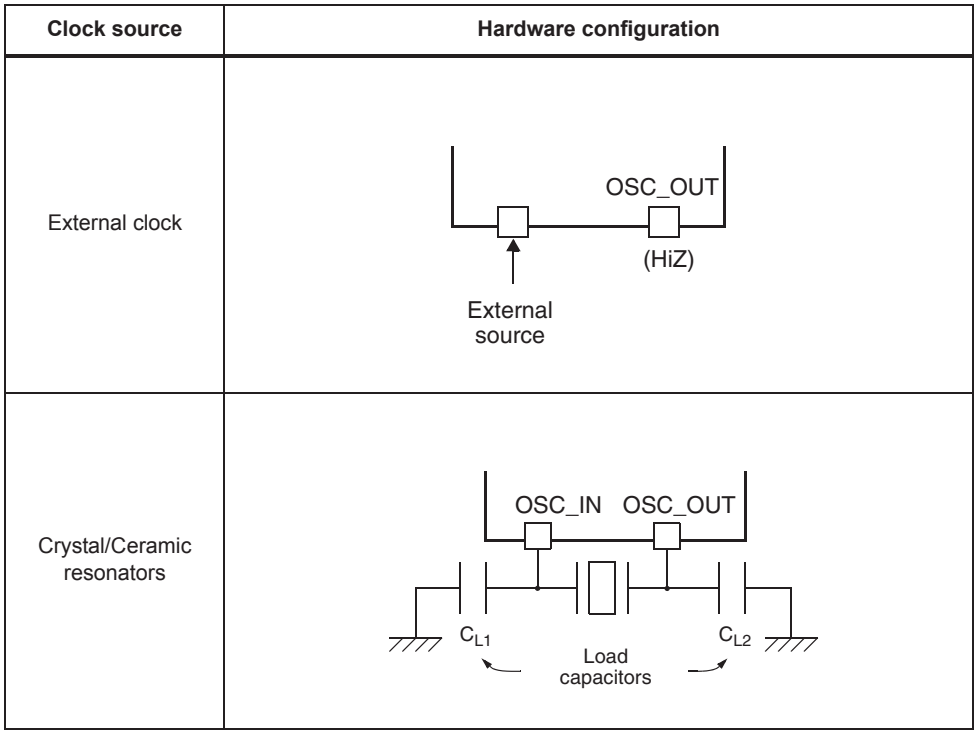
7.2.1. Такты HSE

Высокоскоростной внешний тактовый сигнал (HSE) можно получить из двух источников:

- внешний кварцевый/керамический резонатор HSE
- внешние такты HSE пользователя

Резонатор и конденсаторы нагрузки надо размещать как можно ближе к ножкам генератора для снижения помех и стабилизации времени запуска. Ёмкости нагрузки должны соответствовать выбранному генератору.

Рис. 22. Источники HSE/LSE



Внешний источник (HSE bypass)

В этом режиме внешний источник должен иметь частоту 50 MHz. Он включается битами `HSEBYP` и `HSEON` в регистре `RCC_CR`. Внешний сигнал (меандр, синус или треугольник) с ~50% заполнением подаётся на ножку OSC_IN, ножка OSC_OUT должна оставаться в третьем (hi-Z) состоянии.

Внешний кварцевый/керамический резонатор (HSE crystal)

Выгода внешнего генератора в очень точных главных тактах.

Стабильность внешнего генератора показывается флагом `HSERDY` в регистре `RCC_CR`. При включении такты не выпускаются на волю пока аппаратура не поставит этот бит. При этом может быть вызвано прерывание, если разрешено в регистре `RCC_CIR`.

HSE Crystal включается/выключается битом `HSEON` в регистре `RCC_CR`.

7.2.2. Такты HSI

Тактовый сигнал HSI от внутреннего 16 MHz RC генератора можно использовать как системные такты или подать на вход PLL (ФАПЧ).

Выгода HSI RC генератора в низкой стоимости. Да и запускается быстрее, чем HSE crystal генератор, но точность всё равно ниже кварцевого или керамического даже после калибровки.

Калибровка

Частоты RC генератора каждого чипа калибруются фирмой ST до 1% точности при TA=25°C.

После сброса, заводская калибровка загружается в биты `HSICAL[7:0]` регистра `RCC_CR`.

Если ваша система работает при колебаниях напряжения и температуры, то частота RC генератора может меняться. Подстроить частоту HSI можно битами `HSITRIM[4:0]` в регистре `RCC_CR`.

Флаг `HSIRDY` в регистре `RCC_CR` показывает стабильность HSI RC. При включении такты HSI RC не выпускаются на волю пока аппаратура не поставит этот бит.

HSI RC включается/выключается битом `HSION` в регистре `RCC_CR`.

Сигнал HSI можно использовать как резервный (Auxiliary clock) при сбое HSE. См. *Секцию 7.2.7: Система безопасности тактирования (CSS)*.

7.2.3. Конфигурация PLL (ФАПЧ)

В STM32F4xx есть два PLL:

- Главный PLL , тактируемый от HSE или HSI, с двумя выходами:
  - Высокочастотные системные такты (до 168 MHz)
  - Такты для USB OTG FS (48 MHz), RNG (≤48 MHz) и SDIO (≤48 MHz).
- Отдельный PLL (PLLI2S) для точных тактов интерфейса I2S.

Главный PLL надо конфигурировать до его разрешения.

PLLI2S использует тот же вход, что и PLL (у них общие биты PLLM[5:0] и PLLSRC). Но у PLLI2S отдельные биты включения/выключения и делители (N и R). Конфигурировать PLLI2S надо до включения.

Оба PLL выключаются при входе в режимы Stop и Standby или, при отказе HSE, когда HSE или PLL (с тактами от HSE) используются как системные такты. PLL и PLLI2S конфигурируются регистрами RCC_PLLCFGR и RCC_CFGR.

7.2.4. Такты LSE

LSE crystal это 32.768 kHz низкоскоростной внешний кварцевый или керамический резонатор. Его выгода в высокой точности сигнала для RTC и другой периферии.

Включается/выключается LSE битом LSEON в регистре резервного домена RCC_BDCR.

Флаг LSERDY в регистре RCC_BDCR показывает стабильность LSE crystal. При включении такты LSE crystal не выпускаются на волю пока аппаратура не поставит этот бит. При этом может быть вызвано прерывание, если разрешено в регистре RCC_CIR.

Внешний источник (LSE bypass)

Внешний источник частотой до 1 MHz выбирается установкой битов LSEBYP и LSEON в регистре RCC_BDCR. Внешний сигнал (меандр, синус или треугольник) с ~50% заполнением подаётся на ножку OSC32_IN, ножка OSC32_OUT должна оставаться в третьем (hi-Z) состоянии.

7.2.5. Такты LSI

Генератор LSI RC с частотой около 32 kHz продолжает работать в режимах Stop и Standby ради независимого сторожевого таймера (IWDG) и блока авто-побудки (AWU).

Включается/выключается LSI RC битом LSION в регистре RCC_CSR.

Флаг LSIRDY в регистре RCC_CSR показывает стабильность LSE crystal. При включении такты LSE crystal не выпускаются на волю пока аппаратура не поставит этот бит. При этом может быть вызвано прерывание, если разрешено в регистре RCC_CIR.

7.2.6. Выбор SysClk

После системного сброса такты системы берутся от HSI. Источник системных тактов, подключенный напрямую, или через PLL остановить нельзя.

Переключение на новый источник тактов возможно только если он готов (стабилен после запуска или PLL зафиксирован). Если выбран неготовый источник, то он подключится только после готовности. Биты состояния в регистре RCC_CR показывают готовые генераторы и кто выбран системным.

7.2.7. Система безопасности тактирования (CSS)

CSS активируется программно и включается после задержки запуска HSE, выключается после остановки генератора.

Если в тактах HSE обнаруживается сбой, то он автоматически отключается, таймеру TIM1 посылается событие отключения входа и генерируется прерывание (Clock Security System Interrupt CSSI), подключённое к вектору исключения NMI.

NB: Если при включённой CSS возникает сбой HSE, прерывание NMI происходит независимо от того, очищен ли бит удержания CSS. Следовательно в NMI ISR нужно очистить прерывание CSS установкой бита CSSC в регистре RCC_CIR.

Если генератор HSE служит источником системных тактов напрямую или через PLL/PLL2, то обнаруженный сбой переключает тактирование на HSI и выключает HSE и PLL (в случае его использования).

7.2.8. Такты RTC/AWU

Источником тактов RTCCLK могут быть HSE после предделителя, LSE или LSI. Он выбирается битами RTCSEL[1:0] в регистре RCC_BDCR и битами RTCPRE[4:0] регистра RCC_CFGR. Этот выбор нельзя изменить без сброса домена резервного хранения.

Если тактами RTC выбран LSE, то при выключении резервного или системного питания RTC работает нормально. Если тактами AWU выбран LSI, то при выключении системного питания

состояние AWU не гарантируется. Если тактами RTC выбран HSE, поделённый на число от 2 и 31, то то при выключении резервного или системного питания состояние RTC не гарантируется.

Генератор LSE расположен в резервном домене, а HSE и LSI нет. Следовательно:

- Если RTC тактируется от LSE:
  - RTC продолжает работать при отключении V_{DD}, питаясь от V_{BAT}.
- Если блок авто-побудки (AWU) тактируется от LSI:
  - Состояние AWU при отключении V_{DD} не гарантируется. См. *Секцию 7.2.5*.
- Если RTC тактируется от HSE:
  - При отключении V_{DD} или регулятора напряжения (выключения 1.2 V домена) состояние RTC не гарантируется.

7.2.9. Такты сторожевого таймера

При аппаратном или программном запуске независимого сторожевого таймера (IWDG) генератор LSI включается и не выключается. Тактирование на IWDG пропускается после задержки запуска LSI.

7.2.10. Вывод тактового сигнала

Существует возможность выводить на внешнюю ножку тактовый сигнал микроконтроллера (MCO). Регистры конфигурации соответствующего порты GPIO должны быть переключены в альтернативный режим. Можно выводить один из 8 сигналов.

- SYSCLK
- HSI
- HSE
- PLL/2
- PLL2
- PLL3/2
- внешний XT1 3-25 MHz генератор (для Ethernet)
- PLL3 (для Ethernet)

Он выбирается битами `MCO[3:0]` в регистре `RCC_CFGR`.

Есть две ножки вывода тактов наружу (MCO) с использованием предделителя (от 1 до 5):

- MCO1 (PA8)

  Битами `MCO1PRE`[2:0] и `MCO1`[1:0] в регистре `RCC_CFGR` можно вывести:
  - HSI
  - LSE
  - HSE
  - PLL
- MCO2 (PC9)

  Битами `MCO2PRE`[2:0] и `MCO2`[1:0] в регистре `RCC_CFGR` можно вывести:
  - HSE
  - PLL
  - SYSCLK
  - PLLI2S

Для ножек MCO соответствующий порт GPIO должен быть переведён в режим альтернативных функций. Часто тактов на ножках MCO не должна быть больше 100 MHz.

7.2.11. Измерение частоты тактов с помощью TIM5/TIM11

Частоту всех генераторов системы можно косвенно замерить с помощью захвата входа на TIM5 канал 4 и TIM11 канал 1. См. Рис. 18. и Рис. 19.

Измерение частоты тактов с помощью TIM5 канал 4

Входной мультиплексор TIM5 позволяет захватывать вход по сигналу I/O или по внутренним тактам. Это выбирается битами `TI4_RMP` [1:0] в регистре `TIM5_OR`.

LSE подключён к захвату входа канала 4 для точного измерения частоты HSI (когда он питает системные такты). Измеряется число тактов HSI между фронтами кварцевого LSE. Теперь можно вводить компенсацию отклонений по температуре и напряжению. Для этого у HSI есть биты калибровки.

А можно померить частоту сверх-маломощного, но не точного LSI в тактах HSI процедурой:

1. Включаем TIM5 и ставим канал 4 на захват входа.
2. Записью в биты `TI4_RMP` регистра `TIM5_OR` числа `0x01` подключаем такты LSI к захвату входа TIM5 канал 4.
3. Измеряем частоту тактов LSI по событию или прерыванию TIM5 захват/сравнение 4.
4. Обновляем предделитель RTC точности повышения для.

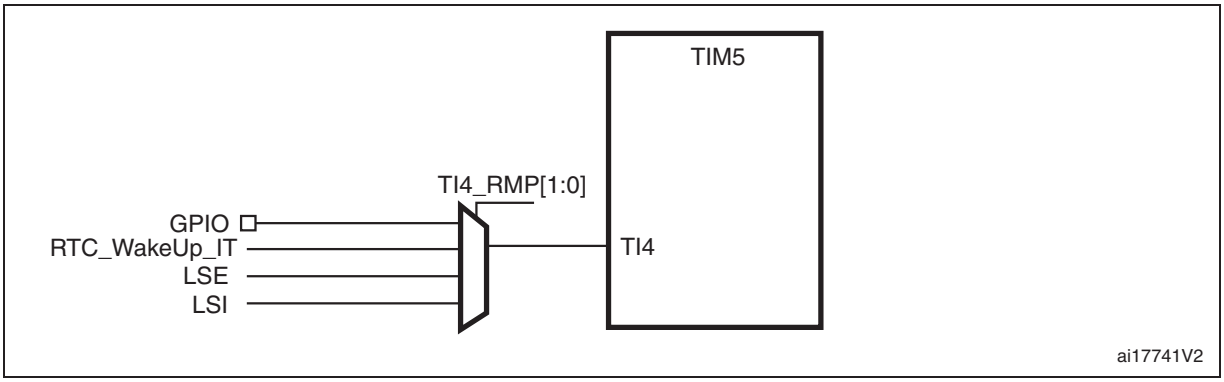


Рис. 23. Измерение частоты с помощью захвата входа TIM5

Измерение частоты тактов с помощью TIM11 канал 1

Входной мультиплексор TIM11 позволяет захватывать вход по сигналу I/O или по внутренним тактам. Это выбирается битами `TI1_RMP [1:0]` в регистре `TIM11_OR`.

Такты HSE_RTC (HSE/предделитель) подключаются к захвату входа канала 1 для грубого определения частоты внешнего кварца. Системные такты должно кормиться от HSI. Это полезно для соблюдения стандарта IEC 60730/IEC 61335, требующего возможности определения гармонических и субгармонических частот (с девиацией –50/+100%).

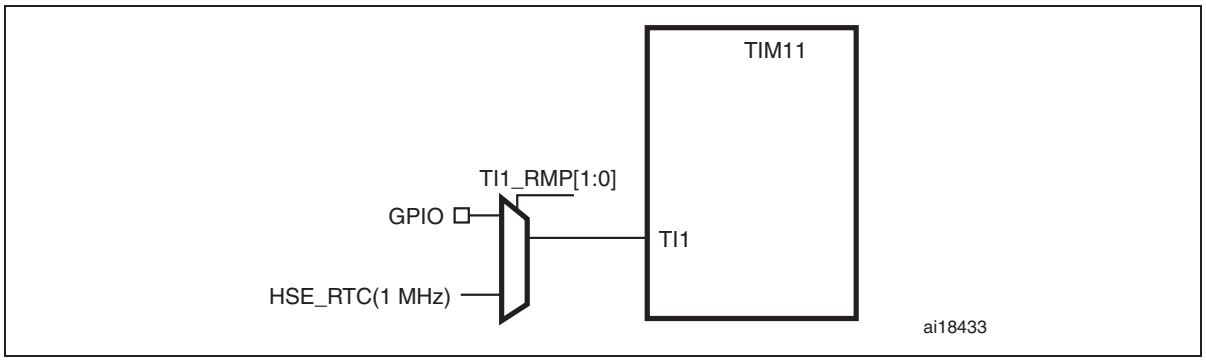


Рис. 24. Измерение частоты с помощью захвата входа TIM11

7.3. Регистры RCC

7.3.1. Регистр управления (RCC_CR)

Смещение адреса: 0x00

По сбросу: 0x0000 XX83 где X не определено.

Доступ: без ожидания, слово, полуслово и байт.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				PLLI2SRDY	PLLI2SON	PLLRDY	PLLON	Reserved				CSSON	HSEBYP	HSERDY	HSEON
				r	rw	r	rw					rw	rw	r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HSICAL[7:0]								HSITRIM[4:0]					Res.	HSIRDY	HSION
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw		r	rw

— Биты 31:28 Резерв, не трогать.

— Бит 27 **PLL2RDY**: Флаг фиксации PLL2.

Ставится аппаратно.

0: PLL2 unlocked

1: PLL2 locked

— Бит 26 **PLL2ON:** Включение PLL2.

Ставится и снимается программно. Аппаратно снимается при входе в режим Stop или Standby. Если PLL2 прямо или непрямо используется как системные такты, то снять его невозможно.

0: PLL2 OFF

1: PLL2 ON

— Бит 25 **PLLRDY**: Флаг фиксации PLL.

Ставится аппаратно.

0: PLL unlocked

1: PLL locked

— Бит 24 **PLLON:** Включение PLL.

Ставится и снимается программно. Аппаратно снимается при входе в режим Stop или Standby. Если PLL прямо или непрямо используется как системные такты, то снять его невозможно.

0: PLL OFF

1: PLL ON

— Биты 23:20 Резерв, не трогать.

— Бит 19 **CSSON:** Включение системы безопасности.

Управляется программно. При стоящем CSSON детектор тактов аппаратно включается при готовом HSE и выключается при отказе HSE.

0: Детектор OFF

1: Детектор ON (ON если HSE готов , OFF если нет).

— Бит 18 **HSEBYP:** Шунт (обход) **HSE**.

Управляется программно. Бит HSEON должен стоять. Бит HSEBYP можно писать только при выключенном HSE.

0: Внешний генератор 3-25 MHz не обойдён

1: Внешний генератор 3-25 MHz обойдён

— Бит 17 **HSERDY:** Флаг готовности HSE.

Ставится аппаратно при стабильном HSE. Снимается через 6 тактов HSE после снятия HSEON.

0: HSE не готов

1: HSE готов

— Бит 16 **HSEON:** Включение HSE.

Управляется программно. Снимается аппаратно при входе в режим Stop или Standby. Если HSE прямо или непрямо используется как источник системных тиков, то снять его невозможно.

0: HSE генератор OFF

1: HSE генератор ON

— Биты 15:8 **HSICAL[7:0]:** Калибровка HSI.

Заводские установки инициализируются при запуске.

— Биты 7:3 **HSITRIM[4:0]:** Подстройка HSI.

Программная подстройка частоты внутреннего HSI RC при колебаниях напряжения и температуры.

— Бит 2 Резерв, не трогать.

— Бит 1 **HSIRDY:** Флаг готовности HSI.

Ставится аппаратно при стабильности внутреннего генератора. Снимается через 6 тактов HSI после снятия HSI.

0: HSI не готов

1: HSI готов

— Бит 0 **HSION:** Включение HSI.

Управляется программно. Ставится аппаратно при выходе из режимов Stop или Standby и в случае отказа внешнего генератора, используемого для системных тактов. Этот бит невозможно снять при прямом или непрямом использовании для системных тактов или выборе стать таковым.

0: HSI OFF

1: HSI ON

7.3.2. Регистр конфигурации PLL (RCC_PLLCFR)

Смещение адреса: 0x04

По сбросу: 0x2400 3010.

Доступ: без ожидания, слово, полуслово или байт.

Конфигурация выходных тактов PLL по формулам:

- $f_{(VCO\ clock)} = f_{(PLL\ clock\ input)} \times (PLLN / PLLM)$
- $f_{(PLL\ general\ clock\ output)} = f_{(VCO\ clock)} / PLLP$
- $f_{(USB\ OTG\ FS, SDIO, RNG\ clock\ output)} = f_{(VCO\ clock)} / PLLQ$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				PLLQ3	PLLQ2	PLLQ1	PLLQ0	Reserved	PLLSRC	Reserved				PLLP1	PLLP0
				rw	rw	rw	rw		rw					rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	PLLN									PLLM5	PLLM4	PLLM3	PLLM2	PLLM1	PLLM0
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:28 Резерв, не трогать.

— Биты 27:24 **PLLQ:** Делитель тактов PLL для USB OTG FS, SDIO и RNG

Ставится и снимается программно при выключенном PLL.

Внимание: USB OTG FS требует правильных тактов 48MHz. SDIO и RNG требуют правильных тактов ≤48MHz.

Частота тактов USB OTG FS = Частота VCO / PLLQ с 2 ≤PLLQ ≤15

0000: PLLQ = 0, плохой выбор

0001: PLLQ = 1, плохой выбор

0010: PLLQ = 2

0011: PLLQ = 3

0100: PLLQ = 4

…

1111: PLLQ = 15

— Бит 23 Резерв, не трогать.

— Бит 22 **PLLSRC:** Источник тактов PLL и PLLI2S

Ставится и снимается программно при выключенных PLL и PLLI2S.

0: Такты от HSI

1: Такты от HSE

— Биты 21:18 Резерв, не трогать.

— Биты 17:16 **PLLP:** Делитель PLL для системных тактов

Ставится и снимается программно при выключенном PLL.

Внимание: Выходная частота PLL не должна превышать 168 MHz для этого домена.

Частота PLL = Частота VCO / PLLP с PLLP = 2, 4, 6, или 8

00: PLLP = 2

01: PLLP = 4

10: PLLP = 6

11: PLLP = 8

— Бит 15 Резерв, не трогать.

— Биты 14:6 **PLLN**: Множитель PLL для VCO

Ставится и снимается программно при выключенном PLL. Доступ словом и полусловом

Внимание: Выходная частота VCO должна быть между 100 и 432 MHz.

Выходные такты VCO = Входные такты VCO × PLLN с 50 ≤PLLN ≤432

000000000: PLLN = 0, плохой выбор

000000001: PLLN = 1, плохой выбор

...

000110010: PLLN = 50

…

001100011: PLLN = 99

001100100: PLLN = 100

...

110110000: PLLN = 432

110110001: PLLN = 433, плохой выбор

…

111111111: PLLN = 511, плохой выбор

NB: Множитель от 50 и 99 возможен при входной частоте VCO больше 1 MHz.

— Биты 5:0 **PLLM**: Делитель входной частоты для PLL и PLLI2S перед VCO

Ставится и снимается программно при выключенных PLL и PLLI2S.

Внимание: Входная частота VCO должна быть между 1 и 2 MHz. Рекомендуется 2 MHz для уменьшения дребезга PLL.

Входные такты VCO = Входные такты PLL × PLLM с 2 ≤PLLM ≤63

000000: PLLM = 0, wrong configuration

000001: PLLM = 1, wrong configuration

000010: PLLM = 2

000011: PLLM = 3

000100: PLLM = 4

…

111110: PLLM = 62

111111: PLLM = 63

7.3.3. Регистр конфигурации (RCC_CFGR)

Смещение адреса: 0x08

По сбросу: 0x0000 0000.

Доступ: 0 ≤ такты ожидания ≤ 2, слово, полуслово или байт. 1 или 2 тактов ожидания появляются только при переключении источника тактов.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCO2		MCO2 PRE[2:0]			MCO1 PRE[2:0]			I2SSCR	MCO1		RTCPRE[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPRE2[2:0]			PPRE1[2:0]			Reserved		HPRE[3:0]				SWS1	SWS0	SW1	SW0
rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	r	r	rw	rw

— Биты 31:30 **MCO2**: Выход 2 тактов MCU

Ставится и снимается программно. Переключение источника может выдать глитчи на MCO2, так что надо ставить биты после сброса и перед включением внешних генераторов и PLL.

00: System clock (SYSCLK)

01: PLLI2S clock

10: HSE oscillator clock

11: PLL clock

— Биты 29:27 **MCO2 PRE[2:0]**: Предделитель MCO2

Ставится и снимается программно. Переключение предделителя может выдать глитчи на MCO2, так что надо ставить биты после сброса и перед включением внешних генераторов и PLL.

0xx: без деления

100: / 2

101: / 3

110: / 4

111: / 5

— Биты 26:24 **MCO1 PRE[2:0]**: Предделитель MCO1

Ставится и снимается программно. Переключение предделителя может выдать глитчи на MCO1, так что надо ставить биты после сброса и перед включением внешних генераторов и PLL.

0xx: без деления

100: / 2
101: / 3
110: / 4
111: / 5

— Бит 23 **I2SSRC**: Выбор источника тактов I2S

Ставится и снимается программно. Рекомендуется ставить биты после сброса и перед включением модуля I2S.

0: Такты от PLLI2S
1: Внешние такты, приведённые на ножку I2S_CKIN.

— Биты 22:21 **MCO1**: Выход 1 тактов MCU

Ставится и снимается программно. Переключение источника может выдать глитчи на MCO2, так что надо ставить биты после сброса и перед включением внешних генераторов и PLL.

00: HSI
01: LSE
10: HSE
11: PLL

— Биты 20:16 **RTCPRE[4:0]:** Предделитель HSE для получения тактов RTC 1 MHz

Ставится и снимается программно. Рекомендуется ставить биты до выбора источника тактов с RTC.

00000: тактов нет
00001: тактов нет
00010: HSE/2
00011: HSE/3
00100: HSE/4
...
11110: HSE/30
11111: HSE/31.

— Биты 15:13 **PPRE2**: Высокоскоростной предделитель APB2.

Пишется программно.

Внимание: Частота не должна превышать 84 MHz. Новое значение вступает в действие после от 1 до 16 тактов AHB.

0xx: AHB как есть
100: AHB / 2
101: AHB / 4
110: AHB / 8
111: AHB / 16

— Биты 12:10 **PPRE1**: Низкоскоростной предделитель APB1.

Пишется программно.

Внимание: Частота не должна превышать 42 MHz. Новое значение вступает в действие после от 1 до 19 тактов AHB.

0xx: AHB как есть
100: AHB / 2
101: AHB / 4
110: AHB / 8
111: AHB / 16

— Биты 9:8 Резерв, не трогать.

— Биты 7:4 **HPRE[3:0]**: Предделитель AHB.

Пишется программно.

Внимание: При использовании Ethernet частота должна быть не ниже 25 MHz. Новое значение вступает в действие после от 1 до 16 тактов AHB.

0xxx: SYSCLK как есть
1000: SYSCLK / 2
1001: SYSCLK / 4
1010: SYSCLK / 8
1011: SYSCLK / 16
1100: SYSCLK / 64
1101: SYSCLK / 128
1110: SYSCLK / 256

1111: SYSCLK / 512

— Биты 3:2 **SWS**: Состояние переключателя системных тактов.

Ставится аппаратно.

00: HSI

01: HSE

10: PLL

11: нету такого

— Биты 1:0 **SW**: Переключатель системных тактов.

Программно ставится для выбора источника SYSCLK.

Аппаратно переключается на HSI при выходе из режима Stop и Standby или при отказе генератора HSE при включённой системе контроля тактирования CSS.

00: HSI

01: HSE

10: PLL

11: не дозволено.

7.3.4. Регистр прерываний (RCC_CIR)

Смещение адреса: 0x0C

По сбросу: 0x0000 0000.

Доступ: без ожидания, слово, полуслово и байт.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								CSSC	Reserved	PLLI2SRDYC	PLLRDYC	HSERDYC	HSIRDYC	LSERDYC	LSIRDYC
								w		w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		PLLI2SRDYIE	PLLRDYIE	HSERDYIE	HSIRDYIE	LSERDYIE	LSIRDYIE	CSSF	Reserved	PLLI2SRDYF	PLLRDYF	HSERDYF	HSIRDYF	LSERDYF	LSIRDYF
		rw	rw	rw	rw	rw	rw	r		r	r	r	r	r	r

Очистка флагов и готовности. Пишется программно. 0: ничего,1: чистит

Разрешение прерывания. 0: Запрещено 1: Разрешено.

— Биты 31:24 Резерв, не трогать.

— Бит 23 **CSSC**: Очистка флага прерывания CSS.

Пишется программно. 0: ничего,1: чистит

— Бит 22 Резерв, не трогать.

— Бит 21 **PLLI2SRDYC:** Очистка флага готовности PLLI2S.

— Бит 20 **PLLRDYC:** Очистка флага готовности PLL.

— Бит 19 **HSERDYC:** Очистка флага готовности HSE.

— Бит 18 **HSIRDYC:** Очистка флага готовности HSI.

— Бит 17 **LSERDYC:** Очистка флага готовности LSE.

— Бит 16 **LSIRDYC:** Очистка флага готовности LSI.

— Биты 15:14 Резерв, не трогать.

— Бит 14 **PLLSAIRDYIE:** Разрешение прерывания готовности PLLSAI.

— Бит 13 **PLLI2SRDYIE:** Разрешение прерывания готовности PLLI2S.

— Бит 12 **PLLRDYIE:** Разрешение прерывания готовности PLL.

— Бит 11 **HSERDYIE:** Разрешение прерывания готовности HSE.

— Бит 10 **HSIRDYIE:** Разрешение прерывания готовности HSI.

— Бит 9 **LSERDYIE:** Разрешение прерывания готовности LSE.

— Бит 8 **LSIRDYIE:** Разрешение прерывания готовности LSI.

— Бит 7 **CSSF:** Флаг CSS.

Ставится аппаратно при отказе HSE. Снимается записью бита CSSC.

0: Отказа нет 1: Отказ есть

— Бит 6 Резерв, не трогать.

— Бит 5 **PLLI2SRDYF:** Флаг прерывания готовности PLLI2S.

Ставится аппаратно при фиксации PLL и стоящем PLLI2SRDYIE. Чистится записью бита PLLI2SRDYC.

0: Прерывания нет 1: Прерывание есть

— Бит 4 **PLLRDYF:** Флаг прерывания готовности PLL.

Ставится аппаратно при фиксации PLL и стоящем PLLRDYDIE. Чистится записью бита PLLRDYC.
0: Прерывания нет 1: Прерывание есть

— Бит 3 **HSERDYF:** Флаг прерывания готовности HSE.
Ставится аппаратно при готовности HSE и стоящем HSERDYDIE. Чистится записью бита HSERDYC.
0: Прерывания нет 1: Прерывание есть

— Бит 2 **HSIRDYF:** Флаг прерывания готовности HSI.
Ставится аппаратно при готовности HSI и стоящем HSIRDYDIE. Чистится записью бита HSIRDYC.
0: Прерывания нет 1: Прерывание есть

— Бит 1 **LSERDYF:** Флаг прерывания готовности LSE.
Ставится аппаратно при готовности LSE и стоящем LSERDYDIE. Чистится записью бита LSERDYC.
0: Прерывания нет 1: Прерывание есть

— Бит 0 **LSIRDYF:** Флаг прерывания готовности LSI.
Ставится аппаратно при готовности LSI и стоящем LSIRDYDIE. Чистится записью бита LSIRDYC.
0: Прерывания нет 1: Прерывание есть

7.3.5. Регистр сброса периферии AHB1 (RCC_AHB1RSTR)

Смещение адреса: 0x10

По сбросу: 0x0000 0000.

Доступ: без ожидания, слово, полуслово и байт. Сброс записью "1".

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		OTGHS RST	Reserved			ETHMAC RST	Reserved		DMA2 RST	DMA1 RST	Reserved				
		rw				rw			rw	rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CRCRST	Reserved			GPIOIRST	GPIOHRST	GPIOGGRST	GPIOFRST	GPIOERST	GPIODRST	GPIOCRST	GPIOBRST	GPIOARST
			rw				rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:30 Резерв, не трогать.
— Бит 29 **OTGHSRST**: Сброс OTG HS.
— Биты 28:26 Резерв, не трогать.
— Бит 25 **ETHMACRST**: Сброс Ethernet MAC.
— Биты 24:23 Резерв, не трогать.
— Бит 22 **DMA2RST**: Сброс DMA2.
— Бит 21 **DMA1RST**: Сброс DMA1.
— Биты 20:16 Резерв, не трогать.
— Бит 12 **CRCRST:** Сброс CRC.
— Биты 11:9 Резерв, не трогать.
— Бит 8 **GPIOIRST:** Сброс GPIOI.
— Бит 7 **GPIOHRST:** Сброс GPIOH.
— Бит 6 **GPIOGRST:** Сброс GPIOG.
— Бит 5 **GPIOFRST:** Сброс GPIOF.
— Бит 4 **GPIOERST:** Сброс GPIOE.
— Бит 3 **GPIODRST:** Сброс IGPIOD.
— Бит 2 **GPIOCRST:** Сброс GPIOC.
— Бит 1 **GPIOBRST:** Сброс GPIOB.
— Бит 0 **GPIOARST:** Сброс GPIOA.

7.3.6. Регистр сброса периферии AHB2 (RCC_AHB2RSTR)

Смещение адреса: 0x14

По сбросу: 0x0000 0000.

Доступ: без ожидания, слово, полуслово и байт. Сброс записью "1" в бит.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								OTGFS RST	RNG RST	HASH RST	CRYP RST	Reserved			DCMI RST
								rw	rw	rw	rw				rw

— Биты 31:8 Резерв, не трогать.
— Бит 7 **OTGFSRST:** Сброс OTG FS.
— Бит 6 **RNGRST:** Сброс RNG.
— Бит 5 **HASHRST:** Сброс HASH.
— Бит 4 **CRYPRST:** Сброс CRYP.
— Биты 3:1 Резерв, не трогать.
— Бит 0 **DCMIRST:** Сброс DCMI.

7.3.7. Регистр сброса периферии AHB3 (RCC_AHB3RSTR)

Смещение адреса: 0x18

По сбросу: 0x0000 0000.

Доступ: без ожидания, слово, полуслово и байт. Сброс записью "1" в бит.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															FSMCRST
															rw

— Бит 0 **FSMCRST:** Сброс FSMC.

7.3.8. Регистр сброса периферии APB1 (RCC_APB1RSTR)

Смещение адреса: 0x20

По сбросу: 0x0000 0000.

Доступ: без ожидания, слово, полуслово и байт. Сброс записью "1".

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		DACRST	PWR RST	Reser-ved	CAN2 RST	CAN1 RST	Reser-ved	I2C3 RST	I2C2 RST	I2C1 RST	UART5 RST	UART4 RST	UART3 RST	UART2 RST	Reser-ved
		rw	rw			rw		rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 RST	SPI2 RST	Reserved		WWDG RST	Reserved		TIM14 RST	TIM13 RST	TIM12 RST	TIM7 RST	TIM6 RST	TIM5 RST	TIM4 RST	TIM3 RST	TIM2 RST
rw	rw			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:30 Резерв, не трогать.
— Бит 29 **DACRST**: Сброс DAC.
— Бит 28 **PWRRST**: Сброс PWR.
— Бит 27 Резерв, не трогать.
— Бит 26 **CAN2RST**: Сброс CAN2.
— Бит 25 **CAN1RST**: Сброс CAN1.
— Бит 24 Резерв, не трогать.
— Бит 23 **I2C3RST**: Сброс I2C3.
— Бит 22 **I2C2RST**: Сброс I2C2.
— Бит 21 **I2C1RST**: Сброс I2C1.
— Бит 20 **UART5RST**: Сброс UART5.
— Бит 19 **UART4RST**: Сброс UART4.
— Бит 18 **UART3RST**: Сброс UART3.
— Бит 17 **UART2RST**: Сброс UART2.
— Бит 16 Резерв, не трогать.
— Бит 15 **SPI3RST**: Сброс SPI3.
— Бит 14 **SPI2RST**: Сброс SPI2.

— Биты 13:12 Резерв, не трогать.
— Бит 11 **WWDGRST:** Сброс WWDG.
— Биты 10:9 Резерв, не трогать.
— Бит 8 **TIM14RST:** Сброс TIM14.
— Бит 7 **TIM13RST:** Сброс TIM13.
— Бит 6 **TIM12RST:** Сброс TIM12.
— Бит 5 **TIM7RST:** Сброс TIM7.
— Бит 4 **TIM6RST:** Сброс TIM6.
— Бит 3 **TIM5RST:** Сброс TIM5.
— Бит 2 **TIM4RST:** Сброс TIM4.
— Бит 1 **TIM3RST:** Сброс TIM3.
— Бит 0 **TIM2RST:** Сброс TIM2.

7.3.9. Регистр сброса периферии APB2 (RCC_APB2RSTR)

Смещение адреса: 0x24

По сбросу: 0x0000 0000.

Доступ: без ожидания, слово, полуслово и байт. Сброс записью "1".

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													TIM11 RST	TIM10 RST	TIM9 RST
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reser-ved	SYSCFG RST	Reser-ved	SPI1 RST	SDIO RST	Reserved		ADC RST	Reserved		USART 6 RST	USART 1 RST	Reserved		TIM8 RST	TIM1 RST
	rw		rw	rw			rw			rw	rw			rw	rw

— Биты 31:19 Резерв, не трогать.
— Бит 18 **TIM11RST**: Сброс TIM11.
— Бит 17 **TIM10RST**: Сброс TIM10.
— Бит 16 **TIM9RST**: Сброс TIM9.
— Бит 15 Резерв, не трогать.
— Бит 14 **SYSCFGRST**: Сброс SYSCFG.
— Бит 13 **SPI4RST**: Сброс SPI4.
— Бит 12 **SPI1RST**: Сброс SPI1.
— Бит 11 **SDIORST**: Сброс SDIO.
— Биты 10:9 Резерв, не трогать.
— Бит 8 **ADCRST**: Сброс ADC.
— Биты 7:6 Резерв, не трогать.
— Бит 5 **USART6RST**: Сброс USART6.
— Бит 4 **USART1RST**: Сброс USART1.
— Биты 3:2 Резерв, не трогать.
— Бит 1 **TIM8RST:** Сброс TIM8.
— Бит 0 **TIM1RST:** Сброс TIM1.

7.3.10. Регистр разрешения тактов периферии AHB1 (RCC_AHB1ENR)

Смещение адреса: 0x30

По сбросу: 0x0010 0000.

Доступ: без ожидания, слово, полуслово и байт.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reser-ved	OTGHS ULPIEN	OTGHSEN	ETHMACPTPEN	ETHMACRXEN	ETHMACTXEN	ETHMACEN	Reserved		DMA2EN	DMA1EN	CCMDATARAMEN	Res.	BKPSRAMEN	Reserved	
	rw	rw	rw	rw	rw	rw			rw	rw			rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CRCEN	Reserved			GPIOIEN	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
			rw				rw	rw	rw	rw	rw	rw	rw	rw	rw

Читается и пишется программно.

0: Выключено

1: Включено

— Бит 31 Резерв, не трогать.
— Бит 30 **OTGHSULPIEN**: Тактирование OTG HSULPI.
— Бит 29 **OTGHSEN**: Тактирование OTG HS.
— Бит 28 **ETHMACPTPEN**: Тактирование Ethernet PTP.
— Бит 27 **ETHMACRXEN**: Тактирование Ethernet RX.
— Бит 26 **ETHMACTXEN**: Тактирование Ethernet TX.
— Бит 25 **ETHMACEN**: Тактирование Ethernet MAC.
— Бит 24:23 Резерв, не трогать.
— Бит 22 **DMA2EN**: Тактирование DMA2.
— Бит 21 **DMA1EN**: Тактирование DMA1.
— Бит 20 **CCMDATARAMEN**: Тактирование CCM RAM.
— Бит 19 Резерв, не трогать.
— Бит 18 **BKPSRAMEN**: Тактирование Резервной SRAM.
— Бит 17:13 Резерв, не трогать.
— Бит 12 **CCREN**: Тактирование CRC.
— Бит 11:9 Резерв, не трогать.
— Бит 8 **GPIOIEN**: Тактирование GPIOI.
— Бит 7 **GPIOHEN**: Тактирование GPIOH.
— Бит 6 **GPIOGEN**: Тактирование GPIOG.
— Бит 5 **GPIOFEN**: Тактирование GPIOF.
— Бит 4 **GPIOEEN**: Тактирование GPIOE.
— Бит 3 **GPIODEN**: Тактирование GPIOD.
— Бит 2 **GPIOCEN**: Тактирование GPIOC.
— Бит 1 **GPIOBEN**: Тактирование GPIOB.
— Бит 0 **GPIOAEN**: Тактирование GPIOA.

7.3.11. Регистр разрешения тактов периферии AHB2 (RCC_AHB2ENR)

Смещение адреса: 0x34

По сбросу: 0x0000 0000.

Доступ: без ожидания, слово, полуслово и байт.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								OTGFS EN	RNG EN	HASH EN	CRYP EN	Reserved			DCMI EN
								rw	rw	rw	rw				rw

Читается и пишется программно.

0: Выключено

1: Включено

— Биты 31:8 Резерв, не трогать.
— Бит 7 **OTGFSEN:** Тактирование OTG FS.
— Бит 6 **RNGEN:** Тактирование RNG.
— Бит 5 **HASHEN:** Тактирование HASH.
— Бит 4 **CRYPEN:** Тактирование CRYP.
— Биты 3:1 Резерв, не трогать.
— Бит 0 **DCMIEN:** Тактирование DCMI.

7.3.12. Регистр разрешения тактов периферии AHB3 (RCC_AHB3ENR)

Смещение адреса: 0x38

По сбросу: 0x0000 0000.

Доступ: без ожидания, слово, полуслово и байт.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															FSMCEN
															rw

Читается и пишется программно.

0: Выключено

1: Включено

— Биты 31:1 Резерв, не трогать.

— Бит 0 **FSMCEN:** Тактирование FSMC.

7.3.13. Регистр разрешения тактов периферии APB1 (RCC_APB1ENR)

Смещение адреса: 0x40

По сбросу: 0x0000 0000.

Доступ: Без ожиданий, слово, полуслово и байт.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		DAC EN	PWR EN	Reser-ved	CAN2 EN	CAN1 EN	Reser-ved	I2C3 EN	I2C2 EN	I2C1 EN	UART5 EN	UART4 EN	USART 3 EN	USART 2 EN	Reser-ved
		rw	rw		rw	rw		rw	rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 EN	SPI2 EN	Reserved		WWDG EN	Reserved		TIM14 EN	TIM13 EN	TIM12 EN	TIM7 EN	TIM6 EN	TIM5 EN	TIM4 EN	TIM3 EN	TIM2 EN
rw	rw			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

Читается и пишется программно.

0: Выключено

1: Включено

— Бит 31:30 Резерв, не трогать.

— Бит 29 **DACEN**: Тактирование DAC.

— Бит 28 **PWREN**: Тактирование PWR.

— Бит 27 Резерв, не трогать.

— Бит 26 **CAN2EN**: Тактирование CAN2.

— Бит 25 **CAN1EN:** Тактирование CAN1.

— Бит 24 Резерв, не трогать.

— Бит 23 **I2C3EN:** Тактирование I2C3.

— Бит 22 **I2C2EN:** Тактирование I2C2.

— Бит 21 **I2C1EN:** Тактирование I2C1.

— Бит 20 **UART5EN:** Тактирование UART5.

— Бит 19 **UART4EN:** Тактирование UART4.

— Бит 18 **USART3EN:** Тактирование USART3.

— Бит 17 **USART2EN:** Тактирование USART2.

— Бит 16 Резерв, не трогать.

— Бит 15 **SPI3EN:** Тактирование SPI3.

— Бит 14 **SPI2EN:** Тактирование SPI2.

— Биты 13:12 Резерв, не трогать.

— Бит 11 **WWDGEN:** Тактирование WWDG.

— Биты 10:9 Резерв, не трогать.

— Бит 8 **TIM14EN:** Тактирование TIM14.

— Бит 7 **TIM13EN:** Тактирование TIM13.

— Бит 6 **TIM12EN:** Тактирование TIM12.

— Бит 5 **TIM7EN:** Тактирование TIM7.

— Бит 4 **TIM6EN:** Тактирование TIM6.

— Бит 3 **TIM5EN:** Тактирование TIM5.

— Бит 2 **TIM4EN:** Тактирование TIM4.

— Бит 1 **TIM3EN:** Тактирование TIM3.
— Бит 0 **TIM2EN:** Тактирование TIM2.

7.3.14. Регистр разрешения тактов периферии APB2 (RCC_APB2ENR)

Смещение адреса: 0x44

По сбросу: 0x0000 0000.

Доступ: Без ожиданий, слово, полуслово и байт.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													TIM11 EN	TIM10 EN	TIM9 EN
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reser-ved	SYSCFG EN	Reser-ved	SPI1 EN	SDIO EN	ADC3 EN	ADC2 EN	ADC1 EN	Reserved		USART 6 EN	USART 1 EN	Reserved		TIM8 EN	TIM1 EN
	rw		rw	rw	rw	rw	rw			rw	rw			rw	rw

Читается и пишется программно.
0: Выключено
1: Включено

— Биты 31:19 Резерв, не трогать.
— Бит 18 **TIM11EN:** Тактирование TIM11.
— Бит 17 **TIM10EN:** Тактирование TIM10.
— Бит 16 **TIM9EN:** Тактирование TIM9.
— Бит 15 Резерв, не трогать.
— Бит 14 **SYSCFGEN:** Тактирование SYSCFG.
— Бит 13 Резерв, не трогать.
— Бит 12 **SPI1EN:** Тактирование SPI1.
— Бит 11 **SDIOEN:** Тактирование SDIO.
— Бит 10 **ADC3EN:** Тактирование ADC3.
— Бит 9 **ADC2EN:** Тактирование ADC2.
— Бит 8 **ADC1EN:** Тактирование ADC1.
— Биты 7:6 Резерв, не трогать.
— Бит 5 **USART6EN:** Тактирование USART6.
— Бит 4 **USART1EN:** Тактирование USART1.
— Биты 3:2 Резерв, не трогать.
— Бит 1 **TIM8EN:** Тактирование TIM8.
— Бит 0 **TIM1EN:** Тактирование TIM1.

7.3.15. Регистр разрешения тактов периферии AHB1 в экономном режиме (RCC_AHB1LPENR)

Смещение адреса: 0x50

По сбросу: 0x7E67 97FF.

Доступ: без ожидания, слово, полуслово и байт.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reser-ved	OTGHS ULPILPEN	OTGHS LPEN	ETHPTP LPEN	ETHRX LPEN	ETHTX LPEN	ETHMAC LPEN	Reserved		DMA2 LPEN	DMA1 LPEN	Reserved		BKPSRAM LPEN	SRAM2 LPEN	SRAM1 LPEN
	rw	rw	rw	rw	rw	rw			rw	rw			rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLITF LPEN	Reserved		CRC LPEN	Reserved			GPIOI LPEN	GPIOH LPEN	GPIOG LPEN	GPIOF LPEN	GPIOE LPEN	GPIOD LPEN	GPIOC LPEN	GPIOB LPEN	GPIOA LPEN
rw			rw				rw	rw	rw	rw	rw	rw	rw	rw	rw

Читается и пишется программно.
0: Выключено
1: Включено

— Бит 31 Резерв, не трогать.
— Бит 30 **OTGHSULPIEN**: Тактирование OTG HSULPI.
— Бит 29 **OTGHSLPEN**: Тактирование OTG HS.
— Бит 28 **ETHMACPTPLPEN**: Тактирование Ethernet PTP.
— Бит 27 **ETHMACRXLPEN**: Тактирование Ethernet RX.
— Бит 26 **ETHMACTXLPEN**: Тактирование Ethernet TX.
— Бит 25 **ETHMACLPEN**: Тактирование Ethernet MAC.
— Биты 24:23 Резерв, не трогать.
— Бит 22 **DMA2LPEN**: Тактирование DMA2.
— Бит 21 **DMA1LPEN**: Тактирование DMA1.
— Бит 20:19 Резерв, не трогать.
— Бит 18 **BKPSRAMLPEN**: Тактирование Резервной SRAM.
— Бит 17 **SRAM2LPEN**: Тактирование SRAM2.
— Бит 16 **SRAM1LPEN**: Тактирование SRAM1.
— Бит 15 **FLITFLPEN**: Тактирование FLITF.
— Бит 14:13 Резерв, не трогать.
— Бит 12 **CCRLPEN**: Тактирование CRC.
— Биты 11:9 Резерв, не трогать.
— Бит 8 **GPIOILPEN**: Тактирование GPIOI.
— Бит 7 **GPIOHLPEN**: Тактирование GPIOH.
— Бит 6 **GPIOGLPEN**: Тактирование GPIOG.
— Бит 5 **GPIOFLPEN**: Тактирование GPIOF.
— Бит 4 **GPIOELPEN**: Тактирование GPIOE.
— Бит 3 **GPIODLPEN**: Тактирование GPIOD.
— Бит 2 **GPIOCLPEN**: Тактирование GPIOC.
— Бит 1 **GPIOBLPEN**: Тактирование GPIOB.
— Бит 0 **GPIOALPEN**: Тактирование GPIOA.

7.3.16. Регистр разрешения тактов периферии AHB2 в экономном режиме (RCC_AHB2LPENR)

Смещение адреса: 0x54

По сбросу: 0x0000 00F1.

Доступ: без ожидания, слово, полуслово и байт.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								OTGFS LPEN	RNG LPEN	HASH LPEN	CRYP LPEN	Reserved			DCMI LPEN
								rw	rw	rw	rw				rw

Читается и пишется программно.
0: Выключено
1: Включено

— Биты 31:8 Резерв, не трогать.
— Бит 7 **OTGFSLPEN:** Тактирование OTG FS.
— Бит 6 **RNGLPEN:** Тактирование RNG.
— Бит 5 **HASHLPEN:** Тактирование HASH.
— Бит 4 **CRYPLPEN:** Тактирование CRYP.
— Биты 3:1 Резерв, не трогать.
— Бит 0 **DCMILPEN:** Тактирование DCMI.

7.3.17. Регистр разрешения тактов периферии AHB3 в экономном режиме (RCC_AHB3LPENR)

Смещение адреса: 0x58

По сбросу: 0x0000 0001.

Доступ: без ожидания, слово, полуслово и байт.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															FSMC LPEN
															rw

Читается и пишется программно.

0: Выключено

1: Включено

— Биты 31:1 Резерв, не трогать.

— Бит 0 **FSMCLPEN:** Тактирование FSMC.

7.3.18. Регистр разрешения тактов периферии APB1 в экономном режиме (RCC_APB1LPENR)

Смещение адреса: 0x60

По сбросу: 0x36FE C9FF.

Доступ: Без ожиданий, слово, полуслово и байт.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		DAC LPEN	PWR LPEN	RESERVED	CAN2 LPEN	CAN1 LPEN	Reserved	I2C3 LPEN	I2C2 LPEN	I2C1 LPEN	UART5 LPEN	UART4 LPEN	USART3 LPEN	USART2 LPEN	Reserved
		rw	rw		rw	rw		rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 LPEN	SPI2 LPEN	Reserved		WWDG LPEN	Reserved		TIM14 LPEN	TIM13 LPEN	TIM12 LPEN	TIM7 LPEN	TIM6 LPEN	TIM5 LPEN	TIM4 LPEN	TIM3 LPEN	TIM2 LPEN
rw	rw			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

Читается и пишется программно.

0: Выключено

1: Включено

— Биты 31:30 Резерв, не трогать.

— Бит 29 **DACLPEN**: Тактирование DAC.

— Бит 28 **PWRLPEN**: Тактирование PWR.

— Бит 27 Резерв, не трогать.

— Бит 26 **CAN2LPEN**: Тактирование CAN2.

— Бит 25 **CAN1LPEN:** Тактирование CAN1.

— Бит 24 Резерв, не трогать.

— Бит 23 **I2C3LPEN:** Тактирование I2C3.

— Бит 22 **I2C2LPEN:** Тактирование I2C2.

— Бит 21 **I2C1LPEN:** Тактирование I2C1.

— Бит 20 **UART5LPEN:** Тактирование UART5.

— Бит 19 **UART4LPEN:** Тактирование UART4.

— Бит 18 **USART3LPEN:** Тактирование USART3.

— Бит 17 **USART2LPEN:** Тактирование USART2.

— Бит 16 Резерв, не трогать.

— Бит 15 **SPI3LPEN:** Тактирование SPI3.

— Бит 14 **SPI2LPEN:** Тактирование SPI2.

— Биты 13:12 Резерв, не трогать.

— Бит 11 **WWDGLPEN:** Тактирование WWDG.

— Биты 10:9 Резерв, не трогать.

— Бит 8 **TIM14LPEN:** Тактирование TIM14.

— Бит 7 **TIM13LPEN:** Тактирование TIM13.

— Бит 6 **TIM12LPEN:** Тактирование TIM12.

— Бит 5 **TIM7LPEN:** Тактирование TIM7.

— Бит 4 **TIM6LPEN:** Тактирование TIM6.

— Бит 3 **TIM5LPEN:** Тактирование TIM5.
— Бит 2 **TIM4LPEN:** Тактирование TIM4.
— Бит 1 **TIM3LPEN:** Тактирование TIM3.
— Бит 0 **TIM2LPEN:** Тактирование TIM2.

7.3.19. Регистр разрешения тактов периферии APB2 в экономном режиме (RCC_APBLP2ENR)

Смещение адреса: 0x64

По сбросу: 0x0407 5F33.

Доступ: Без ожиданий, слово, полуслово и байт.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													TIM11 LPEN	TIM10 LPEN	TIM9 LPEN
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reser-ved	SYSCFG LPEN	Reser-ved	SPI1 LPEN	SDIO LPEN	ADC3 LPEN	ADC2 LPEN	ADC1 LPEN	Reserved		USART 6 LPEN	USART 1 LPEN	Reserved		TIM8 LPEN	TIM1 LPEN
	rw		rw	rw	rw	rw	rw			rw	rw			rw	rw

Читается и пишется программно.
0: Выключено
1: Включено

— Биты 31:19 Резерв, не трогать.
— Бит 18 **TIM11LPEN:** Тактирование TIM11.
— Бит 17 **TIM10LPEN:** Тактирование TIM10.
— Бит 16 **TIM9LPEN:** Тактирование TIM9.
— Бит 15 Резерв, не трогать.
— Бит 14 **SYSCFGLPEN:** Тактирование SYSCFG.
— Бит 13 Резерв, не трогать.
— Бит 12 **SPI1LPEN:** Тактирование SPI1.
— Бит 11 **SDIOLPEN:** Тактирование SDIO.
— Бит 10 **ADC3LPEN:** Тактирование ADC3.
— Бит 9 **ADC2LPEN:** Тактирование ADC2.
— Бит 8 **ADC1LPEN:** Тактирование ADC1.
— Биты 7:6 Резерв, не трогать.
— Бит 5 **USART6LPEN:** Тактирование USART6.
— Бит 4 **USART1LPEN:** Тактирование USART1.
— Биты 3:2 Резерв, не трогать.
— Бит 1 **TIM8LPEN:** Тактирование TIM8.
— Бит 0 **TIM1LPEN:** Тактирование TIM1.

7.3.20. Регистр управления резервным доменом (RCC_BDCR)

Смещение адреса: 0x70

По сбросу Резервного Домена: 0x0000 0000.

Доступ: 0 ≤ тактов ожидания ≤ 3, слово, полуслово и байт. Такты ожидания добавляются при успешном доступе к регистру.

NB: Биты LSEON, LSEBYP, RTCSEL и RTCEN регистра RCC_BDCR лежат в резервном домене. Значит после Сброса они защищены от записи и перед их изменением надо установить бит DBP в регистре PWR_CR. См. *Секцию 7.1.1*. Обнуляются они только Сбросом резервного домена и ничем иным (см. *Секцию 7.1.3*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															BDRST
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTCEN	Reserved					RTCSEL[1:0]		Reserved					LSEBYP	LSERDY	LSEON
rw						rw	rw						rw	r	rw

— Биты 31:17 Резерв, не трогать.

— Бит 16 **BDRST:** Программный сброс резервного домена.

Читается и пишется программно.

0: Ничего

1: Сбрасывает весь домен

NB: Этот сброс на BKPSRAM не влияет, нужен переход уровня защиты с уровня 1 на 0.

— Бит 15 **RTCEN:** Тактирование RTC.

Читается и пишется программно.

0: Выключено

1: Включено

— Биты 14:10 Резерв, не трогать.

— Биты 9:8 **RTCSEL[1:0]:** Выбор источника тактов RTC.

Пишется программно однократно. Сбрасывается только Сбросом всего домена битом BDRST.

00: Нету

01: LSE

10: LSI

11: HSE с предделителем (выбор битами RTCPRE[4:0] в регистре RCC_CFGR

— Биты 7:3 Резерв, не трогать.

— Бит 2 **LSEBYP:** Шунт (обход) LSE.

Читается и пишется программно для обхода LSE в режиме отладки. Пишется только при выключенном LSE.

0: LSE не шунтирован

1: LSE шунтирован

— Бит 1 **LSERDY:** Готовность LSE.

Читается и пишется аппаратно при готовности внешнего 32 КГц LSE. После снятия бита LSEON бит LSERDY снимется через 6 тактов генератора LSE.

0: Не готово

1: Готово

— Бит 0 **LSEON:** Разрешение LSE.

Читается и пишется программно.

0: Выключено

1: Включено

7.3.21. Регистр состояния/управления (RCC_CSR)

Смещение адреса: 0x74

По сбросу: 0x0E00 0000. Сброс системным сбросом, флаги только сбросом питания.

Доступ: 0 ≤ тактов ожидания ≤ 3, слово, полуслово и байт. Такты ожидания добавляются при успешном доступе к регистру.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPWRRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	PORRSTF	PINRSTF	BORRSTF	RMVF	Reserved							
r	r	r	r	r	r	r	rt_w								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														LSIRDY	LSION
														r	rw

Биты флагов ставятся аппаратно. Чистятся записью бита RMVF.

0: Сброса не было

1: Сброс был

— Бит 31 **LPWRRSTF:** Флаг сброса экономного питания.

— Бит 30 **WWDGRSTF:** Флаг сброса WWDG.
— Бит 29 **IWDGRSTF:** Флаг сброса IWDG.
— Бит 28 **SFTRSTF:** Флаг программного сброса.
— Бит 27 **PORRSTF:** Флаг сброса POR/PDR.
— Бит 26 **PINRSTF:** Флаг сброса PIN.
— Бит 25 **BORRSTF:** Флаг сброса BOR.
— Бит 24 **RMVF:** Снятие флагов сброса.
Пишется программно для снятия флагов сброса.
0: Ничего
1: Флаги сброса снимаются
— Биты 23:2 Резерв, не трогать.
— Бит 1 **LSIRDY:** Готовность LSI.
Читается и пишется аппаратно при готовности LSI. После снятия бита LSION бит LSIRDY снимется через 3 такта генератора.
0: Не готово
1: Готово
— Бит 0 **LSION:** Разрешение LSI.
Читается и пишется программно.
0: Выключено
1: Включено

7.3.22. Регистр генератора тактов широкого спектра (RCC_SSCGR)

Смещение адреса: 0x80

По сбросу: 0x0E00 0000.

Доступ: без тактов ожидания, слово, полуслово и байт.

Генерация тактов широкого спектра есть только на главном PLL. Регистр RCC_SSCGR надо писать перед включением или после выключения главного PLL.

NB: Подробности как всегда в описаниях устройств.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SSCGEN	SPREADSEL	Reserved		INCSTEP											
rw	rw			rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INCSTEP			MODPER												
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Бит 31 **SSCGEN:** Включение модуляции широкого спектра
Читается и пишется программно.
0: ВЫКЛ. (Писать после очистки CR[24]=PLLON бит)
1: ВКЛ. (Писать перед установкой CR[24]=PLLON бит)
— Бит 30 **SPREADSEL:** Выбор ширины
Читается и пишется программно.
Писать перед установкой CR[24]=PLLON бит.
0: Централь
1: Спад
— Биты 29:28 Резерв, не трогать.
— Биты 27:13 **INCSTEP:** Шаг инкремента
Читается и пишется программно. Писать перед установкой CR[24]=PLLON бит. Амплитуда профиля модуляции.
— Биты 12:0 **MODPER:** Период модуляции.
Читается и пишется программно. Писать перед установкой CR[24]=PLLON бит. Период профиля модуляции.

7.3.23. Регистр конфигурации PLLI2S (RCC_PLLI2SCFGR)

Смещение адреса: 0x84

По сбросу: 0x2000 3000.

Доступ: без тактов ожидания, слово, полуслово и байт.

Конфигурация выходов PLLI2S по формулам:

$f_{(VCO\ clock)} = f_{(PLLI2S\ clock\ input)} \times (PLLI2SN / PLLM)$

$f_{(PLL\ I2S\ clock\ output)} = f_{(VCO\ clock)} / PLLI2SR$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	PLLI2SR2	PLLI2SR1	PLLI2SR0	Reserved											
	rw	rw	rw												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	PLLI2SN8	PLLI2SN7	PLLI2SN6	PLLI2SN5	PLLI2SN4	PLLI2SN3	PLLI2SN2	PLLI2SN1	PLLI2SN0	Reserved					
	rw	rw	rw	rw	rw	rw	rw	rw	rw						

— Бит 31 Резерв, не трогать.

— Биты 30:28 **PLLI2SR:** Делитель тактов для I2S

Ставится и снимается программно при выключенном PLLI2S. Вместе с предделителем периферии I2S должен дать ошибку 0.3% со стандартным кварцем и 0% с звуковым кварцем. См. *Секцию 28.4.4*.

Внимание: I2S требует частот ≤ 192MHz.

Частота тактов I2S = Частота VCO / PLLR с 2 ≤PLLR ≤7

000: PLLR = 0, Не то

001: PLLR = 1, Не то

010: PLLR = 2

…

111: PLLR = 7

— Биты 27:15 Резерв, не трогать.

— Биты 14:6 **PLLI2SN**: Множитель PLLI2S для VCO

Пишется словами и полусловами программно при выключенном PLLI2S.

Внимание: Выходная частота VCO должна быть от 100 до 432 MHz.

Выходная частота VCO = Входная частота VCO × PLLI2SN при 50 ≤PLLI2SN ≤432

000000000: PLLI2SN = 0, Нельзя

000000001: PLLI2SN = 1, Нельзя

...

000110010: PLLI2SN = 50

…

001100011: PLLI2SN = 99

001100100: PLLI2SN = 100

001100101: PLLI2SN = 101

001100110: PLLI2SN = 102

…

110110000: PLLI2SN = 432

110110001: PLLI2SN = 433, Нельзя

…

111111111: PLLI2SN = 511, Нельзя

NB: Множитель от 50 до 99 возможен для входной частоты VCO больше 1 MHz.

— Биты 5:0 Резерв, не трогать.

7.3.24. Карта регистров RCC

Таблица 34. Карта регистров RCC

Addr. offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	RCC_CR	Reserved				PLL I2SRDY	PLL I2SON	PLL RDY	PLL ON	Reserved				CSSON	HSEBYP	HSERDY	HSEON	HSICAL 7	HSICAL 6	HSICAL 5	HSICAL 4	HSICAL 3	HSICAL 2	HSICAL 1	HSICAL 0	HSITRIM 4	HSITRIM 3	HSITRIM 2	HSITRIM 1	HSITRIM 0	Reserved	HSIRDY	HSION
0x04	RCC_ PLLCFGR	Reserved				PLLQ 3	PLLQ 2	PLLQ 1	PLLQ 0	Reserved	PLLSRC	Reserved				PLLP 1	PLLP 0	Reserved	PLLN 8	PLLN 7	PLLN 6	PLLN 5	PLLN 4	PLLN 3	PLLN 2	PLLN 1	PLLN 0	PLLM 5	PLLM 4	PLLM 3	PLLM 2	PLLM 1	PLLM 0
0x08	RCC_CFGR	MCO2 1	MCO2 0	MCO2PRE2	MCO2PRE1	MCO2PRE0	MCO1PRE2	MCO1PRE1	MCO1PRE0	I2SSRC	MCO1 1	MCO1 0	RTCPRE 4	RTCPRE 3	RTCPRE 2	RTCPRE 1	RTCPRE 0	PPRE2 2	PPRE2 1	PPRE2 0	PPRE1 2	PPRE1 1	PPRE1 0	Reserved		HPRE 3	HPRE 2	HPRE 1	HPRE 0	SWS 1	SWS 0	SW 1	SW 0	
0x0C	RCC_CIR	Reserved								CSSC	Reserved	PLLI2SRDYC	PLLRDYC	HSERDYC	HSIRDYC	LSERDYC	LSIRDYC	Reserved		PLLI2SRDYIE	PLLRDYIE	HSERDYIE	HSIRDYIE	LSERDYIE	LSIRDYIE	CSSF	Reserved	PLLI2SRDYF	PLLRDYF	HSERDYF	HSIRDYF	LSERDYF	LSIRDYF
0x10	RCC_ AHB1RSTR	Reserved		OTGHSRST	Reserved			ETHMACRST	Reserved		DMA2RST	DMA1RST	Reserved								CRCRST	Reserved			GPIOIRST	GPIOHRST	GPIOGRST	GPIOFRST	GPIOERST	GPIODRST	GPIOCRST	GPIOBRST	GPIOARST
0x14	RCC_ AHB2RSTR	Reserved																								OTGFSRST	RNGRST	HSAHRST	CRYPRST	Reserved			DCMIRST
0x18	RCC_ AHB3RSTR	Reserved																															FSMCRST
0x1C	Reserved	Reserved																															
0x20	RCC_ APB1RSTR	Reserved		DACRST	PWRRST	Reserved	CAN2RST	CAN1RST	Reserved	I2C3RST	I2C2RST	I2C1RST	UART5RST	UART4RST	UART3RST	UART2RST	Reserved	SPI3RST	SPI2RST	Reserved			WWDGRST	Reserved		TIM14RST	TIM13RST	TIM12RST	TIM7RST	TIM6RST	TIM5RST	TIM4RST	TIM3RST	TIM2RST
0x24	RCC_ APB2RSTR	Reserved													TIM11RST	TIM10RST	TIM9RST	Reserved	SYSCFGRST	Reserved	SPI1RST	SDIORST	Reserved		ADCRST	Reserved		USART6RST	USART1RST	Reserved		TIM8RST	TIM1RST
0x28	Reserved	Reserved																															
0x2C	Reserved	Reserved																															
0x30	RCC_ AHB1ENR	Reserved	OTGHSULPIEN	OTGHSEN	ETHMACPTPEN	ETHMACRXEN	ETHMACTXEN	ETHMACEN	Reserved	DMA2EN	DMA1EN	CCMDATARAMEN	Reserved	BKPSRAMEN	Reserved					CRCEN	Reserved				GPIOIEN	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
0x34	RCC_ AHB2ENR	Reserved																								OTGFSEN	RNGEN	HASHEN	CRYPEN	Reserved			DCMIEN
0x38	RCC_ AHB3ENR	Reserved																															FSMCEN
0x3C	Reserved	Reserved																															

Addr. offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x40	RCC_APB1ENR	Reserved		DACEN	PWREN	Reserved	CAN2EN	CAN1EN	Reserved	I2C3EN	I2C2EN	I2C1EN	UART5EN	UART4EN	USART3EN	USART2EN	Reserved	SPI3EN	SPI2EN	Reserved			WWDGEN	Reserved			TIM14EN	TIM13EN	TIM12EN	TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN
0x44	RCC_APB2ENR	Reserved													TIM11EN	TIM10EN	TIM9EN	Reserved	SYSCFGEN	Reserved	SPI1EN	SDIOEN	ADC3EN	ADC2EN	ADC1EN	Reserved		USART6EN	USART1EN	Reserved		TIM8EN	TIM1EN
0x48	Reserved	Reserved																															
0x4C	Reserved	Reserved																															
0x50	RCC_AHB1LPENR	Reserved	OTGHSULPILPEN	OTGHSLPEN	ETHMACPTPLPEN	ETHMACRXLPEN	ETHMACTXLPEN	ETHMACLPEN	Reserved	DMA2LPEN	DMA1LPEN	Reserved			BKPSRAMLPEN	SRAM2LPEN	SRAM1LPEN	FLITFLPEN	Reserved	CRCLPEN	Reserved					GPIOILPEN	GPIOHLPEN	GPIOGLPEN	GPIOFLPEN	GPIOELPEN	GPIODLPEN	GPIOCLPEN	GPIOBLPEN	GPIOALPEN
0x54	RCC_AHB2LPENR	Reserved																								OTGFSLPEN	RNGLPEN	HASHLPEN	CRYPLPEN	Reserved			DCMILPEN
0x58	RCC_AHB3LPENR	Reserved																															FSMCLPEN
0x5C	Reserved	Reserved																															
0x60	RCC_APB1LPENR	Reserved		DACLPEN	PWRLPEN	Reserved	CAN2LPEN	CAN1LPEN	Reserved	I2C3LPEN	I2C2LPEN	I2C1LPEN	UART5LPEN	UART4LPEN	USART3LPEN	USART2LPEN	Reserved	SPI3LPEN	SPI2LPEN	Reserved			WWDGLPEN	Reserved			TIM14LPEN	TIM13LPEN	TIM12LPEN	TIM7LPEN	TIM6LPEN	TIM5LPEN	TIM4LPEN	TIM3LPEN	TIM2LPEN
0x64	RCC_APB2LPENR	Reserved													TIM11LPEN	TIM10LPEN	TIM9LPEN	Reserved	SYSCFGLPEN	Reserved	SPI1LPEN	SDIOLPEN	ADC3LPEN	ADC2LPEN	ADC1LPEN	Reserved		USART6LPEN	USART1LPEN	Reserved		TIM8LPEN	TIM1LPEN
0x68	Reserved	Reserved																															
0x6C	Reserved	Reserved																															
0x70	RCC_BDCR	Reserved															BDRST	RTCEN	Reserved				RTCSEL 1	RTCSEL 0	Reserved							LSEBYP	LSERDY	LSEON
0x74	RCC_CSR	LPWRRSTF	WWDGRSTF	WDGRSTF	SFTRSTF	PORRSTF	PADRSTF	BORRSTF	RMVF	Reserved																						LSIRDY	LSION
0x78	Reserved	Reserved																															
0x7C	Reserved	Reserved																															
0x80	RCC_SSCGR	SSCGEN	SPREADSEL	Reserved						INCSTEP													MODPER											
0x84	RCC_PLLI2SCFGR	Reserved	PLLI2SRx							Reserved									PLLI2SNx															

8. Ввод и вывод общего назначения (GPIO)

8.1. Введение в GPIO

Каждый порт I/O общего назначения имеет четыре 32-бит регистра конфигурации (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR и GPIOx_PUPDR), два 32-бит регистра данных (GPIOx_IDR, GPIOx_ODR), 32-бит регистр установки/сброса (GPIOx_BSRR), а 32-бит замка (GPIOx_LCKR) и два 32-бит регистра выбора альтернативных функций (GPIOx_AFRH и GPIOx_AFRL).

8.2. Основные характеристики GPIO

- До 16 I/O
- Режимы выхода: двухтактный или открытый сток + подпорка/подтяжка
- Вывод данных из выводного регистра (GPIOx_ODR) or peripheral (alternate function output)
- Выбор скорости каждого I/O
- Режимы входа: плавающий, подпорка/подтяжка, аналоговый
- Ввод данных во входной регистр (GPIOx_IDR) или периферию (альтернативный ввод)
- Регистр установки/снятия битов (GPIOx_BSRR) для битового доступа к GPIOx_ODR
- Механизм блокировки (GPIOx_LCKR) для заморозки конфигурации I/O
- Аналоговая функция
- Регистр выбора альтернативной функции ввода/вывода (до 16 AF на I/O)
- Быстрое переключение за два такта
- Мультиплексирование ножек I/O как GPIO или периферийные функции

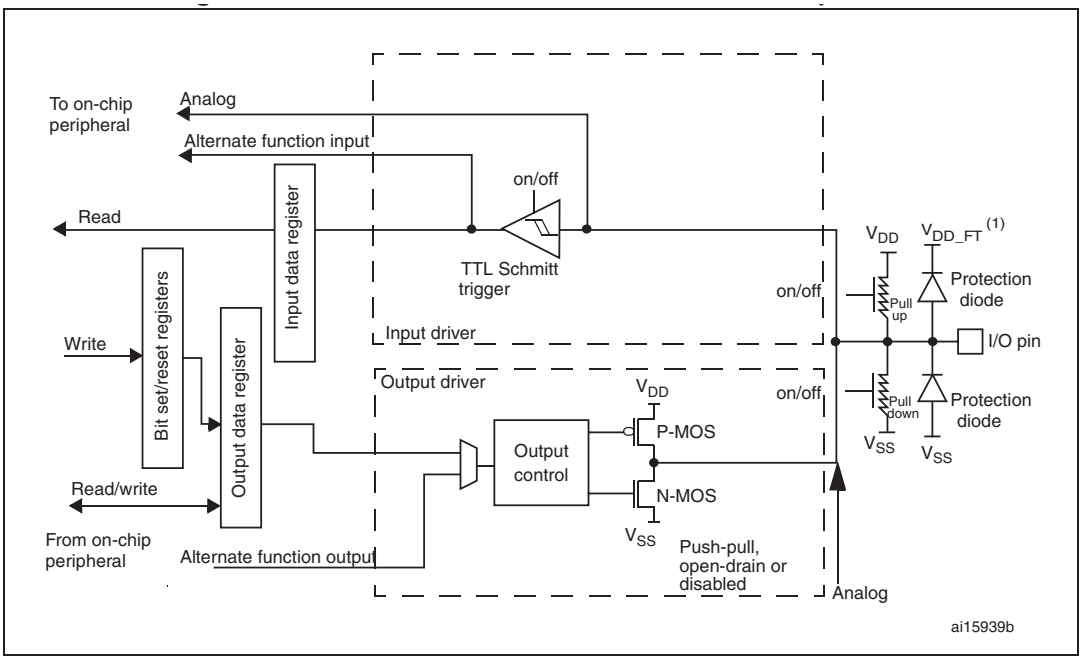
8.3. Функциональное описание GPIO

Конкретные характеристики портов даны в описаниях устройств. Каждый бит порта может быть программно установлен в один из режимов:

- Ввод, плавающий
- Ввод, подпорка вверх
- Ввод, подтянутый вниз
- Аналоговый
- Вывод, открытый сток с подпоркой или подтяжкой
- Вывод, двухтактный с подпоркой или подтяжкой
- Альтернативная функция, двухтактный с подпоркой или подтяжкой
- Альтернативная функция, открытый сток с подпоркой или подтяжкой

Каждый бит I/O порта свободно программируемый, но регистры доступны 32-бит словами, полусловами или байтами. Регистры GPIOx_BSRR и GPIOx_BRR предназначена для атомарного доступа чтения-модификации-записи к регистрам GPIO. Таким образом устраняется риск возникновения IRQ между чтением и модификацией.

Рис.25 Базовая структура бита 5V совместимого порта I/O



1. V_{DD_FT} это потенциал для 5V I/O, отличается от V_{DD}.

Таблица 35. Конфигурация битов порта

MODER(i) [1:0]	OTYPER(i)	OSPEEDR(i) [B:A]	PUPDR(i) [1:0]		I/O configuration	
01	0	SPEED [B:A]	0	0	Выход GP	PP
	0		0	1	Выход GP	PP + PU
	0		1	0	Выход GP	PP + PD
	0		1	1	Резерв	
	1		0	0	Выход GP	OD
	1		0	1	Выход GP	OD + PU
	1		1	0	Выход GP	OD + PD
	1		1	1	Резерв (Выход GP OD)	
10	0	SPEED [B:A]	0	0	AF	PP
	0		0	1	AF	PP + PU
	0		1	0	AF	PP + PD
	0		1	1	Резерв	
	1		0	0	AF	OD
	1		0	1	AF	OD + PU
	1		1	0	AF	OD + PD
	1		1	1	Резерв	
00	x	x	0	0	Вход	Плавающий
	x	x	0	1	Вход	PU
	x	x	1	0	Вход	PD
	x	x	1	1	Резерв (плавающий вход)	
11	x	x	0	0	Input/output	Аналог
	x	x	0	1	Резерв	
	x	x	1	0		
	x	x	1	1		

1. GP = общего назначения, PP = двухтактный, PU = подпорка вверх, PD = подтяжка вниз, OD = открытый сток, AF = альтернативная функция.

8.3.1. Ввод/Вывод общего назначения (GPIO)

Во время и сразу после сброса альтернативные функции отключены и порты I/O ставятся в режим Плавающий Ввод.

После сброса ножки отладки в AF режиме PU/PD:

- PA15: JTDI с подпоркой
- PA14: JTCK/SWCLK с подтяжкой
- PA13: JTMS/SWDAT с подпоркой
- PB4: NJTRST с подпоркой
- PB3: JTDO плавающий

Выходные ножки выдают содержимое выходного регистра данных (`GPIOx_ODR`). Выходной каскад может быть двухтактным или с открытым стоком (при выводе 0 открыт только N-MOS).

Входной регистр данных (`GPIOx_IDR`) считывает данный на ножках каждый такт AHB1.

У всех ножек GPIO есть слабые резисторы подпорки и подтяжки, которые подключаются через регистр GPIOx_PUPDR.

8.3.2. Мультиплексор ножек I/O и картирование

Ножки MCU подключаются к периферии через мультиплексор, по одной единовременно, что может привести к конфликтам общих ножек.

У каждой ножки I/O свой мультиплексор с 16 входами альтернативных функций (AF0 до AF15), которые конфигурируются через регистры GPIOx_AFRL (от 0 до 7) и GPIOx_AFRH (от 8 до 15):

- После сброса все ножки I/O подключены к альтернативной функции системы 0 0 (AF0)
- Альтернативные функции периферии картированы от AF1 до AF13
- EVENTOUT ядра CPU подключен к AF15

См. Рис. 26.

Кроме того, каждое периферийное устройство имеет альтернативные функции, отображённые на различные ножки I/O. Это удобно в малых корпусах микросхем.

Для такого использования I/O надо:

- Системная функция
  Подключаем I/O к AF0 и конфигурируем её в зависимости от нужной функции:
  - JTAG/SWD, после сброса устройства эти ножки назначены для немедленного использования хостом отладчика (контроллером GPIO не управляются)
  - RTC_REFIN: это нога должна стоять в режиме Плавающего ввода
  - MCO1 и MCO2: эти ноги надо поставить в режим альтернативной функции.

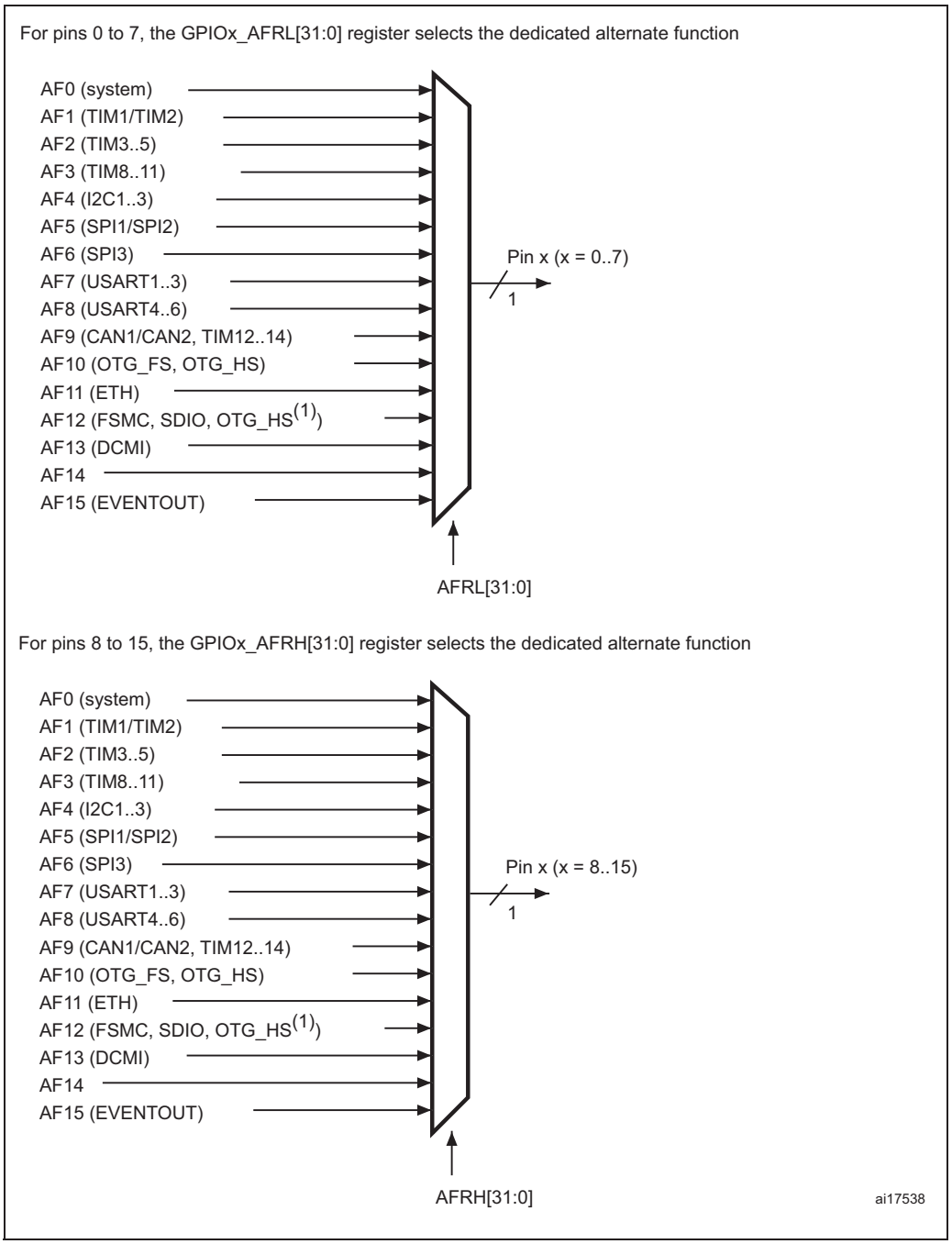
  Можно освобождать все или несколько ножек JTAG/SWD для использования как GPIO (освобождённые ноги в таблице отмечены серым pins).

  См. Секции 7.2.10, и 6.2.10.

Таблица 36. Гибкое назначение ножек SWJ-DP

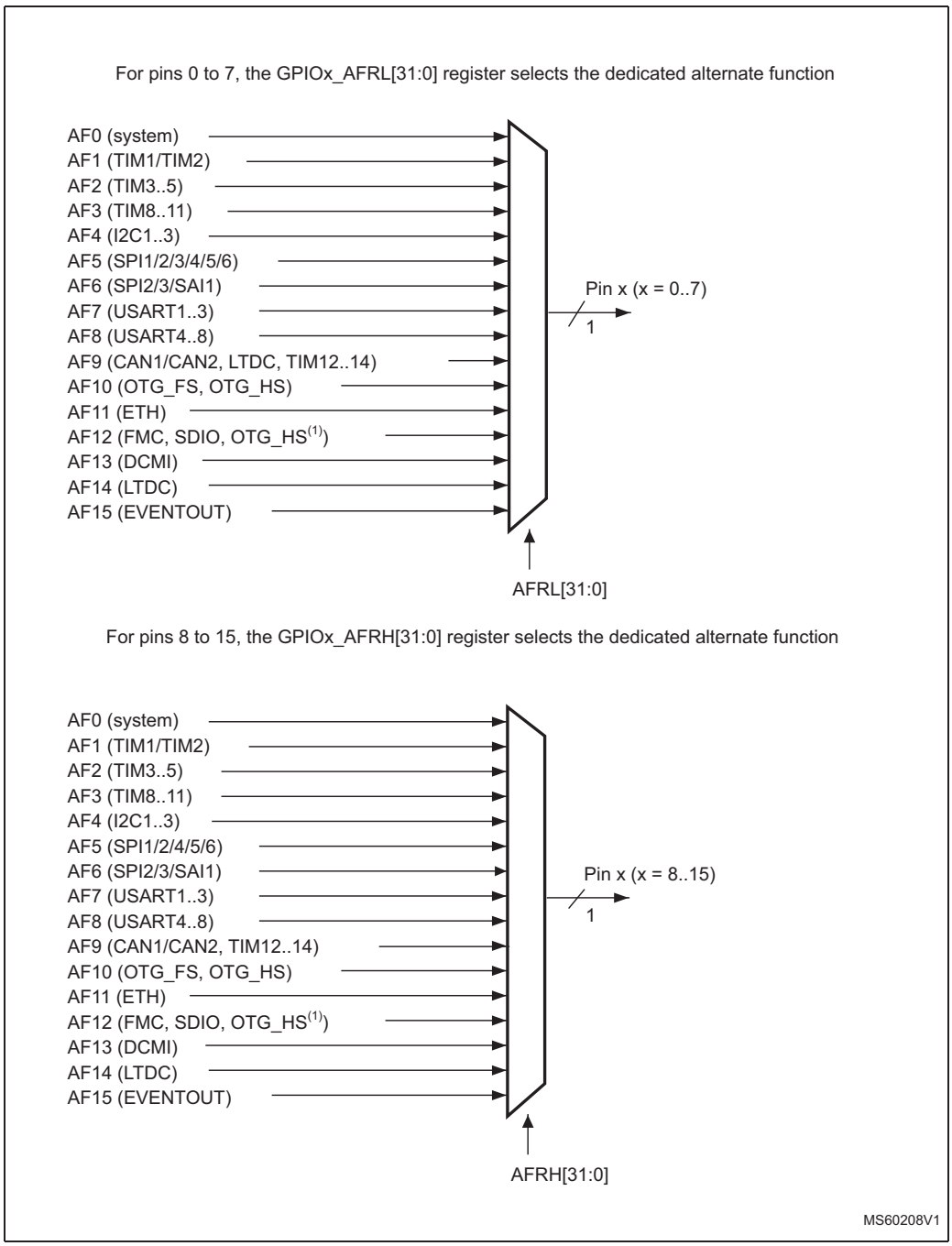
Доступные порты отладки	Назначенные ножки SWJ I/O				
	PA13 / JTMS/ SWDIO	PA14 / JTCK/ SWCLK	PA15 / JTDI	PB3 / JTDO	PB4/ NJTRST
Полный SWJ (JTAG-DP + SW-DP) - Состояние сброса	X	X	X	X	X
Полный SWJ (JTAG-DP + SW-DP) без NJTRST	X	X	X	X	
JTAG-DP Выкл., SW-DP Вкл.	X	X	Свободно		
JTAG-DP Выкл., SW-DP Вкл.					

- GPIO
  В регистре GPIOx_MODER делаем нужные ножки вводом или выводом.
- Альтернативная функция периферии
  Для ADC и DAC в регистре GPIOx_MODER делаем нужные I/O аналоговыми.
  Для другой периферии:
  - В регистре GPIOx_MODER делаем нужные I/O альтернативной функцией
  - В регистрах GPIOx_OTYPER, GPIOx_PUPDR и GPIOx_OSPEEDR ставим тип, подпорку/подтяжку и скорость вывода
  - В регистре GPIOx_AFRL или GPIOx_AFRH подключаем I/O к желанной AF
- EVENTOUT
  Подключаем ножку сигнала EVENTOUT к AF15. EVENTOUT не подключается к ножкам: PC13, PC14, PC15, PH0, PH1 and PI8.



1. Configured in FS.

Рис. 26. Выбор альтернативной функции в STM32F405xx/07xx и STM32F415xx/17xx



1. Configured in FS.

Рис. 27. Выбор альтернативной функции в STM32F42xxx и STM32F43xxx

8.3.3. Регистры управления портом I/O

У каждого GPIO есть четыре 32-бит адресуемых в памяти регистра (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR) для конфигурации до 16 I/O.

Регистр GPIOx_MODER задаёт направление I/O (ввод, вывод, AF, аналог).

Регистры GPIOx_OTYPER и GPIOx_OSPEEDR задают тип (двухтактный или открытый сток) и скорость (ножки скорости I/O напрямую подключены к соответствующему биту регистра GPIOx_OSPEEDR независимо от направления).

Регистр GPIOx_PUPDR подключает к ножкам резисторы подпорки и/или подтяжки независимо от направления.

8.3.4. Регистры данных порта I/O

У каждого GPIO есть два 16-бит адресуемых в памяти регистра: ввод и вывода данных (GPIOx_IDR и GPIOx_ODR). GPIOx_ODR хранит выводимые данные, он доступен по чтению и записи. Вводимые данные пишутся в регистр GPIOx_IDR, он доступен только по чтению.См. *Секции 8.4.5 и 8.4.6*.

8.3.5. Обработка битовых данных I/O

Регистр установки битов (GPIOx_BSRR) это 32-бит регистр, дающий доступ к отдельным битам регистра GPIOx_ODR. Он в два раза больше регистра GPIOx_ODR. На каждый бит регистра GPIOx_ODR, приходится два бита GPIOx_BSRR: BSRR(i) и BSRR(i+SIZE). Запись 1 в BSRR(i) ставит соответствующий бит ODR(i). Запись 1 в BSRR(i+SIZE) чистит соответствующий бит в ODR(i). Запись 0 в GPIOx_BSRR дело бестолковое. При одновременной записи установки и чистки в GPIOx_BSRR, установка приоритетнее.

Использование регистра GPIOx_BSRR операция мгновенная и не блокирует биты GPIOx_ODR. К битам регистра GPIOx_ODR можно обращаться напрямую.

Запрещать прерывания при доступе к GPIOx_ODR на битовом уровне не надо, биты пишутся за одно обращение AHB1.

8.3.6. Механизм блокировки GPIO

Регистры GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL и GPIOx_AFRH. можно заморозить особой последовательностью записи в регистр GPIOx_LCKR.

При правильной последовательности записи LOCK в бит 16 этого регистра (биты LCKR[15:0] должны оставаться неизменными) в управляющих регистрах (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL и GPIOx_AFRH) замораживаются соответствующие биты вплоть до сброса MCU или периферии. Пишут в GPIOx_LCKR только словами. См. *Секцию 8.4.8*.

8.3.7. Альтернативные функции (AF)

Регистры GPIOx_AFRL и GPIOx_AFRH через мультиплексор могут подключать к каждому I/O одну из 16 возможных альтернативных функций периферии для ввода или вывода, всё равно.

8.3.8. Внешние линии прерывания/побудки

Все порты могут иметь входные линии внешнего прерывания. См. *Секции 12.2*. и *12.2.3*.

8.3.9. Конфигурация ввода

При включении порта I/O на ввод:

- Выключается выходной каскад
- Включается входной триггер Шмитта
- Резисторы подпорки и подтяжки включаются по регистру GPIOx_PUPDR
- Данные на ножке I/O читаются в регистр данных каждый такт AHB1
- Данные получают из регистра данных.

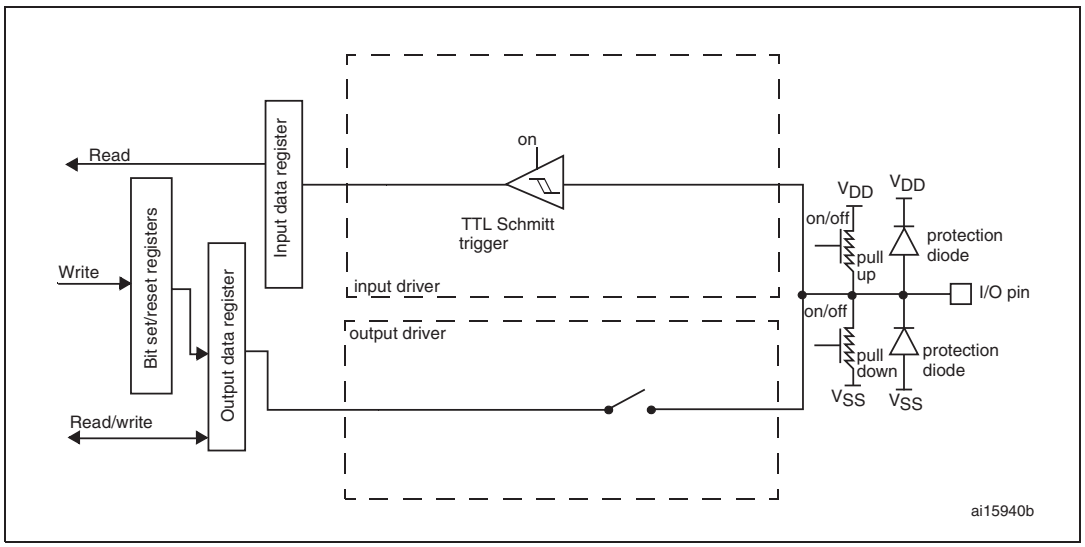


Рис. 28. Входная конфигурация I/O порта.

8.3.10. Конфигурация вывода

При включении порта I/O на вывод:

- Включается выходной каскад:
  - С открытым стоком: “0” в выходном регистре включает N-MOS, а “1” оставляет порт в третьем состоянии Hi-Z (P-MOS никогда не включается)
  - Двухтактный режим: "0" в выходном регистре включает N-MOS, а “1” включает P-MOS
- Включается входной триггер Шмитта
- Резисторы подпорки и подтяжки включаются по регистру `GPIOx_PUPDR`
- Чтение входного регистра выдаёт состояние I/O
- Чтение выходного регистра выдаёт последнее записанное значение.

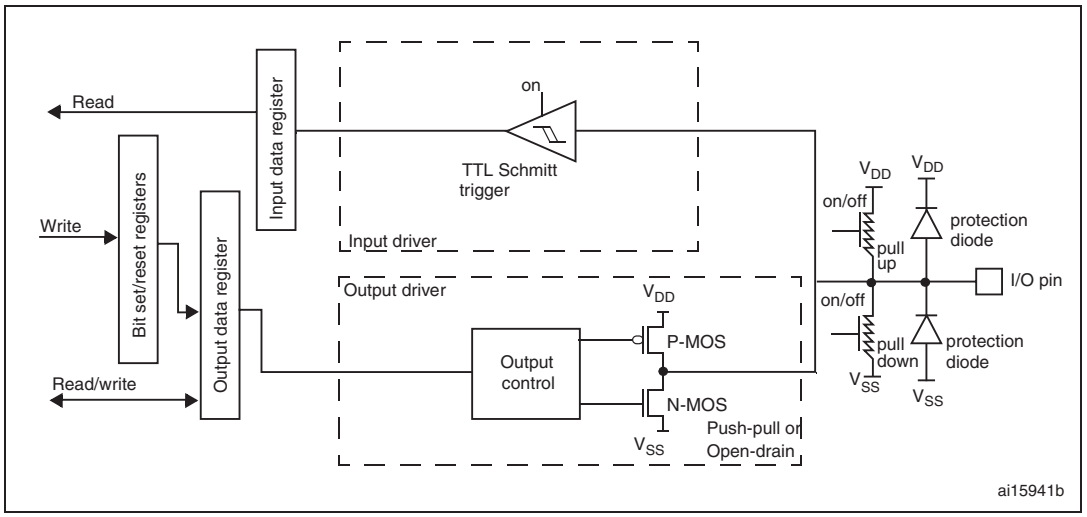


Рис. 29. Выходная конфигурация бита I/O порта.

8.3.11. Конфигурация альтернативных функций

При включении альтернативного режима порта:

- Выходной каскад включается в двухтактный режим или с открытым стоком
- Выходной каскад управляется сигналом от периферии (включение и данные)
- Включается входной триггер Шмитта
- Резисторы подпорки и подтяжки включаются по регистру `GPIOx_PUPDR`
- Данные на ножке I/O читаются в регистр данных каждый цикл AHB1
- Чтение входного регистра выдаёт состояние в режиме с открытым стоком
- Чтение выходного регистра выдаёт последнее записанное значение.

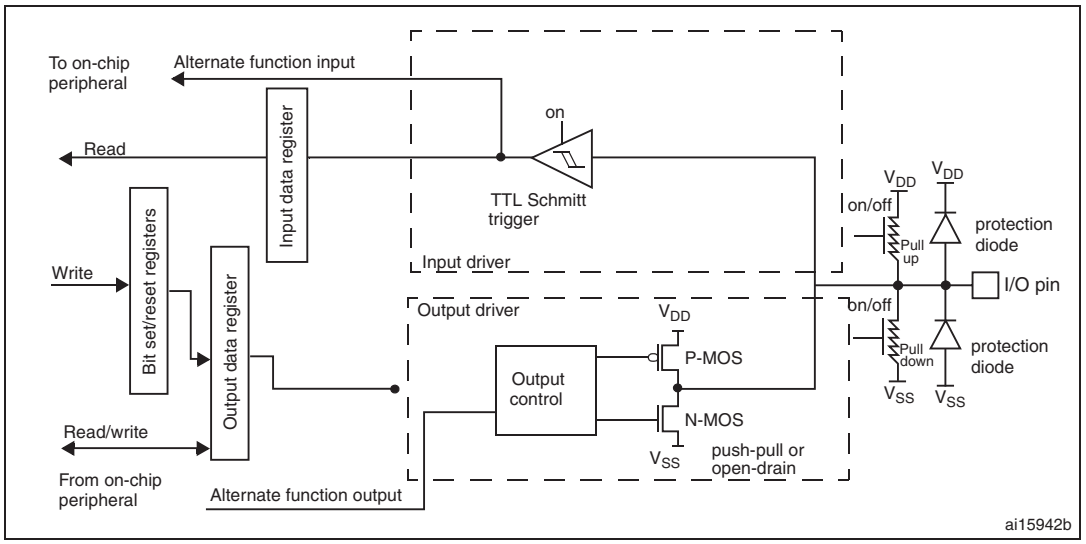


Рис. 30. Альтернативная конфигурация бита I/O порта.

8.3.12. Аналоговая конфигурация

При включении аналогового режима порта:

- Выходной каскад отключается.
- Входной триггер Шмитта отключается, не мешает аналоговому сигналу и постоянно выдаёт “0”.
- Резисторы подпорки и подтяжки отключаются.
- Чтение входного регистра данных выдаёт “0”.

NB: В этом режиме ножки не могут быть совместимы с 5.

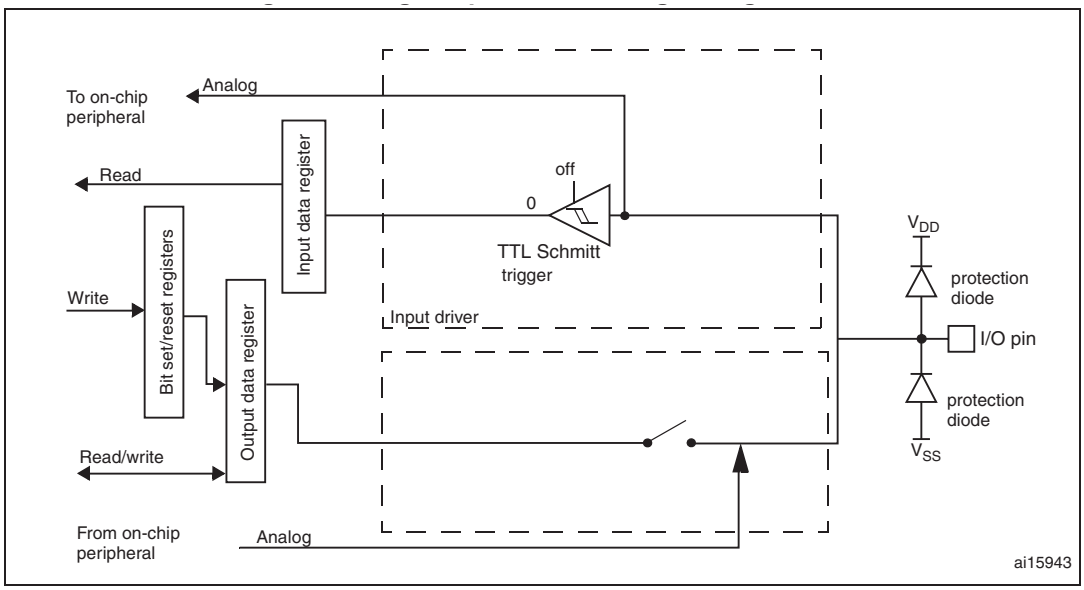


Рис. 31. Аналоговая конфигурация бита I/O порта с высоким импедансом.

8.3.13. Использование ног OSC32_IN/OSC32_OUT как GPIO PC14/PC15

У выключенного генератора LSE ножки OSC32_IN и OSC32_OUT можно использовать как PC14 и PC15 I/O, у включённого они могут быть только OSC32_IN и OSC32_OUT. Он включается установкой бита `LSEON` регистра `RCC_BDCR`. LSE приоритетнее GPIO.

NB: При выключенном домене 1.2 V функциональность PC14/PC15 GPIO теряется, они переводятся в режим аналогового ввода.

8.3.14. Использование ног OSC_IN/OSC_OUT как GPIO PH0/PH1

У выключенного генератора HSE ножки OSC_IN и OSC_OUT можно использовать как PH0 и PH1 I/O, у включённого они могут быть только OSC_IN и OSC_OUT. Он включается установкой бита `HSEON` регистра `RCC_CR`. HSE приоритетнее GPIO.

8.3.15. Выбор альтернативной функции RTC_AF1 и RTC_AF2

GPIO ножки RTC_AF1 и RTC_AF2 можно использовать для обнаружения взлома, события метки времени, как RTC_ALARM или выходы RTC_CALIB RTC.

RTC_AF1 (PC13) можно использовать как:

- RTC_ALARM (RTC Alarm A, RTC Alarm B или RTC Wakeup): биты `OSEL[1:0]` в `RTC_CR`
- Выход RTC_CALIB: бит `COE[23]` в регистре `RTC_CR`
- RTC_TAMP1: событие обнаружения взлома
- RTC_TS: событие метки времени

RTC_AF2 (PI8) можно использовать как:

- RTC_TAMP1: событие обнаружения взлома
- RTC_TAMP2: событие обнаружения взлома
- RTC_TS: событие метки времени

Ножка выбирается в регистре `RTC_TAFCR`:

- `TAMP1INSEL` задаёт ножку для входа RTC_TAMP1
- `TSINSEL` задаёт ножку для входа RTC_TS
- `ALARMOUTTYPE` задаёт выходной каскад RTC_ALARM двухтактным или с открытым стоком.

Таблица 37. Ножка RTC_AF1(1)

Конфигурация и функция	RTC_ALARM включена	RTC_CALIB включена	Tamper включена	Time stamp включена	TAMP1INSEL	TSINSEL TIMESTAMP	ALARMOUTTYPE RTC_ALARM
Alarm out OD	1	Всё равно	Всё равно	Всё равно	Всё равно	Всё равно	0
Alarm out PP	1	Всё равно	Всё равно	Всё равно	Всё равно	Всё равно	1
Calibration out PP	0	1	Всё равно	Всё равно	Всё равно	Всё равно	Всё равно
TAMPER1 вход плав.	0	0	1	0	0	Всё равно	Всё равно
TIMESTAMP и TAMPER1 вход плав.	0	0	1	1	0	0	Всё равно
TIMESTAMP вход плав.	0	0	0	1	Всё равно	0	Всё равно
Стандарт GPIO	0	0	0	0	Всё равно	Всё равно	Всё равно

1. OD: открытый сток; PP: двухтактный.

Таблица 38. Ножка RTC_AF2

Конфигурация и функция	Tamper включена	Time stamp включена	TAMP1INSEL	TSINSEL TIMESTAMP	ALARMOUTTYPE RTC_ALARM
TAMPER1 вход плав.	1	0	1	Всё равно	Всё равно
TIMESTAMP и TAMPER1 вход плав.	1	1	1	1	Всё равно
TIMESTAMP вход плав.	0	1	Всё равно	1	Всё равно
Стандарт GPIO	0	0	Всё равно	Всё равно	Всё равно

8.4. Регистры GPIO

Все регистры доступны 32-бит словами, полусловами и байтами.

8.4.1. Регистр режима порта (GPIOx_MODER) (x=A..K)

Смещение адреса: 0x00

По сбросу:

- 0xA800 0000 порт A
- 0x0000 0280 порт B
- 0x0000 0000 другие порты

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 2y:2y+1

MODERy[1:0]:Биты конфигурации порта **x** (**y**=0..15)

Пишутся программно.

00: Вход (по сбросу)

01: Просто вывод

10: Альтернативная функция

11: Аналоговый

8.4.2. Регистр типа выхода порта (GPIOx_OTYPER) (x=A..K)

Смещение адреса: 0x04

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:16 Резерв, не трогать.

— Биты 15:0 **OTy[15:0]**:Биты конфигурации порта **x** (**y**=0..15)

Пишутся программно.

0: Вывод двухтактный

1: Вывод открытый сток

8.4.3. Регистр скорости выхода порта (GPIOx_SPEEDR) (x=A..K)

Смещение адреса: 0x08

По сбросу:

- 0x0C00 0000 порт A
- 0x0000 00C0 порт B
- 0x0000 0000 другие порты

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEEDR15 [1:0]		OSPEEDR14 [1:0]		OSPEEDR13 [1:0]		OSPEEDR12 [1:0]		OSPEEDR11 [1:0]		OSPEEDR10 [1:0]		OSPEEDR9 [1:0]		OSPEEDR8 [1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1 [1:0]		OSPEEDR0 1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 2y:2y+1

MODERy[1:0]: Биты скорости порта **x** (**y**=0..15)

Пишутся программно.

00: Низкая

01: Средняя

10: Высокая

11: Очень высокая

8.4.4. Регистр подтяжки/подпорки порта (GPIOx_PUPDR) (x=A..K)

Смещение адреса: 0x0C

По сбросу:

- 0x6400 0000 порт A
- 0x0000 0100 порт B
- 0x0000 0000 другие порты

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 2y:2y+1

PUPDRy[1:0]: Биты скорости порта **x** (**y**=0..15)

Пишутся программно.

00: Ничего

01: Подпорка вверх

10: Подтяжка вниз

11: Резерв

8.4.5. Регистр данных ввода порта (GPIOx_IDR) (x=A..K)

Смещение адреса: 0x10

По сбросу: 0x0000 XXXX , где X - не определено.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

— Биты 31:16 Резерв, не трогать.

— Биты 15:0 **IDRy:** Входные данные (**y**=0..15)

Входные данные порта. Чтение только словом.

8.4.6. Регистр данных вывода порта (GPIOx_ODR) (x=A..K)

Смещение адреса: 0x14

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:16 Резерв, не трогать.

— Биты 15:0 **ODRy:** Выходные данные (**y**=0..15)

Выходные данные порта.

NB: При атомарном доступе биты ODR можно менять индивидуально записью в регистр GPIOx_BSRR (x = A .. K).

8.4.7. Регистр установки/сброса битов порта (GPIOx_BSRR) (x=A..K)

Смещение адреса: 0x18

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

— Биты 31:16 **BRy**: Сброс бита **y** в **ODRx** (**y**=0..15).

0: Не изменяется

1: Сбрасывается

— Биты 15:0 **BSRy:** Установка бита **y** в **ODRx** (**y**=0..15)

0: Не изменяется

1: Ставится

8.4.8. Регистр блокировки конфигурации порта (GPIOx_LCKR) (x=A..K)

Используется для блокировки конфигурации портов GPIO по битам `LCKR[15:0]` после завершения последовательности записи LOCK в бит 16 (`LCKK`). Биты [15:0] используются для блокировки конфигурации GPIO. Во время последовательности записи значения замка `LCKR[15:0]` не должны изменяться. После этого порты невозможно конфигурировать вплоть до следующего сброса MCU или периферии. Каждый бит замка блокирует соответствующий регистр конфигурации.

Смещение адреса: 0x1C

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															LCKK
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:17 Резерв, не трогать.

— Бит 16 **LCKK:** Ключ замка.

Читать можно всегда, писать только особой последовательностью.

0: Конфигурация портов возможна

1: Ключ активен. GPIOx_LCKR закрыт до следующего сброса.

Последовательность записи ключа LOCK:

WR LCKR[16] = ‘1’ + LCKR[15:0]

WR LCKR[16] = ‘0’ + LCKR[15:0]

WR LCKR[16] = ‘1’ + LCKR[15:0]

RD LCKR

RD LCKR[16] = ‘1’ (необязательно, но подтверждает активность ключа)

NB: Во время последовательности LOCK значения LCK[15:0] не могут изменяться. Ошибка прерывает последовательность.

— Биты 15:0 **LCKy:** Блокировка порта **y** (**y**=0..15)

Эти биты можно писать только при LCKK=0.

0: Конфигурация порта **y** не блокируется

1: Конфигурация порта **y** блокируется

8.4.9. Младший регистр альтернативных функций (GPIOx_AFRL) (x=A..K)

Смещение адреса: 0x20

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFRL7[3:0]				AFRL6[3:0]				AFRL5[3:0]				AFRL4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFRL3[3:0]				AFRL2[3:0]				AFRL1[3:0]				AFRL0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:16 **AFRLy**: Альтернативная функция для порта **x** бит **y** (y=0..7)

Пишутся программно.

Выбор AFRLy:

0000: AF0 1000: AF8

0001: AF1 1001: AF9

0010: AF2 1010: AF10

0011: AF3 1011: AF11

0100: AF4 1100: AF12

0101: AF5 1101: AF13

0110: AF6 1110: AF14

0111: AF7 1111: AF15

8.4.10. Старший регистр альтернативных функций (GPIOx_AFRH) (x=A..J)

Смещение адреса: 0x24

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFRH15[3:0]				AFRH14[3:0]				AFRH13[3:0]				AFRH12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFRH11[3:0]				AFRH10[3:0]				AFRH9[3:0]				AFRH8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:16 **AFRHy**: Альтернативная функция для порта **x** бит **y** (y=0..7)
Пишутся программно.
Выбор AFRHy:

0000: AF0	1000: AF8
0001: AF1	1001: AF9
0010: AF2	1010: AF10
0011: AF3	1011: AF11
0100: AF4	1100: AF12
0101: AF5	1101: AF13
0110: AF6	1110: AF14
0111: AF7	1111: AF15

8.4.11. Карта регистров GPIO

Таблица 39. Карта регистров GPIO

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**GPIOA_ MODER**	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
	Reset value	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00	**GPIOB_ MODER**	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
0x00	**GPIOx_MODER** (where x = C..I/J/K)	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**GPIOx_ OTYPER** (where x = A..I/J/K)	Reserved																OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**GPIOx_ OSPEEDR** (where x = A..I/J/K except B)	OSPEEDR15[1:0]		OSPEEDR14[1:0]		OSPEEDR13[1:0]		OSPEEDR12[1:0]		OSPEEDR11[1:0]		OSPEEDR10[1:0]		OSPEEDR9[1:0]		OSPEEDR8[1:0]		OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1[1:0]		OSPEEDR0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**GPIOB_ OSPEEDR**	OSPEEDR15[1:0]		OSPEEDR14[1:0]		OSPEEDR13[1:0]		OSPEEDR12[1:0]		OSPEEDR11[1:0]		OSPEEDR10[1:0]		OSPEEDR9[1:0]		OSPEEDR8[1:0]		OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1[1:0]		OSPEEDR0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
0x0C	**GPIOA_PUPDR**	PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]		PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
	Reset value	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**GPIOB_PUPDR**	PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]		PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C	**GPIOx_PUPDR** (where x = C..I/J/K)	PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]		PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**GPIOx_IDR** (where x = A..I/J/K)	Reserved																IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
	Reset value																	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x14	**GPIOx_ODR** (where x = A..I/J/K)	Reserved																ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**GPIOx_BSRR** (where x = A..I/J/K)	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0	BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**GPIOx_LCKR** (where x = A..I/J/K)	Reserved															LCKK	LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**GPIOx_AFRL** (where x = A..I/J/K)	AFRL7[3:0]				AFRL6[3:0]				AFRL5[3:0]				AFRL4[3:0]				AFRL3[3:0]				AFRL2[3:0]				AFRL1[3:0]				AFRL0[3:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	**GPIOx_AFRH** (where x = A..I/J)	AFRH15[3:0]				AFRH14[3:0]				AFRH13[3:0]				AFRH12[3:0]				AFRH11[3:0]				AFRH10[3:0]				AFRH9[3:0]				AFRH8[3:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

9. Контроллер конфигурации системы (SYSCFG)

В основном используется для картирования памяти области кода, выбора интерфейса Ethernet PHY и управления внешними линиями прерываний GPIO.

9.1. Ячейка компенсации I/O

По умолчанию она не используется. Но при конфигурации выходного каскада I/O в режим 50 MHz или 100 MHz, рекомендуется включать её для уменьшения шума на питании I/O. О включении ячейки компенсации говорит флаг READY.

Её можно использовать только при напряжении питания от 2.4 до 3.6 V.

9.2. Регистры SYSCFG для STM32F405xx/07xx и STM32F415xx/17xx

9.2.1. Регистр картирования памяти (SYSCFG_MEMRMP)

Используется для конфигурации типа памяти, доступной по адресу 0x0000 0000. Это программный обход ножек BOOT. После сброса отражают режим, установленный ножками BOOT. При загрузке из основной Flash памяти при ножках BOOT, установленных в 10 [(BOOT1,BOOT0) = (1,0)] этот регистр содержит 0x00.

Если по адресу 0x0000 0000 стоит FSMC, то можно картировать только первые два региона контроллера памяти Банка 1 (Банк 1 NOR/PSRAM 1 и NOR/PSRAM 2). В режиме картирования CPU обращается к внешней памяти по шине ICode вместо Системной, что быстрее будет.

Смещение адреса: 0x00

По сбросу: 0x0000 000X , где X - режим памяти, выбранный ножками BOOT.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														MEM_MODE	
														rw	rw

— Биты 31:2 Резерв, не трогать.
— Биты 1:0 **MEM_MODE**: Режим картирования по адресу 0x0000 0000
Читаются и пишутся программно.
После сброса принимают значение, выбранное ножками Boot (кроме FSMC).
00: Основная Flash память
01: Системная Flash память
10: FSMC Банк 1 (NOR/PSRAM 1 и 2)
11: Встроенная SRAM (SRAM1)
NB: См. Секцию 2.3.

9.2.2. Регистр конфигурации периферии (SYSCFG_PMC)

Смещение адреса: 0x04
По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								MII_RMII_SEL	Reserved						
								rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															

— Биты 31:24 Резерв, не трогать.
— Бит 23 **MII_RMII_SEL**: Выбор интерфейса Ethernet PHY
Читаются и пишутся программно.
0: MII
1: RMII PHY
NB: Писать нужно пока MAC в сбросе и до разрешения тактов MAC.
— Биты 22:0 Резерв, не трогать.

9.2.3. Регистр конфигурации внешних прерываний 1 (SYSCFG_EXTICR1)

Смещение адреса: 0x08
По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:16 Резерв, не трогать.
— Биты 15:0 **EXTIx[3:0]**: Конфигурация EXTIx (x = 0 до 3)
Читаются и пишутся программно.
0000: PA[x]
0001: PB[x]
0010: PC[x]
0011: PD[x]
0100: PE[x]
0101: PF[x]
0110: PG[x]
0111: PH[x]
1000: PI[x].

9.2.4. Регистр конфигурации внешних прерываний 2 (SYSCFG_EXTICR2)

Смещение адреса: 0x0C
По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI7[3:0]				EXTI6[3:0]				EXTI5[3:0]				EXTI4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:16 Резерв, не трогать.

— Биты 15:0 **EXTIx[3:0]**: Конфигурация EXTIx (x = 4 до 7)

Читаются и пишутся программно.

0000: PA[x]
0001: PB[x]
0010: PC[x]
0011: PD[x]
0100: PE[x]
0101: PF[x]
0110: PG[x]
0111: PH[x]
1000: PI[x].

9.2.5. Регистр конфигурации внешних прерываний 3 (SYSCFG_EXTICR3)

Смещение адреса: 0x10

По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI11[3:0]				EXTI10[3:0]				EXTI9[3:0]				EXTI8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:16 Резерв, не трогать.

— Биты 15:0 **EXTIx[3:0]**: Конфигурация EXTIx (x = 8 до 11)

Читаются и пишутся программно.

0000: PA[x]
0001: PB[x]
0010: PC[x]
0011: PD[x]
0100: PE[x]
0101: PF[x]
0110: PG[x]
0111: PH[x]
1000: PI[x].

9.2.6. Регистр конфигурации внешних прерываний 4 (SYSCFG_EXTICR4)

Смещение адреса: 0x14

По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI15[3:0]				EXTI14[3:0]				EXTI13[3:0]				EXTI12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:16 Резерв, не трогать.

— Биты 15:0 **EXTIx[3:0]**: Конфигурация EXTIx (x = 12 до 15)

Читаются и пишутся программно.

0000: PA[x]
0001: PB[x]
0010: PC[x]

0011: PD[x]
0100: PE[x]
0101: PF[x]
0110: PG[x]
0111: PH[x]
1000: PI[x].

9.2.7. Карта регистров STM32F405xx/07xx и STM32F415xx/17xx

Таблица 40. Карта регистров STM32F405xx/07xx и STM32F415xx/17xx

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	SYSCFG_ MEMRMP	Reserved																														MEM_MODE	
	Reset value																															x	x
0x04	SYSCFG_PMC	Reserved								MII_RMII_SEL	Reserved							Reserved															
	Reset value									0																							
0x08	SYSCFG_EXTICR1	Reserved																EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	SYSCFG_EXTICR2	Reserved																EXTI7[3:0]				EXTI6[3:0]				EXTI5[3:0]				EXTI4[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	SYSCFG_EXTICR3	Reserved																EXTI11[3:0]				EXTI10[3:0]				EXTI9[3:0]				EXTI8[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	SYSCFG_EXTICR4	Reserved																EXTI15[3:0]				EXTI14[3:0]				EXTI13[3:0]				EXTI12[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	SYSCFG_CMPCR	Reserved																								READY	Reserved						CMP_PD
	Reset value																									0							0

9.3. Регистры SYSCFG для STM32F2xxx и STM32F43xxx

9.3.1. Регистр картирования памяти (SYSCFG_MEMRMP)

Используется для конфигурации типа памяти, доступной по адресу 0x0000 0000:

- Для физического картирования памяти и программного обхода ножек BOOT есть три бита.
- После сброса они получают значение, предписанное ножками BOOT. Если грузиться из главной Flash памяти при ножках BOOT равных 10 [(BOOT1,BOOT0) = (1,0)] регистр получает значение 0x00.
- Другие биты используются для обмена FMC SDRAM Банк 1/2 и FMC Банк 3/4 и конфигурации картирования Flash Банк 1/2

Для картирования есть два блока FMC:

- FMC Банк 1 (NOR/PSRAM 1 и 2):
  Может картировать только первые два региона контроллера памяти Банка 1 (Банк 1 NOR/PSRAM 1 и NOR/PSRAM 2).
- FMC SDRAM Банк 1.

В режиме картирования CPU обращается к внешней памяти по шине ICode вместо системной, что есть изрядно шустрее.

Загружать из NOR Flash или SDRAM нельзя. Размещать регионы по адресу 0x0000 0000 надо программно.

Смещение адреса: 0x00

По сбросу: 0x0000 000X , где X - режим памяти, выбранный ножками BOOT.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				SWP_FMC		Res.	FB_ MODE	Reserved					MEM_MODE[2:0]		
				rw	rw		rw						rw	rw	rw

— Биты 31:12 Резерв, не трогать.

— Биты 11:10 SWP_FMC: Обмен картирования памяти FMC

Читаются и пишутся программно. Меняют местами FMC SDRAM Банк 1/2 и FMC Банк 3/4 (SDRAM Банк 1/2 and NAND Банк 2/PCCARD Bank) для выполнения кода из банков SDRAM без физического картирования по адресу 0x0000 0000.

00: Без обмена

01: Меняются банки SDRAM и NAND Bank 2/PCCARD. SDRAM Bank 1 и 2 отображены на NAND Bank 2 (0x8000 0000) и PCCARD Bank (0x9000 0000). NAND Bank 2 и PCCARD Bank отображены по адресам 0xC000 0000 и 0xD000 0000.

10: Резерв

11: Резерв

— Бит 9 Резерв, не трогать.

— Бит 8 **FB_MODE**: Картирование банков Flash Bank 1/2

Читается и пишется программно.

0: Flash Bank 1 стоит на 0x0800 0000 (алиас на 0x0000 0000) и Flash Bank 2 стоит на 0x0810 0000 (алиас на 0x0010 0000)

1: Flash Bank 2 стоит на 0x0800 0000 (алиас на 0x0000 0000) и Flash Bank 1 стоит на 0x0810 0000 (алиас на 0x0010 0000)

— Биты 7:3 Резерв, не трогать.

— Биты 2:0 **MEM_MODE[2:0]**: Режим картирования по адресу 0x0000 0000

Читаются и пишутся программно.

После сброса принимают значение, выбранное ножками Boot (кроме FMC).

000: Основная Flash память

001: Системная Flash память

010: FMC Банк 1 (NOR/PSRAM 1 и 2)

011: Встроенная SRAM (SRAM1)

100: FMC/SDRAM Bank 1

Иные конфигурации в резерве

NB: См. Секцию 2.3.

9.3.2. Регистр конфигурации периферии (SYSCFG_PMC)

Смещение адреса: 0x04

По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								MII_RMII _SEL	Reserved				ADCxDC2		
								rw					rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															

— Биты 31:24 Резерв, не трогать.

— Бит 23 **MII_RMII_SEL**: Выбор интерфейса Ethernet PHY

Читаются и пишутся программно.

0: MII

1: RMII PHY

NB: Писать нужно пока MAC в сбросе и до разрешения тактов MAC.

— Биты 22:19 Резерв, не трогать.

— Биты 18:16 **ADCxDC2**:

0: Ничего.

1: См. документ AN4073.

NB: Эти биты можно ставить только при соблюдении условий:

- Частота тактов ADC выше или равна 30 MHz.
- Если преобразования ADC не стартуют одновременно и времена преобразования различаются, то должен быть выбран только один бит ADCxDC2.
- Биты не должны ставиться при стоящем бите ADCDC1 в регистре PWR_CR.

— Биты 15:0 Резерв, не трогать.

9.3.3. Регистр конфигурации внешних прерываний 1 (SYSCFG_EXTICR1)

Смещение адреса: 0x08

По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:16 Резерв, не трогать.

— Биты 15:0 **EXTIx[3:0]**: Конфигурация EXTIx (x = 0 до 3)

Читаются и пишутся программно.

0000: PA[x]
0001: PB[x]
0010: PC[x]
0011: PD[x]
0100: PE[x]
0101: PF[x]
0110: PG[x]
0111: PH[x]
1000: PI[x].
1001: PJ[x].
1010: PK[x].

9.3.4. Регистр конфигурации внешних прерываний 2 (SYSCFG_EXTICR2)

Смещение адреса: 0x0C

По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI7[3:0]				EXTI6[3:0]				EXTI5[3:0]				EXTI4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:16 Резерв, не трогать.

— Биты 15:0 **EXTIx[3:0]**: Конфигурация EXTIx (x = 4 до 7)

Читаются и пишутся программно.

0000: PA[x]
0001: PB[x]
0010: PC[x]
0011: PD[x]
0100: PE[x]
0101: PF[x]
0110: PG[x]
0111: PH[x]
1000: PI[x].
1001: PJ[x].
1010: PK[x].

9.3.5. Регистр конфигурации внешних прерываний 3 (SYSCFG_EXTICR3)

Смещение адреса: 0x10

По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI11[3:0]				EXTI10[3:0]				EXTI9[3:0]				EXTI8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:16 Резерв, не трогать.

— Биты 15:0 **EXTIx[3:0]**: Конфигурация EXTIx (x = 8 до 11)

Читаются и пишутся программно.

0000: PA[x]
0001: PB[x]
0010: PC[x]
0011: PD[x]
0100: PE[x]
0101: PF[x]
0110: PG[x]
0111: PH[x]
1000: PI[x].
1001: PJ[x].

NB: PK[11:8] не используются.

9.3.6. Регистр конфигурации внешних прерываний 4 (SYSCFG_EXTICR4)

Смещение адреса: 0x14

По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI15[3:0]				EXTI14[3:0]				EXTI13[3:0]				EXTI12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:16 Резерв, не трогать.

— Биты 15:0 **EXTIx[3:0]**: Конфигурация EXTIx (x = 12 до 15)

Читаются и пишутся программно.

0000: PA[x]
0001: PB[x]
0010: PC[x]
0011: PD[x]
0100: PE[x]
0101: PF[x]
0110: PG[x]
0111: PH[x]
1000: PI[x].
1001: PJ[x].

NB: PK[15:12] не используются.

9.3.7. Регистр ячейки компенсации (SYSCFG_CMPCR)

Смещение адреса: 0x20

По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							READY	Reserved							CMP_PD
							r								rw

— Биты 31:9 Резерв, не трогать.
— Бит 15:0 **READY**: Флаг готовности
 0: Не готова
 1: Готова
— Биты 7:1 Резерв, не трогать.
— Бит 0 **CMP_CD**: Включение ячейки
 0: Выкл.
 1: Вкл.

9.3.8. Карта регистров STM32F42xxx и STM32F43xxx

Таблица 40. Карта регистров STM32F42xxx и STM32F43xxx

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	SYSCFG_MEMRMP	Reserved																				SWP_FMC		Reserved	FB_MODE	Reserved					MEM_MODE		
	Reset value																					0	0		0						x	x	x
0x04	SYSCFG_PMC	Reserved								MII_RMII_SEL	Reserved				ADC3DC2	ADC2DC2	ADC1DC2	Reserved															
	Reset value									0					0	0	0																
0x08	SYSCFG_EXTICR1	Reserved																EXTI3[3:0]				EXTI2[3:0]				EXTI1[3:0]				EXTI0[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	SYSCFG_EXTICR2	Reserved																EXTI7[3:0]				EXTI6[3:0]				EXTI5[3:0]				EXTI4[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	SYSCFG_EXTICR3	Reserved																EXTI11[3:0]				EXTI10[3:0]				EXTI9[3:0]				EXTI8[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	SYSCFG_EXTICR4	Reserved																EXTI15[3:0]				EXTI14[3:0]				EXTI13[3:0]				EXTI12[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	SYSCFG_CMPCR	Reserved																							READY	Reserved							CMP_PD
	Reset value																								0								0

10. Контроллер ПДП (DMA)

10.1. Введение в DMA

Прямой доступ к памяти работает как при передачах периферия-память, так и память-память. Два контроллера DMA имеют в общем 16 потоков (8 в каждом контроллере). Каждый поток может иметь до 8 каналов (запросов) в общем. Каждый имеет арбитр управления приоритетами.

10.2. Основные свойства DMA

Это:

- Два ведущих шины AHB, один для доступа к памяти и один к периферии
- Программируемый интерфейс ведомого AHB с только 32-бит доступом
- 8 потоков в каждом контроллере DMA до 8 каналов (запросов) на поток

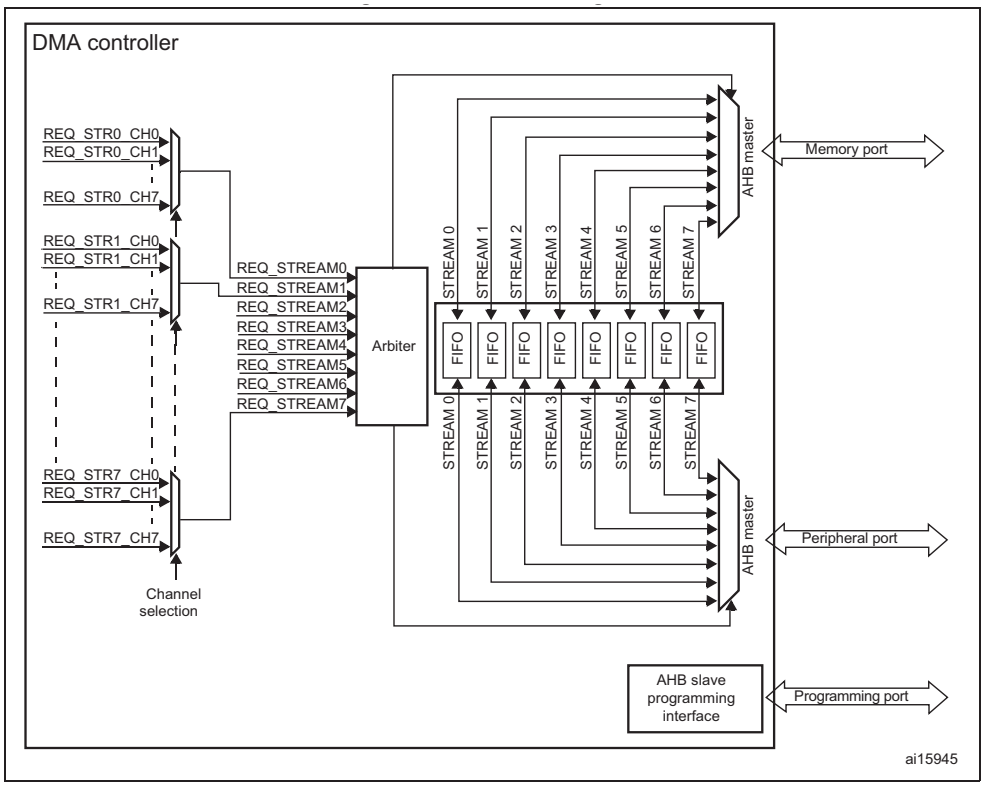
- 32 четырёхсловных буфера на поток, работающих как FIFO или напрямую:
  - Режим FIFO: программируемый порог в 1/4, 1/2 или 3/4 размера FIFO
  - Прямой режим

    Каждый запрос DMA немедленно запускает обмен с памятью. В этом режиме (FIFO выключен) для передач память->периферия DMA загружает из памяти только одно данное во внутренний FIFO для передачи сразу после запроса периферии.
- Аппаратура может сделать каждый поток:
  - регулярным каналом с передачами память<->периферия и память->память
  - двухбуферным каналом с поддержкой двух буферов и со стороны памяти
- Каждый из 8 потоков подключён к своему аппаратному каналу DMA (запросу)
- Приоритеты запросов потоков DMA задаются программно (4 уровня: очень высокий, высокий, средний, низкий) или аппаратно в случае равенства (запрос 0 приоритетнее 1 и т.д.)
- У контроллера DMA2 есть программный запуск передач память-память
- Для запроса потока программно может быть выбран один из 8 возможных запросов каналов.
- Число передаваемых данных определяется контроллером DMA или периферией:
  - Контроллер DMA: число данных от 1 до 65535
  - Периферия: конец передачи указывается аппаратным сигналом от устройства
- Независимая ширина передач источника и получателя (байт, полуслово, слово): DMA автоматически упаковывает/распаковывает данные. Доступно только в режиме FIFO.
- Инкрементная и нет адресация источника и получателя
- Поддержка инкрементной групповой передачи по 4, 8 или 16 частей. Размер группы программируемый, обычно в половину размера FIFO или периферии
- Каждый поток поддерживает кольцевые буферы
- 5 флагов событий (Половина передачи DMA, Конец передачи DMA, Ошибка передачи DMA, Ошибка FIFO DMA, Ошибка Прямого режима) по логическому ИЛИ подаются на один запрос прерывания каждого потока.

10.3. Функциональное описание DMA

10.3.1. Общее описание

Рис. 32. Блок-схема DMA.



Контроллер DMA как ведущий шины AHB захватывает управление матрицей шины AHB для запуска передач AHB.

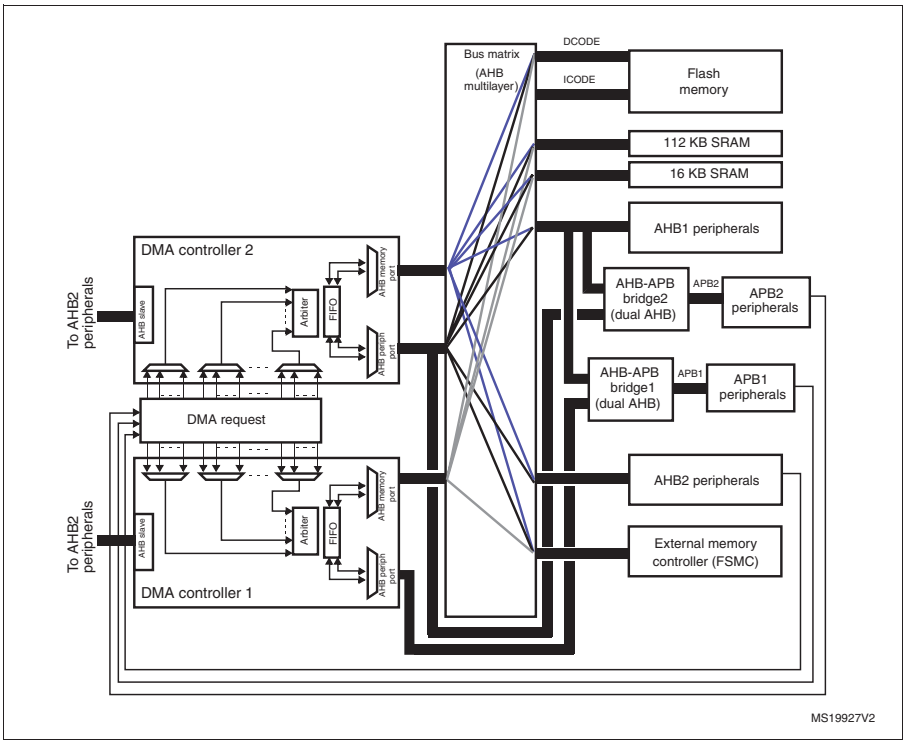
Его передачи:

- периферия->память
- память->периферия
- память->память

Контроллер DMA имеет два ведущих порта AHB: для памяти и для периферии. Но порт периферии имеет доступ к памяти.

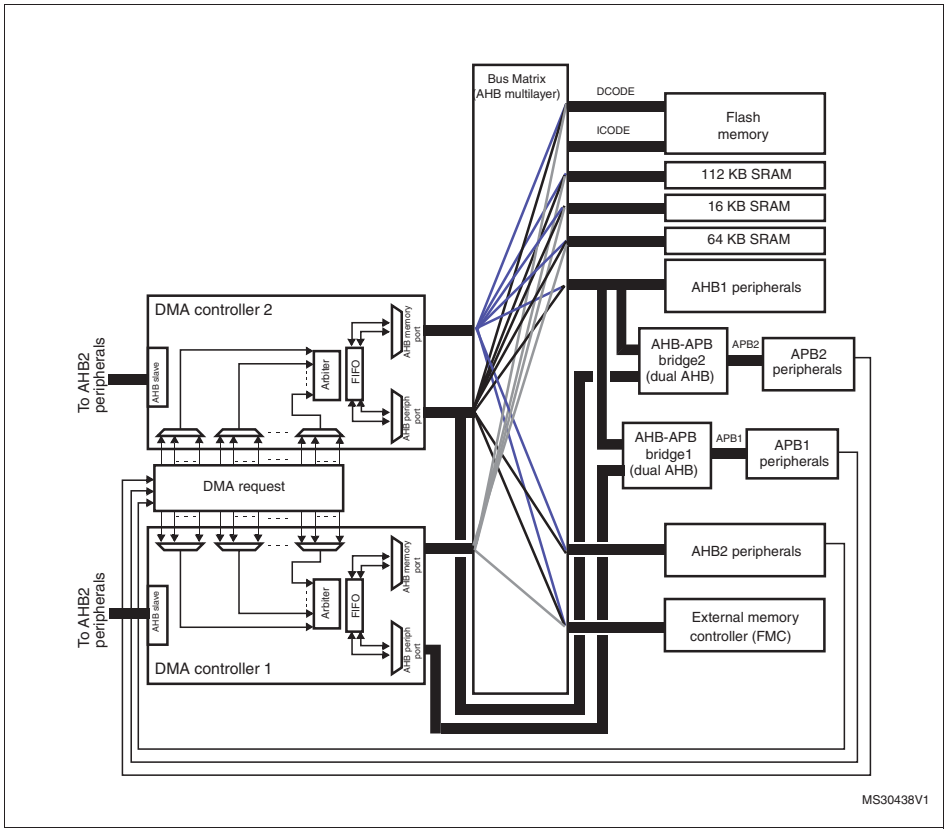
Порт ведомого AHB (с 32-бит доступом) используется для конфигурации контроллера DMA.

Рис. 32. Два контроллера DMA в STM32F405xx/07xx и STM32F415xx/17xx



1. Контроллер периферии DMA1 подключён в шинной матрице не так как DMA2. Так что передавать память->память могут только потоки DMA2.

Рис. 33. Два контроллера DMA в STM32F42xxx и STM32F43xxx



1. Контроллер периферии DMA1 подключён в шинной матрице не так как DMA2. Так что передавать память->память могут только потоки DMA2.

10.3.2. Передачи DMA

Передачи DMA это серия заданного числа чтения/записи данных. Число передаваемых данных и их ширина (8, 16 или 32-бит) программируется.

Передача DMA состоит из трёх операций:

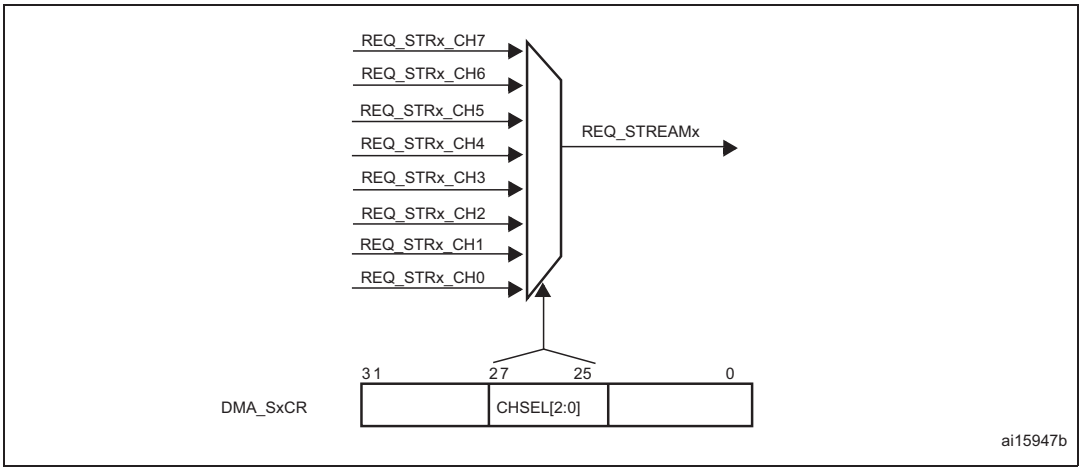
- Загрузка из регистра данных периферии или из памяти, адресуемых регистрами DMA_SxPAR и DMA_SxM0AR
- Запись загруженных данных в устройство или память, адресуемых регистрами DMA_SxPAR и DMA_SxM0AR
- Постдекремент регистра DMA_SxNDTR, содержащего число оставшихся передач

После события устройство посылает сигнал запроса к контроллеру DMA. Он обслуживает запрос в порядке приоритетов. При обращении к устройству контроллер посылает ему сигнал подтверждения (Acknowledge) и устройство снимает запрос. Контроллер в ответ снимает подтверждение. Всё, устройство может запускать следующую передачу.

10.3.3. Выбор канала

Биты CHSEL[2:0] регистра DMA_SxCR позволяют связать поток DMA с одним из 8 возможных запросов каналов.

Рис. 35. Выбор канала



8 запросов от периферии (TIM, ADC, SPI, I2C, и т.д.) независимо подключены к каждому каналу, что зависит от реализации.

Таблица 42. Картирование каналов DMA1

Запрос периферии	Поток 0	Поток 1	Поток 2	Поток 3	Поток 4	Поток 5	Поток 6	Поток 7
Канал 0	SPI3_RX		SPI3_RX	SPI2_RX	SPI2_TX	SPI3_TX		SPI3_TX
Канал 1	I2C1_RX		TIM7_UP		TIM7_UP	I2C1_RX	I2C1_TX	I2C1_TX
Канал 2	TIM4_CH1		I2S3_EXT_RX	TIM4_CH2	I2S2_EXT_TX	I2S3_EXT_TX	TIM4_UP	TIM4_CH3
Канал 3	I2S3_EXT_RX	TIM2_UP TIM2_CH3	I2C3_RX	I2S2_EXT_RX	I2C3_TX	TIM2_CH1	TIM2_CH2 TIM2_CH4	TIM2_UP TIM2_CH4
Канал 4	UART5_RX	USART3_RX	UART4_RX	USART3_TX	UART4_TX	USART2_RX	USART2_TX	UART5_TX
Канал 5	UART8_TX(1)	UART7_TX(1)	TIM3_CH4 TIM3_UP	UART7_RX(1)	TIM3_CH1 TIM3_TRIG	TIM3_CH2	UART8_RX(1)	TIM3_CH3
Канал 6	TIM5_CH3 TIM5_UP	TIM5_CH4 TIM5_TRIG	TIM5_CH1	TIM5_CH4 TIM5_TRIG	TIM5_CH2		TIM5_UP	
Канал 7		TIM6_UP	I2C2_RX	I2C2_RX	USART3_TX	DAC1	DAC2	I2C2_TX

1. Эти запросы доступны только на STM32F42xxx и STM32F43xxx.

Таблица 43. Картирование каналов DMA2

Запрос периферии	Поток 0	Поток 1	Поток 2	Поток 3	Поток 4	Поток 5	Поток 6	Поток 7
Канал 0	ADC1	SAI1_A(1)	TIM8_CH1 TIM8_CH2 TIM8_CH3	SAI1_A(1)	ADC1	SAI1_B(1)	TIM1_CH1 TIM1_CH2 TIM1_CH3	
Канал 1		DCMI	ADC2	ADC2	SAI1_B(1)	SPI6_TX(1)	SPI6_RX(1)	DCMI
Канал 2	ADC3	ADC3		SPI5_RX(1)	SPI5_TX(1)	CRYP_OUT	CRYP_IN	HASH_IN
Канал 3	SPI1_RX		SPI1_RX	SPI1_TX		SPI1_TX		
Канал 4	SPI4_RX(1)	SPI4_TX(1)	USART1_RX	SDIO		USART1_RX	SDIO	USART1_TX
Канал 5		USART6_RX	USART6_RX	SPI4_RX(1)	SPI4_TX(1)		USART6_TX	USART6_TX
Канал 6	TIM1_TRIG	TIM1_CH1	TIM1_CH2	TIM1_CH1	TIM1_CH4 TIM1_TRIG TIM1_COM	TIM1_UP	TIM1_CH3	
Канал 7		TIM8_UP	TIM8_CH1	TIM8_CH2	TIM8_CH3	SPI5_RX(1)	SPI5_TX(1)	TIM8_CH4 TIM8_TRIG TIM8_COM

1. Эти запросы доступны только на STM32F42xxx и STM32F43xxx.

10.3.4. Арбитр

Арбитр управляет запросами 8 потоков DMA на основе их приоритетов для обоих ведущих портов AHB (памяти и периферии) и запускает последовательности доступа к памяти/периферии.

Управление приоритетами идёт в две стадии:

- Программная: В регистре DMA_SxCR задаётся приоритет потока. Есть четыре уровня:
  - Очень высокий
  - Высокий
  - Средний
  - Низкий
- Аппаратная: Если программные приоритеты двух запросов равны, то выбирается первым запрос с меньшим номером.

10.3.5. Потоки DMA

Каждый из 8 потоков DMA обеспечивает одностороннюю передачу источник -> получатель.

Каждый поток может делать:

- Регулярные передачи: память->периферия, периферия->память или память->память.
- Двухбуферные передачи: они используют два указателя памяти (пока DMA работает с одним буфером, программа использует второй).

Число передаваемых данных (up to 65535) программируется и относится к устройству, запросившему передачу DMA по периферийному порту AHB. Регистр с числом оставшихся данных декрементируется после каждой передачи.

10.3.6. Источник, получатель и режимы передачи

Адреса передачи источника и получателя охватывают всю 4 GB область, от 0x0000 0000 до 0xFFFF FFFF.

Биты DIR[1:0] регистра DMA_SxCR задают направление передачи: память->периферия, периферия->память или память->память.

Таблица 44. Адреса источника и получателя

Биты DIR[1:0] в DMA_SxCR	Направление	Источник	Получатель
0	периферия->память	DMA_SxPAR	DMA_SxM0AR
1	память->периферия	DMA_SxM0AR	DMA_SxPAR
10	память->память	DMA_SxPAR	DMA_SxM0AR
11	резерв	-	-

При ширине данных (биты PSIZE или MSIZE в регистре DMA_SxCR) в полуслово или слово, адреса в регистрах DMA_SxCR или DMA_SxM0AR/M1AR должны быть выравнены на соответствующую границу.

Режим периферия->память

Если режим включён (стоит бит EN в регистре DMA_SxCR), то после появления запроса от периферии, поток запускает передачу из источника для заполнения FIFO.

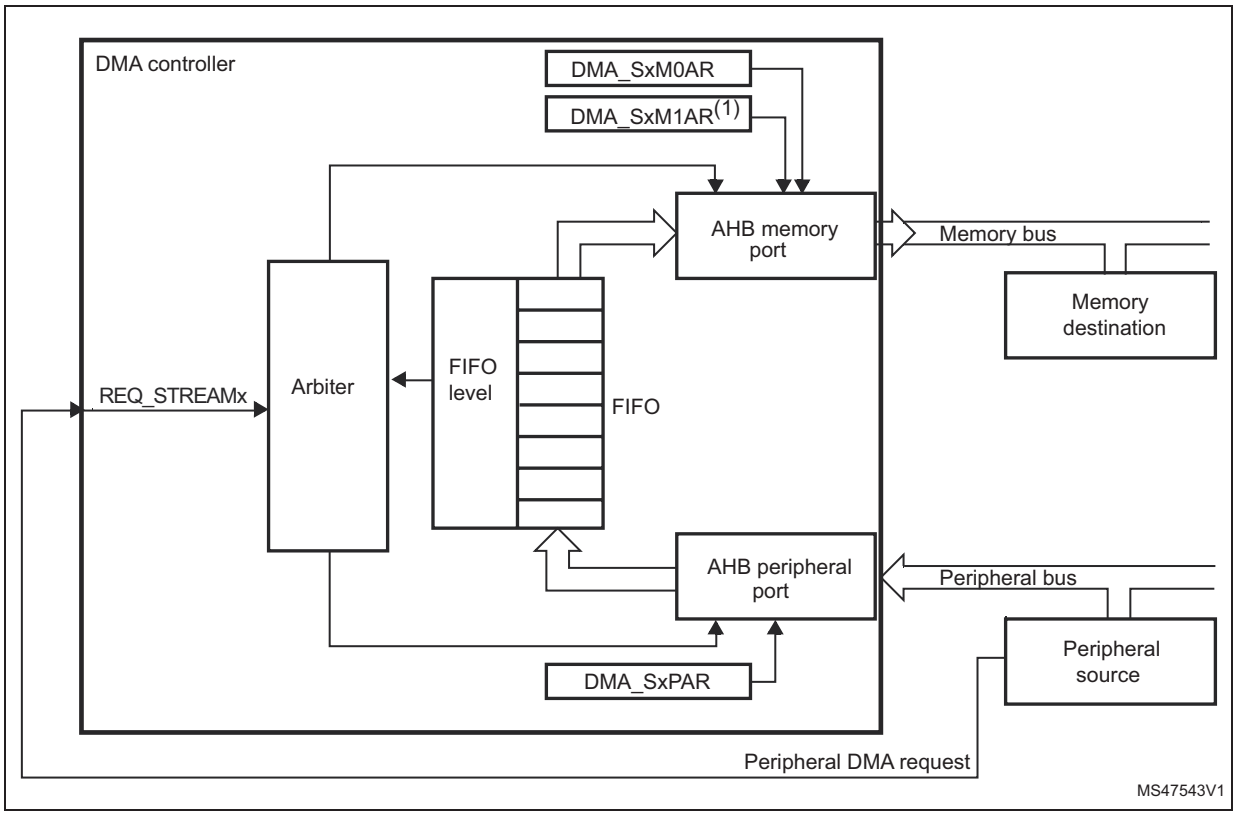
При достижении порога заполнения FIFO он сливается получателю.

Передача останавливается при достижении регистром DMA_SxNDTR нуля, когда периферия запросит конца передачи (в случае периферийного управления потоком) или при программном снятии бита EN в регистре DMA_SxCR.

В прямом режиме (бит DMDIS в регистре DMA_SxFCR равен ‘0’) уровень порога FIFO не используется: поступившее данное сразу сливается из FIFO получателю.

Поток имеет доступ к порту AHB источника или получателя только при выигрыше арбитража. Уровень приоритета пишется в биты PL[1:0] регистра DMA_SxCR.

Рис. 36. Режим периферия->память



1. Для двухбуферного режима.

Режим память->периферия

Если режим включён (стоит бит EN в регистре DMA_SxCR), то поток сразу запускает передачу из памяти для заполнения FIFO.

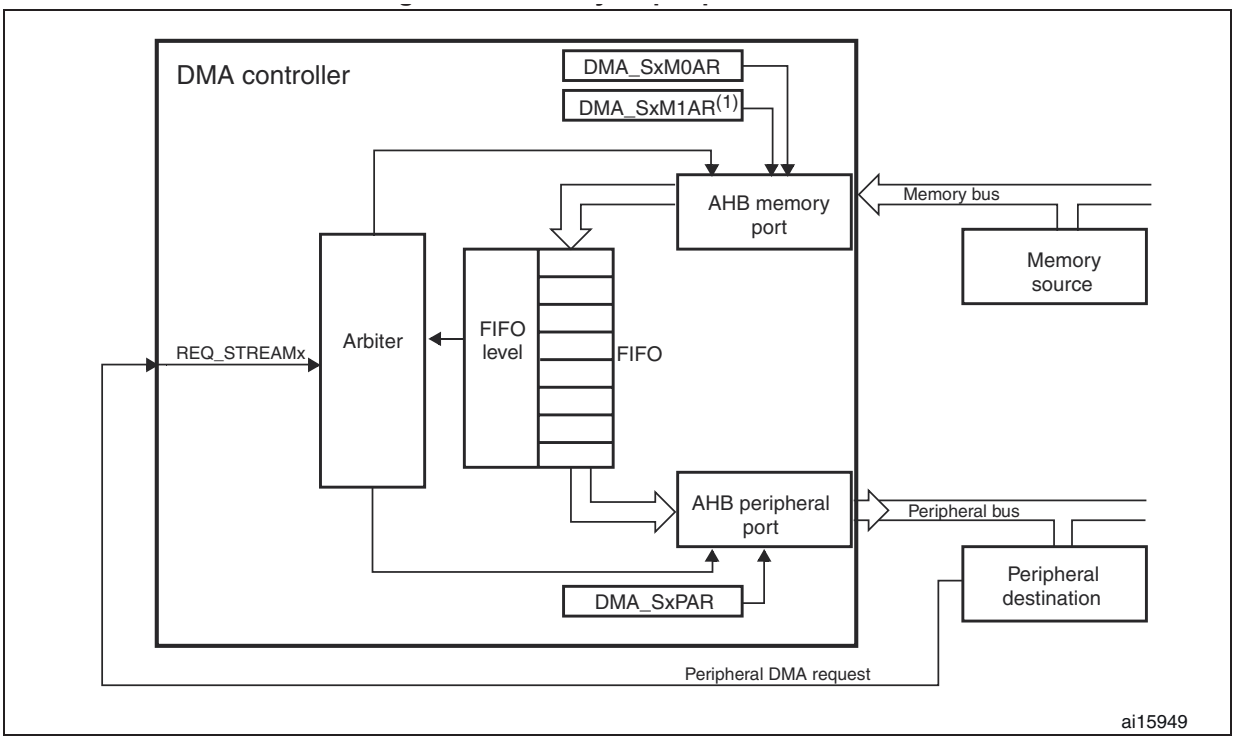
При появлении запроса периферии FIFO сливается получателю. Если уровень заполнения FIFO становится ниже уровня порога, то FIFO пополняется из памяти.

Передача останавливается при достижении регистром DMA_SxNDTR нуля, когда периферия запросит конца передачи (в случае периферийного управления потоком) или при программном снятии бита EN в регистре DMA_SxCR.

В прямом режиме (бит DMDIS в регистре DMA_SxFCR равен ‘0’) уровень порога FIFO не используется: поступившее из памяти данное сразу сливается из FIFO получателю при появлении запроса от него и берётся следующее данное из памяти. Размер данных в памяти пишут в поле PSIZE регистра DMA_SxCR.

Поток имеет доступ к порту AHB источника или получателя только при выигрыше арбитража. Уровень приоритета пишется в биты PL[1:0] регистра DMA_SxCR.

Рис. 37. Режим память->периферия



1. Для двухбуферного режима.

Режим память->память

Каналы DMA2 запускаются без запросов от периферии.

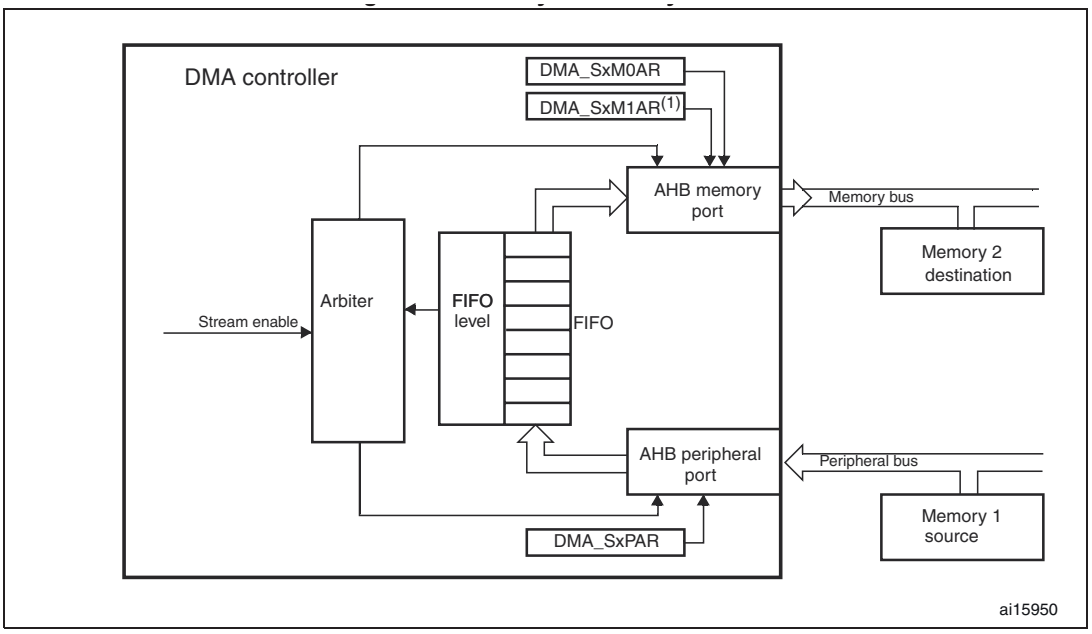
Если режим включён (стоит бит EN в регистре DMA_SxCR), то поток сразу запускает передачу из памяти для заполнения FIFO.

Передача останавливается при достижении регистром DMA_SxNDTR нуля, или при программном снятии бита EN в регистре DMA_SxCR.

Поток имеет доступ к порту AHB источника или получателя только при выигрыше арбитража. Уровень приоритета пишется в биты PL[1:0] регистра DMA_SxCR.

В этом режиме Кольцевой и прямой режимы невозможны. Эти операции доступны только DMA2.

Рис. 37. Режим память->память



1. Для двухбуферного режима.

10.3.7. Инкремент указателей

Инкремент указателей периферии и памяти разрешается битами PINC и MINC в DMA_SxCR.

Без инкремента весьма пригодно работать с одним регистром данных периферии.

Размер приращений адресов источника и получателя определяется шириной данных (биты PSIZE и MSIZE в регистре DMA_SxCR) и равен 1 (байт), 2 (полуслово) или 4 (слово).

Для оптимизации операций упаковки можно фиксировать размер приращения адреса периферийного порта. Бит PINCOS регистра DMA_SxCR приравнивает приращение адреса к размеру данных периферийного порта, или с 32-бит адресом (инкремент 4). Бит PINCOS влияет только на периферийный порт AHB.

Если бит PINCOS стоит, то инкремент периферийного порта всегда равен 4, независимо от значения PSIZE.

10.3.8. Кольцевой режим

Он позволяет использовать кольцевые буферы при обработке непрерывных потоков данных (например при скане ADC). Включается битом CIRC в регистре DMA_SxCR.

При включённом кольцевом режиме количество передаваемых данных автоматически перегружается и передача продолжается.

NB: В этом режиме в случае группового (burst) режима в памяти надо обязательно соблюдать следующее правило:

DMA_SxNDTR = Кратно ((Mburst beat) × (Msize)/(Psize)), где:

- (Mburst beat) = 4, 8 или 16 (биты MBURST в регистре DMA_SxCR register)
- ((Msize)/(Psize)) = 1, 2, 4, 1/2 или 1/4 (Msize и Psize представляют биты MSIZE и PSIZE в регистре DMA_SxCR. Они байт зависимые.)
- DMA_SxNDTR = Число передач по периферийному порту AHB t
  Например: Mburst beat = 8 (INCR8), MSIZE = ‘00’ (байт) и PSIZE = ‘01’ (полуслово), в этом случае: DMA_SxNDTR должно быть кратно (8 × 1/2 = 4).

Если этой формулы не соблюсти, то поведение DMA и целостность данных не гарантируется.

NDTR также должен быть кратным размеру группы (burst) периферии умноженному на размер данных периферии, иначе DMA поведёт себя дурно.

10.3.9. Двухбуферный режим

Он доступен для всех потоков DMA1 и DMA2 и включается установкой бита DBM в регистре DMA_SxCR. При этом используются два указателя памяти. При активном режиме автоматически включается кольцевой режим (бит CIRC в DMA_SxCR не волнует) и в конце каждой передачи указатели памяти меняются местами.

Двухбуферный режим может работать в обоих направлениях (память может быть источником и получателем). См. Таблицу 45.

NB: В этом режиме можно менять базовый адрес (DMA_SxM0AR или DMA_SxM1AR) порта памяти AHB налету, но надо соблюсти условия:

- Когда бит CT в регистре DMA_SxCR равен ‘0’ то можно писать регистр DMA_SxM1AR.
  Попытка записи при CT = '1' поставит флаг ошибки (TEIF) и поток автоматически встанет.
- Когда бит CT в регистре DMA_SxCR равен ‘1’ то можно писать регистр DMA_SxM0AR.
  Попытка записи при CT = '0' поставит флаг ошибки (TEIF) и поток автоматически встанет.

Во избежание этого менять адрес надо как только встанет флаг TCIF. В это время меняются адреса буферов.

В других режимах при разрешённых потоках регистр адреса памяти защищён от записи.

Таблица 45. Регистры адреса в двухбуферном режиме (DBM=1)

Биты DIR[1:0] регистра DMA_SxCR	Направление	Адрес источника	Адрес получателя
0	Периферия->память	DMA_SxPAR	DMA_SxM0AR / DMA_SxM1AR
1	Память->периферия	DMA_SxM0AR / DMA_SxM1AR	DMA_SxPAR
10	Не можно(1)		
11	Резерв	-	-

1. В двухбуферном режиме автоматически включается кольцевой, а при нём передачи память->память невозможны.

10.3.10.Программируемая ширина данных, упаковка/распаковка и порядок битов (endian)

Число передаваемых данных пишут в регистр DMA_SxNDTR до включения потока (кроме случая когда контроллер периферийный, стоит бит PFCTRL в DMA_SxCR).

При использовании внутреннего FIFO ширины данных источника и получателя пишут в биты PSIZE и MSIZE в регистре DMA_SxCR (может быть 8-, 16- или 32-бит).

Если PSIZE и MSIZE не равны:

- Ширина передаваемых данных равна ширине шины периферии (биты PSIZE в DMA_SxCR).
- С порядком битов (endian) расправляется контроллер DMA. См. Таблицу 46.

Остановка операции упаковки/распаковки данных до её завершения может их разрушить. Но при групповой передаче в потоке каждая группа неразделима (см. 10.3.11.).

В прямом режиме(DMDIS = 0 в DMA_SxFCR) упаковка/распаковка невозможна. В этом случае ширины данных не могут различаться (обе равны PSIZE из DMA_SxCR на MSIZE плюют свысока).

Таблица 46. Упаковка/распаковка и порядок битов (endian)

Ширина порта памяти AHB	Ширина порта периферии AHB	Число данных передачи (NDT)	Число передач памяти	Адрес порта памяти / линии байта	Число передач периф.	Адрес порта периферии / линии байта	
						PINCOS = 1	PINCOS = 0
8	8	4	1 2 3 4	0x0 / B0[7:0] 0x1 / B1[7:0] 0x2 / B2[7:0] 0x3 / B3[7:0]	1 2 3 4	0x0 / B0[7:0] 0x4 / B1[7:0] 0x8 / B2[7:0] 0xC / B3[7:0]	0x0 / B0[7:0] 0x1 / B1[7:0] 0x2 / B2[7:0] 0x3 / B3[7:0]
8	16	2	1 2 3 4	0x0 / B0[7:0] 0x1 / B1[7:0] 0x2 / B2[7:0] 0x3 / B3[7:0]	1 2	0x0 / B1|B0[15:0] 0x4 / B3|B2[15:0]	0x0 / B1|B0[15:0] 0x2 / B3|B2[15:0]
8	32	1	1 2 3 4	0x0 / B0[7:0] 0x1 / B1[7:0] 0x2 / B2[7:0] 0x3 / B3[7:0]	1	0x0 / B3|B2|B1|B0[31:0]	0x0 / B3|B2|B1|B0[31:0]
16	8	4	1 2	0x0 / B1|B0[15:0] 0x2 / B3|B2[15:0]	1 2 3 4	0x0 / B0[7:0] 0x4 / B1[7:0] 0x8 / B2[7:0] 0xC / B3[7:0]	0x0 / B0[7:0] 0x1 / B1[7:0] 0x2 / B2[7:0] 0x3 / B3[7:0]
16	16	2	1 2	0x0 / B1|B0[15:0] 0x2 / B1|B0[15:0]	1 2	0x0 / B1|B0[15:0] 0x4 / B3|B2[15:0]	0x0 / B1|B0[15:0] 0x2 / B3|B2[15:0]
16	32	1	1 2	0x0 / B1|B0[15:0] 0x2 / B3|B2[15:0]	1	0x0 / B3|B2|B1|B0[31:0]	0x0 / B3|B2|B1|B0[31:0]
32	8	4	1	0x0 / B3|B2|B1|B0[31:0]	1 2 3 4	0x0 / B0[7:0] 0x4 / B1[7:0] 0x8 / B2[7:0] 0xC / B3[7:0]	0x0 / B0[7:0] 0x1 / B1[7:0] 0x2 / B2[7:0] 0x3 / B3[7:0]
32	16	2	1	0x0 /B3|B2|B1|B0[31:0]	1 2	0x0 / B1|B0[15:0] 0x4 / B3|B2[15:0]	0x0 / B1|B0[15:0] 0x2 / B3|B2[15:0]
32	32	1	1	0x0 /B3|B2|B1|B0[31:0]	1	0x0 /B3|B2|B1|B0[31:0]	0x0 / B3|B2|B1|B0[31:0]

NB: Порт периферии может быть и источником и получателем, равно как и память <—> память.

PSIZE, MSIZE и NDT[15:0] надо ставить так чтобы последняя передача была завершена. Это может случиться если PSIZE меньше MSIZE. См. Таблицу 46.

Таблица 47. Ограничения NDT супротив PSIZE и MSIZE

PSIZE[1:0] в DMA_SxCR	MSIZE[1:0] в DMA_SxCR	NDT[15:0] в DMA_SxNDTR
00 (8-бит)	01 (16-бит)	кратно 2
00 (8-бит)	10 (32-бит)	кратно 4
01 (16-бит)	10 (32-бит)	кратно 2

10.3.11. Одинарные и групповые передачи

Контроллер DMA может генерировать одинарные или инкрементные групповые передачи по 4, 8 или 16 кусков. Размер группы задаётся раздельно для двух AHB портов битами MBURST[1:0] и PBURST[1:0] в регистре DMA_SxCR. Размер группы это число кусков в группе, а не байтов. Каждый кусок, составляющий группу неразделим: AHB блокируется и арбитр не вмешивается в групповую передачу.

Для одинарной и групповой конфигурации каждый запрос DMA запускает разное число передач порта периферии AHB:

- При одинарной передаче запрос DMA пускает передачу согласно PSIZE[1:0] в DMA_SxCR
- При групповой передаче запрос DMA пускает передачу 4,8 или 16 кусков байтов, полуслов или слов согласно PBURST[1:0] и PSIZE[1:0] в регистре DMA_SxCR.

То же самое относится и к передачам порта памяти AHB с битами MBURST и MSIZE.

В прямом режиме поток пускает только одинарные передачи и биты MBURST[1:0] и PBURST[1:0] ставятся аппаратно.

Указатели адресов (регистры DMA_SxPAR и DMA_SxM0AR) должны быть выравнены на границу размера передачи.

Размер групп не должен пересекать границу 1 Кб, это минимальный размер адресного пространства для ведомого AHB, иначе будет ошибка AHB о которой в регистрах DMA молчат.

10.3.12. FIFO

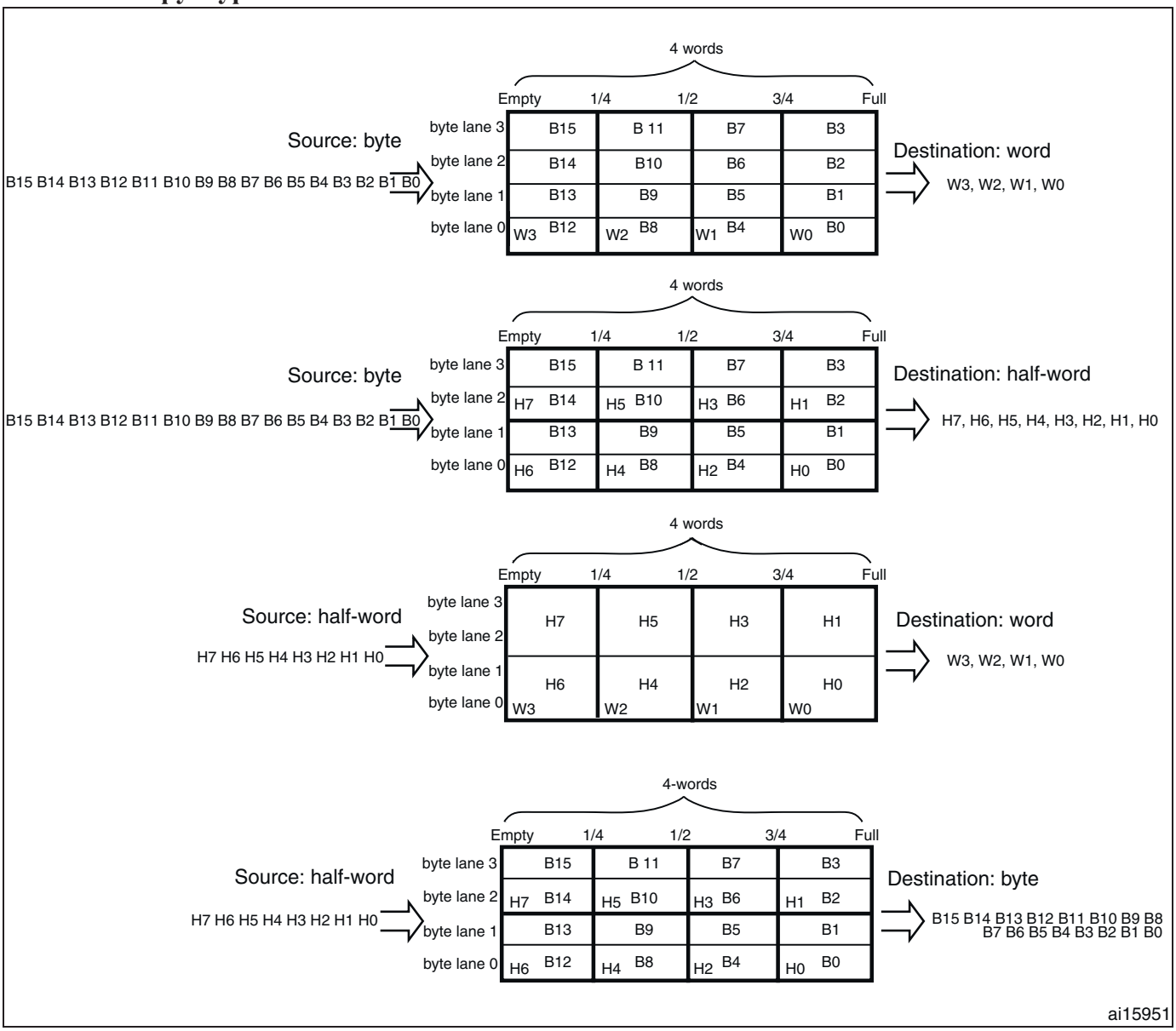
FIFO используются для временного хранения данных при передаче от источника получателю.

У каждого потока есть свой 4-слов FIFO и порог заполнения в 1/4, 1/2, 3/4 или полный размер.

Для использования порога FIFO прямой режим нужно выключить битом DMDIS в DMA_SxFCR.

Структура FIFO различается в зависимости от ширины данных источника и получателя.

Рис. 39. Структура FIFO



Порог FIFO и групповая конфигурация

Порог FIFO (биты FTH[1:0] в DMA_SxFCR) и размер группы в памяти (MBURST[1:0] в регистре DMA_SxCR) надо ставить так, чтобы содержимое, указываемое порогом FIFO, точно совпадало с целым числом групповых передач памяти, иначе выдаётся ошибка FIFO (флаг FEIFx в регистре DMA_HISR или DMA_LISR) и поток автоматически остановится.

Таблица 48. Конфигурации порога FIFO

MSIZE	FIFO level	MBURST = INCR4	MBURST = INCR8	MBURST = INCR16
Байт	1/4	1 группа из 4 кусков	запрет	запрет
	1/2	2 группы из 4 кусков	1 группа из 8 кусков	
	3/4	3 группы из 4 кусков	запрет	
	Полный	4 группы из 4 кусков	2 группы из 8 кусков	1 группа из 16 кусков
Полуслово	1/4	запрет	запрет	запрет
	1/2	1 группа из 4 кусков		
	3/4	запрет		
	Полный	2 группы из 4 кусков	1 группа из 8 кусков	
Слово	1/4	запрет	запрет	
	1/2			
	3/4			
	Полный	1 группа из 4 кусков		

В любом случае размер группы, умноженный на размер данного (1 (байт), 2 (полуслово) или 4 (слово)), не должен превышать размер FIFO.

Незавершённая группа в конце передачи DMA может случиться если:

- Для порта периферии AHB: общее число данных (в регистре DMA_SxNDTR) не кратно размеру группы, умноженному на размер данного
- Для порта памяти AHB: число оставшихся данных в FIFO не кратно размеру группы, умноженному на размер данного

В таких случаях остаток данных передаётся в одинарном режиме не смотря на запрос от потока.

При (PBURST × PSIZE) = FIFO_SIZE (4 слова) с PSIZE = 1, 2 или 4 и PBURST = 4, 8 или 16 порог FIFO = 3/4 запрещён.

Слив FIFO.

FIFO можно слить снятием бита EN в регистре DMA_SxCR и при передачах периферия->память и память->память: Если при выключении потока в FIFO остаются данные, то они сливаются получателю и ставится бит завершения (TCIFx) в регистре DMA_LISR или DMA_HISR.

Остаток счётчика данных DMA_SxNDTR в этом случае сохраняется для определения свободного места в памяти получателя.

Если во время слива FIFO число оставшихся данных меньше ширины данных памяти, то всё равно посылается MSIZE данных, возможно и нежелательных. Плохие байты можно определить по регистру DMA_SxNDTR.

Если число оставшихся данных в FIFO меньше размера группы, то слив выполняется в одинарном режиме.

Прямой режим

По умолчанию FIFO работает в прямом режиме (бит DMDIS в регистре DMA_SxFCR снят) и порог FIFO не используется. При передачах память->периферия в FIFO грузится одно данное, которое выдаётся периферии по его запросу.

Во избежание насыщения FIFO потоку надо давать высокий приоритет

Режим имеет ограничения:

- Ширины передачи источника и получателя равны и заданы битами PSIZE[1:0] в DMA_SxCR (биты MSIZE[1:0] не работают)
- Групповые передачи невозможны (PBURST[1:0] и MBURST[1:0] в DMA_SxCR не нужны)

Прямого режима нет для передач память->память.

10.3.13. Завершение передач DMA

По концу передачи бит TCIFx в регистрах DMA_LISR или DMA_HISR ставится когда:

- В режиме контроллера потока DMA:
  - Счётчик DMA_SxNDTR достиг нуля при передаче память->периферия
  - Поток выключен до конца передачи (снятием бита EN в регистре DMA_SxCR) или (при передачах периферия->память и память->память) и остаток данных слит из FIFO в память
- В режиме контроллера потока Периферии:
  - От периферии получена последняя группа или одинарный запрос и (при передачах периферия->память) остаток данных слит из FIFO в память
  - Поток выключен программой, и (при передачах периферия->память) остаток данных слит из FIFO в память

Завершение передачи зависит от остатка данных в FIFO только при передачах периферия->память.

В не-кольцевом режиме после конца передачи (данные закончились) DMA стопорится (аппаратно снимается бит EN в регистре DMA_SxCR) и запросы DMA не обслуживаются вплоть до повторного включения установкой бита EN в регистре DMA_SxCR.

10.3.14. Приостановка передач DMA

Передачу DMA можно в любой момент приостановить, потом возобновить или выключить.

Есть два случая:

- Поток выключает передачу насовсем. Нужно только выключить поток снятием бита EN в регистре DMA_SxCR. Это может занять время на завершение передачи. И вот ставится флаг

завершения передачи (TCIF в DMA_LISR или DMA_HISR) и снимается бит EN. Регистр DMA_SxNDTR содержит число недопереданных данных.

- Поток приостанавливается до обнуления регистра DMA_SxNDTR в надежде возобновить передачу. Чтобы пустить передачу с прерванного места надо бы сначала остановить поток снятием бита EN в регистре DMA_SxCR, дождаться его обнуления, прочитать остаток передачи в регистре DMA_SxNDTR и затем когда надо:
  - Подправить адреса периферии и/или памяти
  - В регистр DMA_SxNDTR поставить прочитанный там же остаток передачи
  - Запустить поток снова

NB: При прерывании передачи флаг окончания (TCIF в DMA_LISR или DMA_HISR) всё равно ставится.

10.3.15. Контроллер хода передачи

Штуковина, управляющая числом передаваемых данных, конфигурируется независимо для каждого потока битом PFCTRL в регистре DMA_SxCR.

Она может быть:

- Контроллером DMA: в этом случае число данных пишут в регистр DMA_SxNDTR до пуска потока DMA.
- Источником или получателем периферии: в этом случае число будущей передачи заранее неизвестно. О передаче последнего данного периферия извещает контроллер DMA аппаратно. Но на такое способно только – SDIO

Когда периферийный контроллер работает в данном режиме число в регистре DMA_SxNDTR на передачи DMA не влияет. После пуска потока туда аппаратно пишется 0xFFFF, дабы соблюсти следующие схемы:

- Ожидаемое прерывание потока: бит EN в DMA_SxCR снимается программно перед аппаратным сигналом последнего данного от периферии (одинарным или групповым). Здесь поток выключается и при передаче периферия->память запускается слив FIFO. Флаг TCIFx соответствующего потока вздымается. Число переданных данных находят так:
  – Число_переданных_данных = 0xFFFF – DMA_SxNDTR
- Нормальное прерывание потока по приёму аппаратного сигнала последнего данного: поток стопорится сам, ставится флаг TCIFx, число переданных данных ищут как и ранее.
- Регистр DMA_SxNDTR достигает 0: ставится флаг TCIFx соответствующего потока. Поток выключается сам, даже если не был принят аппаратный сигнал последнего данного от периферии (одинарным или групповым). Принятые данные не теряются. То есть даже при периферийном контроллере может быть до 65535 данных в одной передаче.

При передачах память->память контроллер всегда DMA и бит PFCTRL снимается аппаратно.

При периферийном контроллере кольцевой режим запрещён.

10.3.16. Возможные конфигурации DMA

Таблица 49. Возможные конфигурации DMA

Режим передачи DMA	Источник	Получатель	Контроллер	Кольцевой режим	Тип передачи	Прямой режим	Два буфера
Периферия-память	Периф. порт AHB	Порт памяти AHB	DMA	Можно	Одинарн.	Можно	Можно
					Группа	Запрет	
			Периферия	Запрет	Одинарн.	Можно	Запрет
					Группа	Запрет	
Память-периферия	Порт памяти AHB	Периф. порт AHB	DMA	Можно	Одинарн.	Можно	possible
					Группа	Запрет	
			Периферия	Запрет	Одинарн.	Можно	Запрет
					Группа	Запрет	
Память-память	Периф. порт AHB	Порт памяти AHB	Только DMA	Запрет	Одинарн.	Запрет	Запрет
					Группа		

10.3.17. Процедура конфигурации потока

Вот такая она:

1. Если поток включён, то выключаем его снятием бита EN в регистре DMA_SxCR и ждём его обнуления. Перед запуском потока биты в регистрах состояния (DMA_LISR и DMA_HISR) от предыдущих передач DMA должны быть чистыми.
2. Ставим адрес периферии в регистре DMA_SxPAR.
3. Ставим адрес памяти в регистре DMA_SxMA0R (и DMA_SxMA1R при двух буферах).
4. Пишем общее число передаваемых данных в регистр DMA_SxNDTR. После каждого события периферии или куска группы он декрементируется.
5. Битами CHSEL[2:0] в регистре DMA_SxCR выбираем канал (запрос) DMA.
6. Если нужен периферийный контроллер, то ставим бит PFCTRL в регистре DMA_SxCR.
7. В битах PL[1:0] регистра DMA_SxCR пишем приоритет потока.
8. Определяем параметры FIFO (вкл/выкл, пороги приёма и передачи)
9. Пишем направление передачи, инкремент периферии и памяти, одинарные/групповые, ширины данных, кольцевой режим, двухбуферный режим и прерывания после половины или всей передачи, ошибок в регистр DMA_SxCR.
10. Пускаем поток установкой бита EN в регистре DMA_SxCR.

Теперь поток может работать с подключенной к нему периферией.

После передачи получателю половины запланированных данных ставится флаг HTIF и выдаётся прерывание половины передачи если стоит бит разрешения (HTIE).

После передачи получателю всех запланированных данных ставится флаг TCIF и выдаётся прерывание половины передачи если стоит бит разрешения (TCIE).

Внимание: Для выключения присоединённой к потоку периферии надо сначала остановить поток битом EN и дождаться его обнуления. Теперь можно выключать устройство.

10.3.18. Обработка ошибок

Контроллер DMA может обнаруживать следующие ошибки:

- Ошибка передачи: флаг прерывания ошибки передачи (TEIFx) ставится когда:
  - При чтении или записи DMA появляется ошибка шины
  - В двухбуферном режиме появилась программная запись в буфер, в настоящий момент занятый работающим потоком DMA
- Ошибка FIFO: флаг прерывания ошибки FIFO (FEIFx) ставится когда:
  - Появилось исчерпание FIFO
  - Появилось переполнение FIFO (кроме режима память->память)
  - Поток включён с порогом FIFO, несовместимым с размером группы памяти
- Ошибка прямого режима: флаг прерывания ошибки прямого режима (DMEIFx) ставится только при передачах периферия->память в прямом режиме при снятом бите MINC в регистре DMA_SxCR. Флаг ставится при появлении запроса DMA и необработанным предыдущим данным (шину памяти не предоставили). То есть, 2 порции данных переданы по одному адресу, а вдруг получатель не умеет с этим разбираться.

В прямом режиме флаг ошибки FIFO может встать если:

- В режиме периферия->память при исчерпании/переполнении FIFO из-за не предоставленной шины памяти для нескольких запросов
- В режиме память->периферия возникло исчерпание из-за не предоставленной шины памяти до появления запроса периферии

Если флаги TEIFx или FEIFx встают из-за несовместимости размера группы и порога FIFO, то сбойный поток стопорится аппаратным снятием бита EN в регистре DMA_SxCR.

Если флаги DMEIFx или FEIFx встают из-за переполнения или исчерпания, то сбойный поток не стопорится. Решайте сами, что с ним делать, данные не потеряны.

Прерывания по флагам ошибки (TEIF, FEIF, DMEIF) в регистрах DMA_LISR или DMA_HISR разрешаются своими битами (TEIE, FEIE, DMIE) в регистрах DMA_SxCR или DMA_SxFCR.

При переполнении и исчерпании FIFO данные не теряются, поскольку периферия не получает подтверждения на запрос пока условия не будут очищены. Если это продлится слишком долго, то и периферия может поймать переполнение или исчерпание своего внутреннего буфера и данные могут потеряться.

10.4. Прерывания DMA

Таблица 50. Запросы прерываний DMA

Событие прерывания	Флаг события	Бит разрешения
Половина передачи	HTIF	HTIE
Полная передача	TCIF	TCIE
Ошибка передачи	TEIF	TEIE
Переполнение/Исчерпание FIFO	FEIF	FEIE
Ошибка прямого режима	DMEIF	DMEIE

Бит разрешения надо ставить при снятом соответствующем флаге, иначе вас прервут немедленно.

10.5. Регистры DMA

Доступны только словами.

10.5.1. Младший регистр состояния (DMA_LISR)

Смещение адреса: 0x00

По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				TCIF3	HTIF3	TEIF3	DMEIF3	Reserved	FEIF3	TCIF2	HTIF2	TEIF2	DMEIF2	Reserved	FEIF2
r	r	r	r	r	r	r	r		r	r	r	r	r		r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				TCIF1	HTIF1	TEIF1	DMEIF1	Reserved	FEIF1	TCIF0	HTIF0	TEIF0	DMEIF0	Reserved	FEIF0
r	r	r	r	r	r	r	r		r	r	r	r	r		r

— Биты 31:28, 15:12 Резерв, не трогать.

— Биты 27, 21, 11, 5 **TCIFx**: Флаг прерывания конца передачи канала **x (x=3..0)**

Ставится аппаратно. Снимают записью 1 в регистр DMA_LIFCR.

0: Ещё не кончилось

1: Конец есть

— Биты 26,20,10,4 **HTIFx**: Флаг прерывания половины передачи канала **x (x=3..0)**

Ставится аппаратно. Снимают записью 1 в регистр DMA_LIFCR.

0: Ещё не половина

1: Половина есть

— Биты 25,19,9,3 **TEIFx**: Флаг прерывания ошибки передачи канала **x (x=3..0)**

Ставится аппаратно. Снимают записью 1 в регистр DMA_LIFCR.

0: Ошибки нет

1: Ошибка есть

— Биты 24,18,8,2 **DMEIFx**: Флаг прерывания ошибки прямого режима канала **x (x=3..0)**

Ставится аппаратно. Снимают записью 1 в регистр DMA_LIFCR.

0: Ошибки нет

1: Ошибка есть

— Биты 23,17,7,1 Резерв, не трогать.

— Биты 22,16,6,0 **FEIFx**: Флаг прерывания ошибки FIFO канала **x (x=3..0)**

Ставится аппаратно. Снимают записью 1 в регистр DMA_LIFCR.

0: Ошибки нет

1: Ошибка есть

10.5.2. Старший регистр состояния (DMA_HISR)

Смещение адреса: 0x04

По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				TCIF7	HTIF7	TEIF7	DMEIF7	Reserved	FEIF7	TCIF6	HTIF6	TEIF6	DMEIF6	Reserved	FEIF6
				r	r	r	r		r	r	r	r	r		r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				TCIF5	HTIF5	TEIF5	DMEIF5	Reserved	FEIF5	TCIF4	HTIF4	TEIF4	DMEIF4	Reserved	FEIF4
				r	r	r	r		r	r	r	r	r		r

— Биты 31:28, 15:12 Резерв, не трогать.

— Биты 27, 21, 11, 5 **TCIFx**: Флаг прерывания конца передачи канала **x (x=7..4)**

Ставится аппаратно. Снимают записью 1 в регистр DMA_HIFCR.

0: Ещё не кончилось

1: Конец есть

— Биты 26,20,10,4 **HTIFx**: Флаг прерывания половины передачи канала **x (x=7..4)**

Ставится аппаратно. Снимают записью 1 в свой регистр DMA_HIFCR.

0: Ещё не половина

1: Половина есть

— Биты 25,19,9,3 **TEIFx**: Флаг прерывания ошибки передачи канала **x (x=7..4)**

Ставится аппаратно. Снимают записью 1 в свой регистр DMA_HIFCR.

0: Ошибки нет

1: Ошибка есть

— Биты 24,18,8,2 **DMEIFx**: Флаг прерывания ошибки прямого режима канала **x (x=7..4)**

Ставится аппаратно. Снимают записью 1 в свой регистр DMA_HIFCR.

0: Ошибки нет

1: Ошибка есть

— Биты 23,17,7,1 Резерв, не трогать.

— Биты 22,16,6,0 **FEIFx**: Флаг прерывания ошибки FIFO канала **x (x=7..4)**

Ставится аппаратно. Снимают записью 1 в свой регистр DMA_HIFCR.

0: Ошибки нет

1: Ошибка есть

10.5.3. Младший регистр очистки флагов прерываний (DMA_LIFCR)

Смещение адреса: 0x08

По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				CTCIF3	CHTIF3	CTEIF3	CDMEIF3	Reserved	CFEIF3	CTCIF2	CHTIF2	CTEIF2	CDMEIF2	Reserved	CFEIF2
				w	w	w	w		w	w	w	w	w		w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				CTCIF1	CHTIF1	CTEIF1	CDMEIF1	Reserved	CFEIF1	CTCIF0	CHTIF0	CTEIF0	CDMEIF0	Reserved	CFEIF0
				w	w	w	w		w	w	w	w	w		w

Запись 1 чистит свой флаг в регистре DMA_LISR.

— Биты 31:28, 15:12 Резерв, не трогать.

— Биты 27, 21, 11, 5 **CTCIFx**: Флаг прерывания конца передачи канала **x (x=3..0)**

— Биты 26,20,10,4 **CHTIFx**: Флаг прерывания половины передачи канала **x (x=3..0)**

— Биты 25,19,9,3 **CTEIFx**: Флаг прерывания ошибки передачи канала **x (x=3..0)**

— Биты 24,18,8,2 **CDMEIFx**: Флаг прерывания ошибки прямого режима канала **x (x=3..0)**

— Биты 23,17,7,1 Резерв, не трогать.

— Биты 22,16,6,0 **CFEIFx**: Флаг прерывания ошибки FIFO канала **x (x=3..0)**

10.5.4. Старший регистр очистки флагов прерываний (DMA_HIFCR)

Смещение адреса: 0x0C

По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				TCIF7	HTIF7	TEIF7	DMEIF7	Reserved	FEIF7	TCIF6	HTIF6	TEIF6	DMEIF6	Reserved	FEIF6
				r	r	r	r		r	r	r	r	r		r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				TCIF5	HTIF5	TEIF5	DMEIF5	Reserved	FEIF5	TCIF4	HTIF4	TEIF4	DMEIF4	Reserved	FEIF4
				r	r	r	r		r	r	r	r	r		r

Запись 1 чистит свой флаг в регистре DMA_LISR.

— Биты 31:28, 15:12 Резерв, не трогать.
— Биты 27, 21, 11, 5 **CTCIFx**: Флаг прерывания конца передачи канала **x (x=7..4)**
— Биты 26,20,10,4 **CHTIFx**: Флаг прерывания половины передачи канала **x (x=7..4)**
— Биты 25,19,9,3 **CTEIFx**: Флаг прерывания ошибки передачи канала **x (x=7..4)**
— Биты 24,18,8,2 **CDMEIFx**: Флаг прерывания ошибки прямого режима канала **x (x=7..4)**
— Биты 23,17,7,1 Резерв, не трогать.
— Биты 22,16,6,0 **CFEIFx**: Флаг прерывания ошибки FIFO канала **x (x=7..4)**

10.5.5. Регистр конфигурации потока x (DMA_SxCR) (x = 0..7)

Конфигурация потока **x**.

Смещение адреса: 0x10 + 0x18 x *номер потока*

По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				CHSEL[2:0]			MBURST [1:0]		PBURST[1:0]		Reserved	CT	DBM	PL[1:0]	
				rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:28 Резерв, не трогать.
— Биты 27:25 **CHSEL[2:0]**: Выбор канала (0-7)
Читаются и пишутся программно. Биты защищены, пишут только при EN=0
— Биты 24:23 **MBURST[1:0]**: Конфигурация групповой передачи памяти
Читаются и пишутся программно. Биты защищены, пишут только при EN=0
00: однократная
01: INCR4 (инкрементная группа по 4 куска)
10: INCR8 (инкрементная группа по 8 кусков)
11: INCR16 (инкрементная группа по 16 кусков)
В прямом режиме аппаратно ставятся в 0x0 после установки EN=1.
— Биты 22:21 **PBURST[1:0]**: Конфигурация групповой передачи периферии
Читаются и пишутся программно. Биты защищены, пишут только при EN=0
00: однократная
01: INCR4 (инкрементная группа по 4 куска)
10: INCR8 (инкрементная группа по 8 кусков)
11: INCR16 (инкрементная группа по 16 кусков)
В прямом режиме аппаратно ставятся в 0x0.
— Бит 20 Резерв, не трогать.
— Бит 19 **CT**: Текущий буфер (только в двухбуферном режиме)
Переключается аппаратно, можно писать программно при EN=0.
0: Адрес буфера в регистре DMA_SxM0AR
1: Адрес буфера в регистре DMA_SxM1AR
— Бит 18 **DBM**: Двухбуферный режим
Читается и пишется программно. Бит защищён, пишут только при EN=0
0: Один буфер
1: Два буфера
— Биты 17:16 **PL[1:0]**: Уровень приоритета
Читаются и пишутся программно. Биты защищены, пишут только при EN=0.

00: Низкий
01: Средний
10: Высокий
11: Очень высокий

— Бит 15 **PINCOS**: Размер смещения инкремента периферии

Читается и пишется программно. Бит защищён, пишут только при EN=0.

Бит не работает при PINC = '0'. Бит аппаратно обнуляется при EN = '1' в прямом режиме или PBURST не равно “00”.

0: Размер смещения адреса периферии привязан к PSIZE
1: Размер смещения адреса периферии равно 4 (32-бит выравнивание).

— Биты 14:13 **MSIZE[1:0]:** Размер данных памяти

Читается и пишется программно. Бит защищён, пишут только при EN=0.

В прямом режиме аппаратно пишется из PSIZE после записи EN = '1'.

00: байт (8-bit)
01: полуслово (16-bit)
10: слово (32-bit)
11: резерв

— Биты 12:11 **PSIZE[1:0]**: Размер данных периферии

Читается и пишется программно. Бит защищён, пишут только при EN=0.

00: байт (8-bit)
01: полуслово (16-bit)
10: слово (32-bit)
11: резерв

— Бит 10 **MINC**: Режим инкремента адреса памяти

Читается и пишется программно. Бит защищён, пишут только при EN=0.

0: Адрес фиксирован
1: После передачи данных адрес инкрементируется в соответствии с MSIZE.

— Бит 9 **PINC**: Режим инкремента адреса периферии

Читается и пишется программно. Бит защищён, пишут только при EN=0.

0: Адрес фиксирован
1: После передачи данных адрес инкрементируется в соответствии с PSIZE.

— Бит 8 **CIRC**: Кольцевой режим

Читается и пишется программно, может сниматься аппаратно. Бит защищён, пишут только при EN=0.

0: Выкл.
1: Вкл.

Если ходом передачи управляет периферия (бит PFCTRL=1) и поток включён (бит EN=1), то этот бит аппаратно обнуляется 0.

Он аппаратно ставится в 1 при стоящем бите DBM сразу после пуска потока (бит EN ='1').

— Биты 7:6 **DIR[1:0]**: Направление передачи данных

Читаются и пишутся программно. Биты защищены, пишут только при EN=0.

00: Периферия->память
01: Память->периферия
10: Память->память

— Бит 5 **PFCTRL**: Периферийный контроллер хода передачи

Читается и пишется программно. Бит защищён, пишут только при EN=0.

0: Контроллер DMA
1: Периферийный контроллер

В режиме память=>память этот бит аппаратно обнуляется.

— Бит 4 **TCIE**: Разрешение прерывания конца передачи

Читается и пишется программно.

0: Нельзя
1: Можно

— Бит 3 **HTIE**: Разрешение прерывания половины передачи

Читается и пишется программно.

0: Нельзя
1: Можно

— Бит 2 **TEIE**: Разрешение прерывания ошибки передачи

Читается и пишется программно.

0: Нельзя

1: Можно

— Бит 2 **DMEIE**: Разрешение прерывания ошибки прямого режима

Читается и пишется программно.

0: Нельзя

1: Можно

— Бит 1 **DMEIE**: Разрешение прерывания ошибки прямого режима

Читается и пишется программно.

0: Нельзя

1: Можно

— Бит 0 **EN**: Включение потока / флаг готовности конфигурации потока (читается как 0)

Читается и пишется программно.

0: Выкл.

1: Вкл.

Может сниматься аппаратно:

– по концу передачи DMA (поток можно конфигурировать)

– при ошибке ведущего шин AHB

– при несовместимости порога FIFO порта памяти AHB и размера группы

NB: Перед запуском потока установкой бита EN в '1' надо очистить флаги событий этого потока в регистре DMA_LISR или DMA_HISR.

10.5.6. Регистр числа данных потока x (DMA_SxNDTR) (x = 0..7)

Смещение адреса: 0x14 + 0x18 x *номер потока*

По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NDT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:16 Резерв, не трогать.

— Биты 15:0 **NDT[15:0]**: Число данных передачи (0 до 65535).

Пишется только при выключенном потоке. Чтение у работающего потока даёт число оставшихся передач. Декрементируется после каждой передачи DMA.

По концу передачи регистр может оставаться обнулённым или автоматически перегрузиться предыдущим значением если:

– поток работает в кольцевом режиме.

– поток запускается снова установкой бита EN в '1'

При обнулённом регистре установка бита EN в '1' передачи не запускает.

10.5.7. Регистр адреса периферии потока x (DMA_SxPAR) (x = 0..7)

Смещение адреса: 0x18 + 0x18 x *номер потока*

По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PAR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PAR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:0 **PAR[31:0]**: Базовый адрес чтения/записи периферии.

Читаются и пишутся программно. Биты защищены, пишут только при EN=0

10.5.8. Регистр адреса памяти 0 потока x (DMA_SxM0AR) (x = 0..7)

Смещение адреса: 0x1C + 0x18 x *номер потока*

По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M0A[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M0A[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:0 **M0A[31:0]**: Базовый адрес чтения/записи памяти 0.
Читаются и пишутся программно. Биты защищены, пишут только если:
– поток выключен (бит EN= '0') или
– поток включён (бит EN=’1’) и CT = '1' (в двухбуферном режиме).

10.5.9. Регистр адреса памяти 1 потока x (DMA_SxM1AR) (x = 0..7)

Смещение адреса: 0x20 + 0x18 x *номер потока*
По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M1A[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M1A[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:0 **M1A[31:0]**: Базовый адрес чтения/записи памяти 1.
Читаются и пишутся программно. Биты защищены, пишут только если:
– поток выключен (бит EN= '0') или
– поток включён (бит EN=’1’) и CT = '0' (в двухбуферном режиме).

10.5.10. Регистр управления FIFO потока x (DMA_SxFCR) (x = 0..7)

Смещение адреса: 0x24 + 0x24 x *номер потока*
По сбросу: 0x0000 0021.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								FEIE	Reserved	FS[2:0]			DMDIS	FTH[1:0]	
								rw		r	r	r	rw	rw	rw

— Биты 31:8 Резерв, не трогать.
— Бит 7 **FEIE**: Разрешение прерывания ошибки FIFO
Читается и пишется программно.
0: Нельзя
1: Можно
— Бит 6 Резерв, не трогать.
— Биты 5:3 **FS[2:0]**: Статус FIFO
Только чтение.
000: 0 < уровень_fifo < 1/4
001: 1/4 ≤ уровень_fifo < 1/2
010: 1/2 ≤ уровень_fifo < 3/4
011: 3/4 ≤ уровень_fifo < полный
100: FIFO пуст
101: FIFO полон
иное: бестолковщина
В прямом режиме (бит DMDIS=0) бессмыслица.
— Бит 2 **DMDIS**: Выключение прямого режима
Читается и пишется программно при EN = '0’. Может ставиться аппаратно.
0: Прямой режим включён
1: Прямой режим выключен

Аппаратно ставится в режиме память->память и EN = ‘1’.

— Биты 1:0 **FTH[1:0]**: Уровень порога FIFO

Читается и пишется программно при EN = '0’.

00: 1/4 FIFO

01: 1/2 FIFO

10: 3/4 FIFO

11: весь FIFO

В прямом режиме не используется.

10.5.11. Карта регистров DMA

Рис. 51. Карта регистров DMA

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	**DMA_LISR**	Reserved				TCIF3	HTIF3	TEIF3	DMEIF3	Reserved	FEIF3	TCIF2	HTIF2	TEIF2	DMEIF2	Reserved	FEIF2	Reserved				TCIF1	HTIF1	TEIF1	DMEIF1	Reserved	FEIF1	TCIF0	HTIF0	TEIF0	DMEIF0	Reserved	FEIF0
	Reset value					0	0	0	0		0	0	0	0	0		0					0	0	0	0		0	0	0	0	0		0
0x0004	**DMA_HISR**	Reserved				TCIF7	HTIF7	TEIF7	DMEIF7	Reserved	FEIF7	TCIF6	HTIF6	TEIF6	DMEIF6	Reserved	FEIF6	Reserved				TCIF5	HTIF5	TEIF5	DMEIF5	Reserved	FEIF5	TCIF4	HTIF4	TEIF4	DMEIF4	Reserved	FEIF4
	Reset value					0	0	0	0		0	0	0	0	0		0					0	0	0	0		0	0	0	0	0		0
0x0008	**DMA_LIFCR**	Reserved				CTCIF3	CHTIF3	TEIF3	CDMEIF3	Reserved	CFEIF3	CTCIF2	CHTIF2	CTEIF2	CDMEIF2	Reserved	CFEIF2	Reserved				CTCIF1	CHTIF1	CTEIF1	CDMEIF1	Reserved	CFEIF1	CTCIF0	CHTIF0	CTEIF0	CDMEIF0	Reserved	CFEIF0
	Reset value					0	0	0	0		0	0	0	0	0	Reserved	0					0	0	0	0	Reserved	0	0	0	0	0	Reserved	0
0x000C	**DMA_HIFCR**	Reserved				CTCIF7	CHTIF7	CTEIF7	CDMEIF7	Reserved	CFEIF7	CTCIF6	CHTIF6	CTEIF6	CDMEIF6		CFEIF6	Reserved				CTCIF5	CHTIF5	CTEIF5	CDMEIF5		CFEIF5	CTCIF4	CHTIF4	CTEIF4	CDMEIF4		CFEIF4
	Reset value					0	0	0	0		0	0	0	0	0		0					0	0	0	0		0	0	0	0	0		0
0x0010	**DMA_S0CR**	Reserved				CHSEL[2:0]			MBURST[1:0]		PBURST[1:0]		Reserved	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value					0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0014	**DMA_S0NDTR**	Reserved																NDT[15:.]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0018	**DMA_S0PAR**	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x001C	**DMA_S0M0AR**	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0020	**DMA_S0M1AR**	M1A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0024	**DMA_S0FCR**	Reserved																								FEIE	Reserved	FS[2:0]			DMDIS	FTH [1:0]	
	Reset value																									0		1	0	0	0	0	1
0x0028	**DMA_S1CR**	Reserved				CHSEL [2:0]			MBURST[1:]		PBURST[1:0]		Reserved	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value					0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x002C	**DMA_S1NDTR**	Reserved																NDT[15:.]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0030	**DMA_S1PAR**	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0034	**DMA_S1M0A**R	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0038	**DMA_S1M1AR**	M1A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x003C	**DMA_S1FCR**	Reserved																								FEIE	Reserved	FS[2:0]			DMDIS	FTH [1:0]	
	Reset value																									0		1	0	0	0	0	1
0x0040	**DMA_S2CR**	Reserved				CHSEL [2:0]			MBURST[1:0]		PBURST[1:0]		Reserved	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR [1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value					0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0044	**DMA_S2NDTR**	Reserved																NDT[15:.]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0048	**DMA_S2PAR**	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004C	**DMA_S2M0AR**	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0050	**DMA_S2M1AR**	M1A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0054	**DMA_S2FCR**	Reserved																								FEIE	Reserved	FS[2:0]			DMDIS	FTH [1:0]	
	Reset value																									0		1	0	0	0	0	1
0x0058	**DMA_S3CR**	Reserved				CHSEL[2:0]			MBURST[1:0]		PBURST[1:0]		Reserved	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value					0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x005C	**DMA_S3NDTR**	Reserved																NDT[15:.]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0060	**DMA_S3PAR**	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0064	**DMA_S3M0AR**	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0068	**DMA_S3M1AR**	M1A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x006C	**DMA_S3FCR**	Reserved																								FEIE	Reserved	FS[2:0]			DMDIS	FTH [1:0]	
	Reset value																									0		1	0	0	0	0	1
0x0070	**DMA_S4C**R	Reserved				CHSEL[2:0]			MBURST[1:0]		PBURST[1:0]		Reserved	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR [1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value					0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0074	**DMA_S4NDTR**	Reserved																NDT[15:.]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0078	**DMA_S4PAR**	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x007C	**DMA_S4M0AR**	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0080	**DMA_S4M1AR**	M1A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0084	**DMA_S4FCR**	Reserved																								FEIE	Reserved	FS[2:0]			DMDIS	FTH [1:0]	
	Reset value																									0		1	0	0	0	0	1
0x0088	**DMA_S5CR**	Reserved				CHSEL[2:0]			MBURST[1:0]		PBURST[1:0]		Reserved	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value					0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008C	**DMA_S5NDTR**	Reserved																NDT[15:.]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0090	**DMA_S5PAR**	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0094	**DMA_S5M0AR**	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0098	**DMA_S5M1AR**	M1A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x009C	**DMA_S5FCR**	Reserved																								FEIE	Reserved	FS[2:0]			DMDIS	FTH [1:0]	
	Reset value																									0		1	0	0	0	0	1
0x00A0	**DMA_S6CR**	Reserved				CHSEL[2:0]			MBURST[1:0]		PBURST[1:0]		Reserved	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value					0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00A4	**DMA_S6NDTR**	Reserved																NDT[15:.]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00A8	**DMA_S6PAR**	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00AC	**DMA_S6M0AR**	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00B0	**DMA_S6M1AR**	M1A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00B4	**DMA_S6FCR**	Reserved																								FEIE	Reserved	FS[2:0]			DMDIS	FTH [1:0]	
	Reset value																									0		1	0	0	0	0	1
0x00B8	**DMA_S7CR**	Reserved				CHSEL[2:0]			MBURST[1:0]		PBURST[1:0]		Reserved	CT	DBM	PL[1:0]		PINCOS	MSIZE[1:0]		PSIZE[1:0]		MINC	PINC	CIRC	DIR[1:0]		PFCTRL	TCIE	HTIE	TEIE	DMEIE	EN
	Reset value					0	0	0	0	0	0	0		0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00BC	**DMA_S7NDTR**	Reserved																NDT[15:.]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C0	**DMA_S7PAR**	PA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C4	**DMA_S7M0AR**	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C8	**DMA_S7M1AR**	M1A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00CC	**DMA_S7FCR**	Reserved																								FEIE	Reserved	FS[2:0]			DMDIS	FTH [1:0]	
	Reset value																									0		1	0	0	0	0	1

11. Контроллер Chrom-Art Accelerator™ (DMA2D)

11.1. Введение в DMA2D

Это специализированный DMA для манипуляции изображениями. Он может:

- Заполнять всё или часть изображения заданным цветом
- Копировать всё или часть исходного изображения во всё или часть другого изображения
- Копировать всё или часть исходного изображения во всё или часть другого изображения с преобразованием формата пикселей
- Смешивать часть и/или два исходных полных изображения с разными форматами пикселей в часть или всё целое изображение с другим форматом цвета.

Поддерживаются все классические цветовые схемы от 4-бит до 32-бит на пиксель с индексным или прямым режимом цвета. У DMA2D есть своя выделенная память для CLUT (таблиц выбора цвета).

11.2. Основные свойства DMA2D

Это:

- Архитектура с одним ведущим шины AHB.
- Интерфейс программирования ведомого AHB с поддержкой 8/16/32-бит доступом (кроме CLUT, у которого 32-бит доступ).
- Программируемый размер рабочей области
- Программируемое смещение областей источника и получателя
- Программируемые адреса источника и получателя во всём пространстве памяти
- До 2 источников операции смешивания
- Значение альфа можно менять (исходное значение, фиксированное или модулированное)
- Программируемый формат цвета источника и получателя
- До 11 форматов цвета от 4-бит до 32-бит на пиксель с косвенным или прямым кодированием цвета.

- 2 блока внутренней памяти для CLUT в косвенном режиме цвета
- Автоматическая загрузка или программирование CLUT через CPU
- Программируемый размер CLUT
- Внутренний таймер контроля полосы пропускания AHB
- 4 режима работы: регистр->память, память->память, память->память с изменением формата пикселей и память->память с изменением формата пикселей и смешиванием
- Заполнение области заданным цветом
- Копирование из области в другую
- Копирование изображений изменением формата пикселей
- Копирование из двух источников с независимым форматом цвета и смешиванием
- Аборт и остановка операций DMA2D
- Прерывание Водяного знака на программируемой линии пользователя
- Выдача прерывания по ошибке шины или конфликту доступа
- Выдача прерывания по концу процесса.

11.3. Функциональное описание DMA2D

11.3.1. Общее описание

Контроллер DMA2D выполняет прямые передачи в памяти. Как ведущий AHB он захватывает управление матрицей шин AHB и запускает операции AHB.

память->память работает в режимах:

- регистр->память,
- память->память,
- память->память с изменением формата пикселей,
- память->память с изменением формата пикселей и смешиванием

DMA2D программируется через порт ведомого AHB.

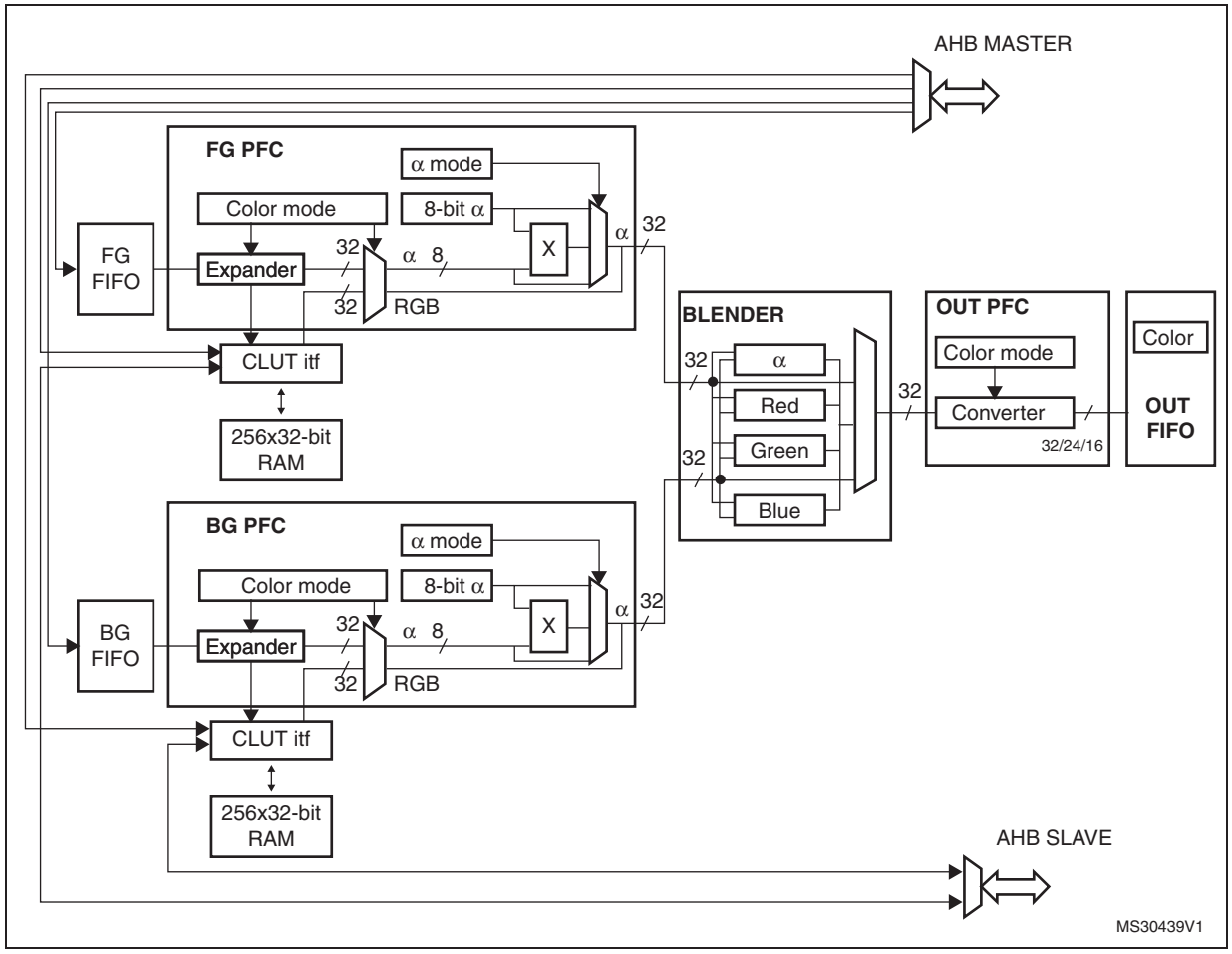


Рис. 40. Блок-схема DMA2D

11.3.2. Управление DMA2D

Контроллер DMA2D конфигурируется через регистр DMA2D_CR, который позволяет:

- Выбрать режим работы
- Разрешать прерывания DMA2D
- Пускать/задерживать/отменять передачи данных

11.3.3. FIFO переднего и заднего плана DMA2D

FIFO переднего (FG) и заднего (BG) плана DMA2D читают копируемые или обрабатываемые данные. Они читают пиксели в соответствии с их форматом, определённым в их конвертере форматов (PFC).

Программируют их через регистры:

- Регистр адреса памяти переднего плана (DMA2D_FGMAR)
- Регистр смещения переднего плана (DMA2D_FGOR)
- Регистр адреса памяти заднего плана (DMA2D_BGMAR)
- Регистр смещения заднего плана (DMA2D_BGBOR)
- Регистр числа строк (число строк и пикселей на строку) (DMA2D_NLR)

В режиме регистр->память FIFO не используются.

В режиме память->память (без преобразования формата пикселей и смешивания) используется только FG FIFO в качестве буфера.

В режиме память->память с преобразованием формата пикселей (без смешивания) BG FIFO не используется.

11.3.4. Конвертор формата пикселей переднего и заднего плана (PFC)

Они преобразуют пиксели в 32-бит формат и могут изменять значение альфа-канала.

Сначала преобразуется формат цвета. Исходный формат пикселя задают битами CM[3:0] в регистрах DMA2D_FGPFCCR и DMA2D_BGPFCCR.

Таблица 52. Поддерживаемые режимы цвета на входе

CM[3:0]	Режим цвета
0	ARGB8888
1	RGB888
10	RGB565
11	ARGB1555
100	ARGB4444
101	L8
110	AL44
111	AL88
1000	L4
1001	A8
1010	A4

Обозначения такие:

- Поле Alpha: 0xFF - непрозрачно, 0x00 - прозрачно.
- Поле R - красный
- Поле G -зелёный
- Поле B - синий
- Поле L - яркость. Это индекс в CLUT для извлечения RGB/ARGB компонентов.

Если исходный формат это прямой режим цвета, то 8-бит на канал расширяется копированием MSBs в LSBs. Это даёт линейность преобразования.

Если в исходном формате нет канала альфа, то он автоматически ставится в 0xFF.

Если исходный формат это косвенный режим цвета, то каждому конвертору придаётся 32-бит CLUT из 256 строк.

В режимах альфа A4 и A8 цвета нет вообще, он, единственный, берётся из регистров DMA2D_FGCOLR и DMA2D_BGCOLR.

Таблица 53. Порядок данных в памяти

Color Mode	@+3	@+2	@+1	@+0
ARGB8888	A_0[7:0]	R_0[7:0]	G_0[7:0]	B_0[7:0]
RGB888	B_1[7:0]	R_0[7:0]	G_0[7:0]	B_0[7:0]
	G_2[7:0]	B_2[7:0]	R_1[7:0]	G_1[7:0]
	R_3[7:0]	G_3[7:0]	B_3[7:0]	R_2[7:0]
RGB565	R_1[4:0]G_1[5:3]	G_1[2:0]B_1[4:0]	R_0[4:0]G_0[5:3]	G_0[2:0]B_0[4:0]
ARGB1555	A_1[0]R_1[4:0]G_1[4:3]	G_1[2:0]B_1[4:0]	A_0[0]R_0[4:0]G_0[4:3]	G_0[2:0]B_0[4:0]
ARGB4444	A_1[3:0]R_1[3:0]	G_1[3:0]B_1[3:0]	A_0[3:0]R_0[3:0]	G_0[3:0]B_0[3:0]
L8	L_3[7:0]	L_2[7:0]	L_1[7:0]	L_0[7:0]
AL44	A_3[3:0]L_3[3:0]	A_2[3:0]L_2[3:0]	A_1[3:0]L_1[3:0]	A_0[3:0]L_0[3:0]
AL88	A_1[7:0]	L_1[7:0]	A_0[7:0]	L_0[7:0]
L4	L_7[3:0]L_6[3:0]	L_5[3:0]L_4[3:0]	L_3[3:0]L_2[3:0]	L_1[3:0]L_0[3:0]
A8	A_3[7:0]	A_2[7:0]	A_1[7:0]	A_0[7:0]
A4	A_7[3:0]A_6[3:0]	A_5[3:0]A_4[3:0]	A_3[3:0]A_2[3:0]	A_1[3:0]A_0[3:0]

24-bit RGB888, выравненный на 32-бит, поддерживается через режим ARGB8888.

В полученном 32-бит значении альфа-канал может быть изменён по полю AM[1:0] в регистрах DMA2D_FGPFCCR и DMA2D_BGPFCCR. Альфа-канал может быть:

- сохранён как есть,
- заменён на ALPHA[7:0] из регистров DMA2D_FGPFCCR или DMA2D_BGPFCCR,
- заменён на произведение исходного альфа и ALPHA[7:0] из регистров DMA2D_FGPFCCR или DMA2D_BGPFCCR, делённых на 255.

Таблица 54. Конфигурация режима альфа

AM[1:0]	Alpha mode
0	Неизменно
1	Замена значением DMA2D_xxPFCCR
10	Замена на произведение исходного альфа и значения DMA2D_xxPFCCR / 255
11	Резерв

11.3.5. Интерфейс CLUT переднего и заднего плана

Он предоставляет доступ к памяти и автоматическую загрузку CLUT.

Есть три типа доступа:

- Чтение CLUT от PFC при преобразовании форматов
- Доступ к CLUT по порту ведомого AHB когда CPU читает и пишет данные в CLUT
- Запись в CLUT по порту ведущего AHB при автоматической загрузке CLUT

Память CLUT грузят двумя способами:

- Автоматическая загрузка

Последовательность такая:

a) Пишем адрес CLUT в регистр DMA2D_FGCMAR или DMA2D_BGCMAR

b) Пишем размер CLUT в поле CS[7:0] регистра DMA2D_FGPFCCR или DMA2D_BGPFCCR.

c) Ставим бит START в регистре DMA2D_FGPFCCR register (foreground CLUT) or DMA2D_BGPFCCR register (background CLUT) to start the transfer. During this automatic loading process, the CLUT is not accessible by the CPU. If a conflict occurs, a CLUT access error interrupt is raised assuming CAEIE is set to ‘1’ in DMA2D_CR.

- Ручная загрузка

  Пишут CLUT через порт ведомого DMA2D AHB, где отображена внутренняя память CLUT
  CLUT переднего плана лежит по адресу 0x0400, CLUT заднего плана по адресу 0x0800.

Формат CLUT может быть 24 или 32 бита. Это задаёт бит CCM в регистре DMA2D_FGPFCCR или DMA2D_BGPFCCR .

Таблица 55: Поддерживаемые режимы цвета CLUT.

CCM	CLUT color mode
0	32-bit ARGB8888
1	24-bit RGB888

Таблица 56: Порядок данных в памяти CLUT.

CLUT Color Mode	@+3	@+2	@+1	@+0
ARGB8888	A_0[7:0]	R_0[7:0]	G_0[7:0]	B_0[7:0]
RGB888	B_1[7:0]	R_0[7:0]	G_0[7:0]	B_0[7:0]
	G_2[7:0]	B_2[7:0]	R_1[7:0]	G_1[7:0]
	R_3[7:0]	G_3[7:0]	B_3[7:0]	R_2[7:0]

11.3.6. Смеситель CLUT

Блендер DMA2D смешивает пары пикселей по формуле:

$$\text{with } \alpha_{Mult} = \frac{\alpha_{FG} \cdot \alpha_{BG}}{255}$$

$$\alpha_{OUT} = \alpha_{FG} + \alpha_{BG} - \alpha_{Mult}$$

$$C_{OUT} = \frac{C_{FG}.\alpha_{FG} + C_{BG}.\alpha_{BG} - C_{BG}.\alpha_{Mult}}{\alpha_{OUT}} \quad \text{with C = R or G or B}$$

Деление округляется до ближайшего меньшего целого.

Использование смесителя зависит от режима в битах MODE[1:0] регистра DMA2D_CR.

11.3.7. Выходной PFC

Он преобразует пиксель из 32 бит в формат, заданный полем CM[2:0] регистра DMA2D_OPFCCR.

Таблица 57. Поддерживаемые режимы цвета на выходе

CM[2:0]	Color mode
0	ARGB8888
1	RGB888
10	RGB565
11	ARGB1555
100	ARGB4444

11.3.8. Выходной FIFO

Выходной FIFO пишет пиксели в формате выходного PFC.

Область получателя задана регистрами:

- Регистр выходной памяти (DMA2D_OMAR)
- Регистр смещения выходной памяти (DMA2D_OOR)
- Регистр числа строк (число строк и пиксели на строку) (DMA2D_NLR)

Если DMA2D работает в режиме регистр->память, то выходной прямоугольник заполняется цветом из регистра DMA2D_OCOLR с фиксированным 32-, 24- или 16-бит числом. Формат определён полем CM[2:0] в регистре DMA2D_OPFCCR.

Таблица 58: Порядок данных в памяти

Color Mode	@+3	@+2	@+1	@+0
ARGB8888	A_0[7:0]	R_0[7:0]	G_0[7:0]	B_0[7:0]
RGB888	B_1[7:0]	R_0[7:0]	G_0[7:0]	B_0[7:0]
	G_2[7:0]	B_2[7:0]	R_1[7:0]	G_1[7:0]
	R_3[7:0]	G_3[7:0]	B_3[7:0]	R_2[7:0]
RGB565	R_1[4:0]G_1[5:3]	G_1[2:0]B_1[4:0]	R_0[4:0]G_0[5:3]	G_0[2:0]B_0[4:0]
ARGB1555	A_1[0]R_1[4:0]G_1[4:3]	G_1[2:0]B_1[4:0]	A_0[0]R_0[4:0]G_0[4:3]	G_0[2:0]B_0[4:0]
ARGB4444	A_1[3:0]R_1[3:0]	G_1[3:0]B_1[3:0]	A_0[3:0]R_0[3:0]	G_0[3:0]B_0[3:0]

24-bit RGB888, выравненный на 32-бит, поддерживается через режим ARGB8888.

11.3.9. Таймер ведущего шины AHB

В блок ведущего шины AHB встроен 8-бит таймер, позволяющий ограничить пропускную способность шины. Он вносит задержку между двумя последовательными обращениями к шине, замедляя её. Им управляют через регистр DMA2D_AMPTCR.

11.3.10. Передачи DMA2D

Каждая передача данных DMA2D состоит из 4 шагов:

1. Читаются данные из памяти по адресу из регистра DMA2D_FGMAR в формате из регистра DMA2D_FGCR.
2. Читаются данные из памяти по адресу из регистра DMA2D_BGMAR в формате из регистра DMA2D_BGCR.
3. Извлечённые пиксели смешиваются в соответствии с их альфа и операцией альфа PFC.
4. Формат пикселя преобразуется по регистру DMA2D_OCR и данные пишутся в память по адресу из регистра DMA2D_OMAR.

11.3.11. Конфигурация DMA2D

Адреса источника и получателя могут лежать во всём диапазоне 4 Гбайт.

Биты MODE[1:0] в регистре DMA2D_CR задают четыре режима передачи:

- Регистр->память
- Память->память
- Память->память с PFC
- Память->память с PFC и смешиванием

Регистр->память

Это заполнение области, заданной в регистрах DMA2D_OMAR, DMA2D_NLR и DMA2D_OOR цветом из регистра DMA2D_OCOLR. Формат цвета стоит регистре DMA2D_OPFCCR.

Память->память

Это перенос данных через FIFO переднего плана из области, заданной регистром DMA2D_FGMAR, в область, заданную регистром DMA2D_OMAR. Режим цвета обеих областей стоит в битах CM[3:0] регистра DMA2D_FGPFCCR. Размер области источника пишут в регистры DMA2D_NLR и DMA2D_FGOR, область получателя в регистры DMA2D_NLR и DMA2D_OOR.

Память->память с PFC

Это перенос данных из области, заданной регистром DMA2D_FGMAR, в область, заданную регистром DMA2D_OMAR. с преобразованием формата пикселей в PFC среднего плана. Размер области источника пишут в регистры DMA2D_NLR и DMA2D_FGOR, область получателя в регистры DMA2D_NLR и DMA2D_OOR. Исходный формат пикселей лежит регистре DMA2D_FGPFCCR.

Для прямого режима цвета исходных пикселей каналы цвета расширяются до 8 бит.

Для косвенного режима цвета в память грузится CLUT:

1. Ставим адрес CLUT в DMA2D_FGCMAR.
2. Ставим размер CLUT в биты CS[7:0] регистра DMA2D_FGPFCCR.
3. Ставим формат CLUT (24 или 32 бит) в биты CCM регистра DMA2D_FGPFCCR.

4. Пускаем загрузку CLUT установкой бита START регистра DMA2D_FGPFCCR.

По завершению загрузки встаёт флаг CTCIF в регистре DMA2D_IFR и, если разрешено битом CTCIE в регистре DMA2D_CR, выдаётся прерывание. Автоматическая загрузка CLUT не может работать одновременно с классическими передачами DMA2D.

CLUT можно писать через CPU или любой другой ведущий порта APB. Во время передачи DMA2D с работающим CLUT (в косвенном режиме) доступ к CLUT невозможен.

Параллельно с преобразованием цвета можно добавить или изменить значение альфа в соответствии с значением регистра DMA2D_FGPFCCR. Если в исходном изображении канала альфа нет, то автоматически 0xFF для получения непрозрачного изображения. Значение альфа можно изменить в соответствии с битами AM[1:0] регистра DMA2D_FGPFCCR:

- Можно не менять.
- Можно заменить значением ALPHA[7:0] в регистре DMA2D_FGPFCCR.
- Можно заменить исходным значением, умноженным на ALPHA[7:0] из DMA2D_FGPFCCR, делённым на 255.

Итоговые 32-бит данные кодируются OUT PFC в формат, заданный полем CM[2:0] в регистре DMA2D_OPFCCR. Выходной формат пикселя не может быть в косвенном режиме, он не поддерживается. Полученные данные пишутся по адресу из регистра DMA2D_OMAR.

Память->память с PFC и смешиванием

В этом режиме в FIFO переднего и заднего плана выбираются два источника по адресам из регистров DMA2D_FGMAR и DMA2D_BGMAR.

Преобразование выполняют два раздельных PFC со своими CLUT.

Преобразованные в 32 бита пиксели смешиваются по уравнению:

$$\text{with } \alpha_{Mult} = \frac{\alpha_{FG} \cdot \alpha_{BG}}{255}$$

$$\alpha_{OUT} = \alpha_{FG} + \alpha_{BG} - \alpha_{Mult}$$

$$C_{OUT} = \frac{C_{FG} \cdot \alpha_{FG} + C_{BG} \cdot \alpha_{BG} - C_{BG} \cdot \alpha_{Mult}}{\alpha_{OUT}} \quad \text{with C = R or G or B}$$

Результат деления округляется до ближайшего меньшего целого.

32-бит результат кодируется выходным PFC в заданный формат и пишется по адресу из регистра DMA2D_OMAR.

Ошибки конфигурации

DMA2D проверяет конфигурацию перед каждой передачей/автоматической загрузкой и, если надо, ставит флаг ошибки. При стоящем бите CEIE в регистре DMA2D_CR выдаётся прерывание.

Непристойные конфигурации:

- Загрузка CLUT переднего плана: биты MA в DMA2D_FGCMAR не выравнены с CCM из DMA2D_FGPFCCR.
- Загрузка CLUT заднего плана: биты MA в DMA2D_BGCMAR не выравнены с CCM из DMA2D_BGPFCCR
- Передачи памяти (кроме регистр->память): MA из DMA2D_FGMAR не выравнены с CM из DMA2D_FGPFCCR
- Передачи памяти (кроме регистр->память): CM в DMA2D_FGPFCCR неверны
- Передачи памяти (кроме регистр->память): биты PL из DMA2D_NLR нечётные при CM из DMA2D_FGPFCCR равным A4 или L4
- Передачи памяти (кроме регистр->память): биты LO из DMA2D_FGOR нечётные при CM из DMA2D_FGPFCCR равным A4 или L4
- Передачи памяти (только при смешивании): биты MA из DMA2D_BGMAR не выравнены с CM из DMA2D_BGPFCCR
- Передачи памяти (только при смешивании): биты CM из DMA2D_BGPFCCR неверны

- Передачи памяти (только при смешивании): биты PL из DMA2D_NLR нечётные при CM из DMA2D_BGPFCCR равным A4 или L4
- Передачи памяти (только при смешивании): биты LO из DMA2D_BGOR нечётные при CM из DMA2D_BGPFCCR равным A4 или L4
- Передачи памяти (кроме память->память): биты MA из DMA2D_OMAR не выравнены с CM из DMA2D_OPFCCR.
- Передачи памяти (кроме память->память): биты CM из DMA2D_OPFCCR неверны
- Передачи памяти: биты NL в DMA2D_NLR = 0
- Передачи памяти: биты PL в DMA2D_NLR = 0

11.3.12. Управление передачей (старт, останов, аборт и конец)

Передача сконфигурированного DMA2D запускается установкой бита START в регистре DMA2D_CR. По концу передачи бит START автоматически снимается и встаёт флаг TCIF в регистре DMA2D_ISR. Прерывание выдаётся при стоящем бите TCIE в регистре DMA2D_CR.

Приостановить DMA2D можно в любое время установкой бита SUSP в регистре DMA2D_CR. Прекратить эту передачу можно установкой бита ABORT в регистре DMA2D_CR. Возобновить такую передачу можно снятием бита SUSP в регистре DMA2D_CR.

Работающую передачу можно прекратить установкой бита ABORT в регистре DMA2D_CR. В этом случае флаг TCIF не поднимается.

Автоматическую загрузку CLUT также можно прекратить или остановить битами ABORT или SUSP в регистре DMA2D_CR.

11.3.13. Водной знак

Он ставится для выдачи прерывания после записи в память последнего пикселя заданной строки.

Номер строки пишут в поле LW[15:0] регистра DMA2D_LWR. После записи последнего пикселя ставится бит TWIF в регистре DMA2D_ISR и при стоящем бите TWIE в регистре DMA2D_CR выдаётся прерывание.

11.3.14. Обработка ошибок

Есть два типа ошибок:

- Ошибки ведущего порта AHB ставят флаг TEIF в регистре DMA2D_ISR.
- Конфликты доступа к CLUT (CPU пытается обратиться к CLUT во время её загрузки или при работающей передаче DMA2D) ставят флаг CAEIF в регистре DMA2D_ISR.

Оба флага связана со своими прерываниями, которые разрешают битами TEIE и CAEIE в регистре DMA2D_CR.

11.3.15. Паузы AHB

Для ограничения использования полосы пропускания шины AHB между двумя последовательными доступами можно вставлять программируемую паузу. Включается сиё битом EN в DMA2D_AMTCR.

Значение паузы пишут в поле DT[7:0] регистра DMA2D_AMTCR. Это гарантированный минимум циклов шины между двумя последовательными передачами.

Изменение паузы работающего DMA2D работает при следующей передаче AHB.

11.4. Прерывания DMA2D

Прерывания выдаются по следующим событиям:

- Ошибка конфигурации
- Передача CLUT завершена
- Ошибка доступа к CLUT
- Достигнут водяной знак
- Передача завершена
- Ошибка передачи

Таблица 59. Прерывания DMA2D

Событие	Флаг события	Бит разрешения
Ошибка конфигурации	CEIF	CEIE
Передача CLUT завершена	CTCIF	CTCIE
Ошибка доступа к CLUT	CAEIF	CAEIE
Достигнут водяной знак	TWF	TWIE
Передача завершена	TCIF	TCIE
Ошибка передачи	TEIF	TEIE

11.5. Регистры DMA2D

11.5.1. Регистр управления (DMA2D_CR)

Смещение адреса: 0x0000

По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved														MODE	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		CEIE	CTCIE	CAEIE	TWIE	TCIE	TEIE	Reserved					ABORT	SUSP	START
		rw	rw	rw	rw	rw	rw						rs	rw	rs

Биты разрешения прерываний ставят и снимают программно.

0: Запрещено

1: Разрешено

— Биты 31:18 Резерв, не трогать.

— Биты 17:16 **MODE**: Режим DMA2D

Ставят и снимают программно при стоящей передаче.

00: Память->память (чтение только FG)

01: Память->память с PFC (чтение только FG с рабочим FG PFC)

10: Память->память с PFC и смешивание (чтение FG и BG с PFC и смешивание)

11: Регистр->память (ни FG, ни BG, только выход)

— Биты 15:14 Резерв, не трогать.

— Бит 13 **CEIE**: Разрешение прерывания Ошибки конфигурации

— Бит 12 **CTCIE**: Разрешение прерывания конца загрузки CLUT

— Бит 11 **CAEIE**: Разрешение прерывания ошибки доступа к CLUT

— Бит 10 **TWIE**: Разрешение прерывания водяного знака

— Бит 9 **TCIE**: Разрешение прерывания конца передачи

— Бит 8 **TEIE**: Разрешение прерывания ошибки передачи

— Биты 7:3 Резерв, не трогать.

— Бит 2 **ABORT**: Аборт текущей передачи

Ставят и снимают программно, автоматически снимается при снятии бита START.

0: Нету

1: Делаем

— Бит 1 **SUSP**: Останов

Ставят и снимают программно, автоматически снимается при снятии бита START.

0: Нету

1: Делаем

— Бит 0 **START**: Пуск DMA2D

Автоматически снимается при:

- Конце передачи
- Аборте передачи битом ABORT в DMA2D_CR
- Ошибке передачи данных
- Уже работающей передаче или загрузке CLUT.

11.5.2. Регистр статуса прерываний (DMA2D_ISR)

Смещение адреса: 0x0004

По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										CEIF	CTCIF	CAEIF	TWIF	TCIF	TEIF
										r	r	r	r	r	r

— Биты 31:6 Резерв, не трогать.

— Бит 5 **CEIF**: Флаг прерывания ошибки конфигурации

Ставится при установке бита START в DMA2D_CR и неверной конфигурации DMA2D_FGPFCCR или DMA2D_BGPFCCR.

— Бит 4 **CTCIF**: Флаг прерывания конца загрузки CLUT

Ставится по концу загрузки CLUT из системной памяти.

— Бит 3 **CAEIF**: Флаг прерывания ошибки доступа к CLUT

Ставится при доступе к CLUT во время загрузки CLUT из системной памяти.

— Бит 2 **TWIF**: Флаг прерывания передачи водяного знака

Ставится после передачи последнего пикселя отмеченной строки.

— Бит 1 **TCIF**: Флаг прерывания конца передачи данных.

— Бит 0 **TEIF**: Флаг прерывания ошибки передачи.

Ставится при появлении ошибки AHB во время передачи или загрузки CLUT.

11.5.3. Регистр очистки флагов прерываний (DMA2D_IFCR)

Смещение адреса: 0x0008

По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										CCEIF	CCTCIF	CAECIF	CTWIF	CTCIF	CTEIF
										rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Запись 1 в эти биты снимает соответствующий бит в регистре DMA2D_ISR.

— Биты 31:6 Резерв, не трогать.

— Бит 5 **CCEIF**: Чистка флага прерывания ошибки конфигурации

— Бит 4 **CCTCIF**: Чистка флага прерывания конца загрузки CLUT

— Бит 3 **CCAEIF**: Чистка флага прерывания ошибки доступа к CLUT

— Бит 2 **CTWIF**: Чистка флага прерывания передачи водяного знака

— Бит 1 **CTCIF**: Чистка флага прерывания конца передачи данных.

— Бит 0 **CTEIF**: Чистка флага прерывания ошибки передачи.

11.5.4. Регистр адреса памяти переднего плана (DMA2D_FGMAR)

Смещение адреса: 0x000C

По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:0 **MA[31:0]**: Адрес памяти переднего плана
Пишут только при стоящих передачах, иначе только читают. Выравнивание адреса должно соответствовать формату пикселей.

11.5.5. Регистр смещения переднего плана (DMA2D_FGOR)

Смещение адреса: 0x0010

По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	LO[13:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:14 Резерв, не трогать.

— Биты 13:0 **LO[31:0]**: Смещение строки
Смещение строки изображения переднего плана в пикселях, используется для передвижения по строкам. Писать можно только при стоящей передаче. При 4-бит пикселях должно быть чётным.

11.5.6. Регистр адреса памяти заднего плана (DMA2D_BGMAR)

Смещение адреса: 0x0014

По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:0 **MA[31:0]**: Адрес памяти заднего плана
Пишут только при стоящих передачах, иначе только читают. Выравнивание адреса должно соответствовать формату пикселей.

11.5.7. Регистр смещения заднего плана (DMA2D_BGOR)

Смещение адреса: 0x0018

По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	LO[13:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:14 Резерв, не трогать.

— Биты 13:0 **LO[31:0]**: Смещение строки
Смещение строки изображения заднего плана в пикселях, используется для передвижения по строкам. Писать можно только при стоящей передаче. При 4-бит пикселях должно быть чётным.

11.5.8. Регистр управления PFC переднего плана (DMA2D_FGPFCCR)

Смещение адреса: 0x001C

По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ALPHA[7:0]								Reserved						AM[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw							rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CS[7:0]								Reserved		START	CCM	CM[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw			rs	rw	rw	rw	rw	rw

— Биты 31:24 **ALPHA[7:0]**: Значение Альфа

Им можно заменять исходное значение или умножить на исходное значение в соответствии с режимом из битов AM[1:0]. Писать можно только при остановленной передаче.

— Биты 23:18 Резерв, не трогать.

— Биты 17:16 **AM[1:0]**: Режим Альфа

Режим использования значения альфа для изображения переднего плана. Писать можно только при остановленной передаче.

00: Никакой модификации

01: Заменить исходный канал альфа на значение из ALPHA[7:0]

10: Заменить исходный канал альфа на значение из ALPHA[7:0], умноженное на исходное значение

Иное: Бестолковщина

— Биты 15:8 **CS[7:0]**: Размер CLUT изображения переднего плана

Писать можно только при остановленной загрузке CLUT. Число строк CLUT равно CS[7:0] + 1.

— Биты 7:6 Резерв, не трогать.

— Бит 5 **START**: Пуск автоматической загрузки CLUT.

Автоматически снимается:

- по концу передачи
- при аборте передачи битом ABORT в регистре DMA2D_CR
- при ошибке передачи
- когда передача не стартует из-за ошибки конфигурации или работающей передаче или загрузки CLUT заднего плана.

— Бит 4 **CCM**: Формат цвета CLUT

Писать можно только при остановленной передаче.

0: ARGB8888

1: RGB888

Иное: Бред

— Биты 3:0 **CM[3:0]**: Формат цвета изображения переднего плана

Писать можно только при остановленной передаче.

0000: ARGB8888

0001: RGB888

0010: RGB565

0011: ARGB1555

0100: ARGB4444

0101: L8

0110: AL44

0111: AL88

1000: L4

1001: A8

1010: A4

Иное: Фигня

11.5.9. Регистр цвета переднего плана (DMA2D_FGCOLR)

Смещение адреса: 0x0020

По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								RED[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GREEN[7:0]								BLUE[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Это значения Цвета изображения переднего плана для режимов A4 или A8. Писать можно только при остановленной передаче.

— Биты 31:24 Резерв, не трогать.
— Биты 17:16 **RED[7:0]**: Значение Красного
— Биты 17:16 **GREEN[7:0]**: Значение Зелёного
— Биты 15:8 **BLUE[7:0]**: Значение Синего.

11.5.10. Регистр управления PFC заднего плана (DMA2D_BGPFCCR)

Смещение адреса: 0x0024

По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ALPHA[7:0]								Reserved						AM[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw							rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CS[7:0]								Reserved		START	CCM	CM[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw			rs	rw	rw	rw	rw	rw

— Биты 31:24 **ALPHA[7:0]**: Значение Альфа
Им можно заменять исходное значение или умножить на исходное значение в соответствии с режимом из битов AM[1:0]. Писать можно только при остановленной передаче.
— Биты 23:18 Резерв, не трогать.
— Биты 17:16 **AM[1:0]**: Режим Альфа
Режим использования значения альфа для изображения заднего плана. Писать можно только при остановленной передаче.
00: Никакой модификации
01: Заменить исходный канал альфа на значение из ALPHA[7:0]
10: Заменить исходный канал альфа на значение из ALPHA[7:0], умноженное на исходное значение
Иное: Бестолковщина
— Биты 15:8 **CS[7:0]**: Размер CLUT изображения переднего плана
Писать можно только при остановленной загрузке CLUT. Число строк CLUT равно CS[7:0] + 1.
— Биты 7:6 Резерв, не трогать.
— Бит 5 **START**: Пуск автоматической загрузки CLUT.
Автоматически снимается:
- по концу передачи
- при аборте передачи битом ABORT в регистре DMA2D_CR
- при ошибке передачи
- когда передача не стартует из-за ошибки конфигурации или работающей передаче или загрузки CLUT переднего плана.

— Бит 4 **CCM**: Формат цвета CLUT
Писать можно только при остановленной передаче.
0: ARGB8888
1: RGB888
Иное: Бред
— Биты 3:0 **CM[3:0]**: Формат цвета изображения заднего плана
Писать можно только при остановленной передаче.
0000: ARGB8888
0001: RGB888
0010: RGB565

0011: ARGB1555
0100: ARGB4444
0101: L8
0110: AL44
0111: AL88
1000: L4
1001: A8
1010: A4
Иное: Фигня

11.5.11. Регистр цвета заднего плана (DMA2D_BGCOLR)

Смещение адреса: 0x0028
По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								RED[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GREEN[7:0]								BLUE[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Это значения Цвета изображения заднего плана для режимов A4 или A8. Писать можно только при остановленной передаче.

— Биты 31:24 Резерв, не трогать.
— Биты 17:16 **RED[7:0]**: Значение Красного
— Биты 17:16 **GREEN[7:0]**: Значение Зелёного
— Биты 15:8 **BLUE[7:0]**: Значение Синего.

11.5.12. Регистр адреса памяти CLUT переднего плана (DMA2D_FGCMAR)

Смещение адреса: 0x002C
По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:0 **MA[31:0]**: Адрес памяти CLUT переднего плана
Пишут только при стоящих передачах, иначе только читают. Выравнивание адреса должно соответствовать формату пикселей.

11.5.13. Регистр адреса памяти заднего плана (DMA2D_BGCMAR)

Смещение адреса: 0x0030
По сбросу: 0x0000 0000.

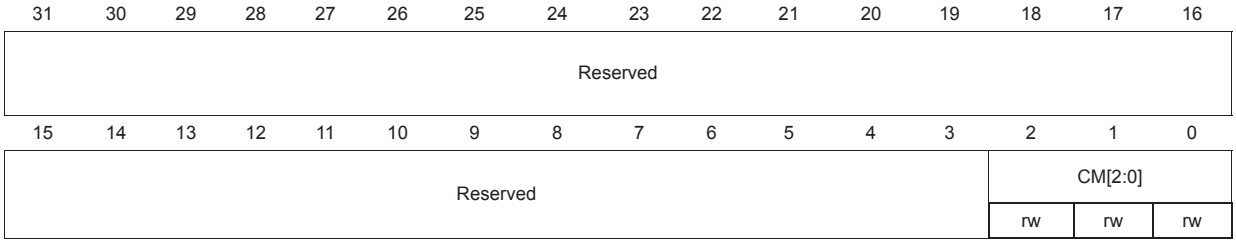
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:0 **MA[31:0]**: Адрес памяти CLUT заднего плана
Пишут только при стоящих передачах, иначе только читают. Выравнивание адреса должно соответствовать формату пикселей.

11.5.14. Регистр управления выходного PFC (DMA2D_OPFCCR)

Смещение адреса: 0x0034

По сбросу: 0x0000 0000.



— Биты 31:3 Резерв, не трогать.

— Биты 31:0 **CM[2:0]**: Формат цвета выходного изображения

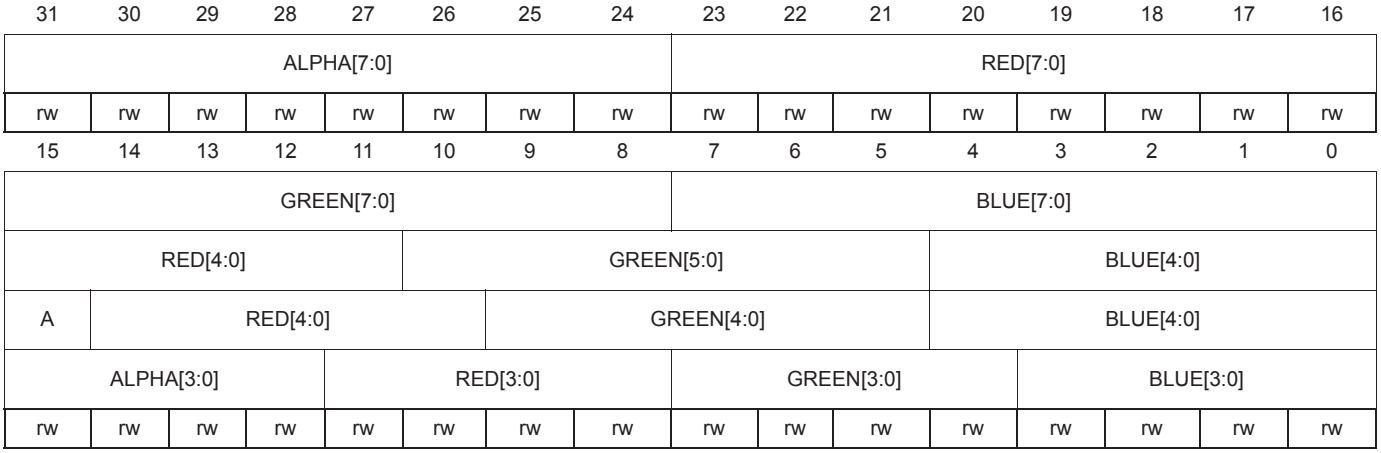
Писать можно только при остановленной передаче.

000: ARGB8888
001: RGB888
010: RGB565
011: ARGB1555
100: ARGB4444
Иное: Фигня

11.5.15. Регистр выходного цвета (DMA2D_OCOLR)

Смещение адреса: 0x0038

По сбросу: 0x0000 0000.



Это значения Цвета выходного изображения. Писать можно только при остановленной передаче.

— Биты 31:24 **ALPHA[7:0]**:

— Биты 17:16 **RED[7:0]**: Значение Красного

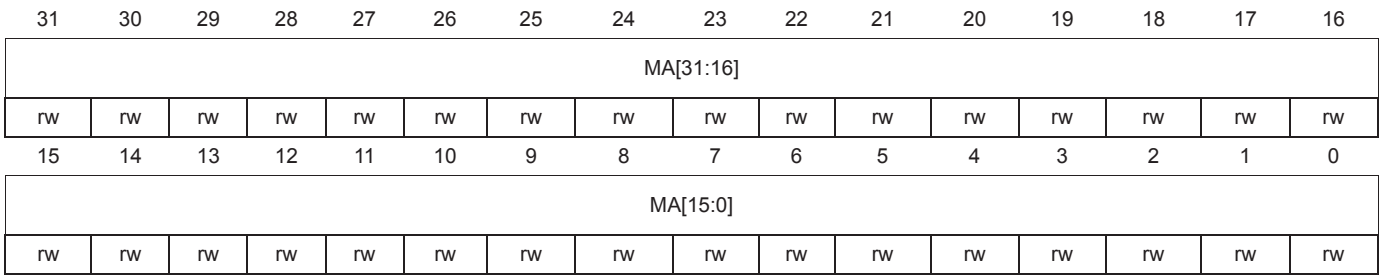
— Биты 17:16 **GREEN[7:0]**: Значение Зелёного

— Биты 15:8 **BLUE[7:0]**: Значение Синего.

11.5.16. Регистр адреса памяти выхода (DMA2D_OMAR)

Смещение адреса: 0x003C

По сбросу: 0x0000 0000.



— Биты 31:0 **MA[31:0]**: Адрес памяти для выходного FIFO

Пишут только при стоящих передачах, иначе только читают. Выравнивание адреса должно соответствовать формату пикселей.

11.5.17. Регистр смещения выхода (DMA2D_OOR)

Смещение адреса: 0x0040

По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		LO[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:14 Резерв, не трогать.

— Биты 13:0 **LO[13:0]**: Смещение строки

Смещение строки изображения заднего плана в пикселях, используется для передвижения по строкам. Писать можно только при стоящей передаче. При 4-бит пикселях должно быть чётным.

11.5.18. Регистр числа строк (DMA2D_NLR)

Смещение адреса: 0x0044

По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		PL[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:30 Резерв, не трогать.

— Биты 29:16 **PL[13:0]**: Число пикселей на строку в передаваемой области

Писать можно только при стоящей передаче. При 4-бит пикселях должно быть чётным.

— Биты 15:0 **NL[15:0]**: Число строк в в передаваемой области

Писать можно только при стоящей передаче.

11.5.19. Регистр строки водяного знака (DMA2D_LWR)

Смещение адреса: 0x0048

По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LW[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:16 Резерв, не трогать.

— Биты 15:0 **LW[15:0]**: Номер строки водяного знака

После передачи строки водяного знака выдаётся прерывание. Писать можно только при стоящей передаче.

11.5.20. Регистр таймера ведущего AHB (DMA2D_AMTCR)

Смещение адреса: 0x004C

По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DT[7:0]								Reserved							EN
rw	rw	rw	rw	rw	rw	rw	rw								rw

— Биты 31:16 Резерв, не трогать.

— Биты 15:8 **DT[7:0]**: Число циклов AHB между двумя последовательными передачами по AHB

— Биты 7:1 Резерв, не трогать.

— Бит 0 **EN**: Включение паузы

11.5.21. Карта регистров DMA2D

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	DMA2D_CR	Reserved														MODE[1:0]		Reserved			CEIE	CTCIE	CAEIE	TWIE	TCIE	TEIE	Reserved				ABORT	SUSP	START
	Reset value															0	0				0	0	0	0	0	0					0	0	0
0x0004	DMA2D_ISR	Reserved																										CEIF	CTCIF	CAEIF	TWIF	TCIF	TEIF
	Reset value																											0	0	0	0	0	0
0x0008	DMA2D_IFCR	Reserved																										CCEIF	CCTCIF	CAECIF	CTWIF	CTCIF	CTEIF
	Reset value																											0	0	0	0	0	0
0x000C	DMA2D_FGMAR	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0010	DMA2D_FGOR	Reserved																			LO[13:0]												
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x0014	DMA2D_BGMAR	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0018	DMA2D_BGOR	Reserved																			LO[13:0]												
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x001C	DMA2D_FGPFCCR	ALPHA[7:0]								Reserved						AM[1:0]		CS[7:0]								Res	START	CCM	CM[3:0]				
	Reset value	0	0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x0020	DMA2D_FGCOLR	APLHA[7:0]								RED[7:0]								GREEN[7:0]								BLUE[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0024	DMA2D_BGPFCCR	ALPHA[7:0]								Reserved						AM[1:0]		CS[7:0]								Res	START	CCM	CM[3:0]				
	Reset value	0	0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x0028	DMA2D_BGCOLR	APLHA[7:0]								RED[7:0]								GREEN[7:0]								BLUE[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x002C	DMA2D_FGCMAR	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0030	DMA2D_BGCMAR	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0034	DMA2D_OPFCCR	Reserved																													CM[2:0]		
	Reset value																														0	0	0
0x0038	DMA2D_OCOLR	APLHA[7:0]								RED[7:0]								GREEN[7:0]								BLUE[7:0]							
		Reserved																RED[4:0]					GREEN[6:0]							BLUE[4:0]		
		Reserved																A	RED[4:0]				GREEN[4:0]							BLUE[4:0]		
		Reserved																ALPHA[3:0]				RED[3:0]				GREEN[3:0]				BLUE[3:0]		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x003C	DMA2D_OMAR	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0040	DMA2D_OOR	Reserved																LO[13:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0044	DMA2D_NLR	Res	PL[13:0]															NL[15:0]															
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0048	DMA2D_LWR	Reserved																LW[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004C	DMA2D_AMTCR	Reserved																DT[7:0]								Reserved							EN
	Reset value																	0	0	0	0	0	0	0	0								0
0x0050-Ox03FF	-	Reserved																															
0x0400-0x07FF	DMA2D_FGCLUT	APLHA[7:0][255:0]								RED[7:0][255:0]								GREEN[7:0][255:0]								BLUE[7:0][255:0]							
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x0800-0x0BFF	DMA2D_BGCLUT	APLHA[7:0][255:0]								RED[7:0][255:0]								GREEN[7:0][255:0]								BLUE[7:0][255:0]							
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

12. Прерывания и события

12.1. Контроллер вложенных прерываний (NVIC)

12.1.1. Характеристики NVIC

- 82 маскируемых прерываний для STM32F405xx/07xx and STM32F415xx/17xx и до 91 маскируемых прерывания для STM32F42xxx and STM32F43xxx (не включая 16 линий прерываний в Cortex®-M4 с FPU)
- 16 программируемых уровней приоритета (4 бита)
- Низкая задержка обработки прерываний и исключений
- Управление питанием
- Регистры управления

NVIC и ядро тесно связаны, отсюда низкая задержка и обработка запоздавших прерываний.

Все прерывания и исключения в системе обрабатываются именно NVIC. См. PM0214.

12.1.2. Регистр калибровки SysTick

Содержит число 18750, что даёт опорную базу 1 ms с SysTick, установленным на 18.75 MHz (HCLK/8, с HCLK установленным на 150 MHz).

12.1.3. Векторы прерываний и исключений

Таблица 61. Таблица векторов для STM32F405xx/07xx и STM32F415xx/17xx

№	Pri	Тип приор.	Имя	Описание прерывания	Адрес
—	—	—	—	Резерв	0x0000_0000
—	-3	фикс.	Сброс	Сброс	0x0000_0004
—	-2	фикс.	NMI	Не маскируемое прерывание. Здесь же и CSS	0x0000_0008
—	-1	фикс.	HardFault	Все классы сбоев	0x0000_000C
—	0	устан.	MemManage	Управление памятью	0x0000_0010
—	1	устан.	BusFault	Сбой предвыборки и доступа к памяти	0x0000_0014
—	2	устан.	UsageFault	Неопределённая команда или недопустимое состояние	0x0000_0018
—	—	—	—	Резерв	0x0000_001C - 0x0000_002B

№	Pri	Тип приор.	Имя	Описание прерывания	Адрес
—	3	устан.	SVCall	Вызов Супервизора	0x0000_002C
—	4	устан.	Debug Monitor	Монитор отладки	0x0000_0030
—	—	—	—	Резерв	0x0000_0034
—	5	устан.	PendSV	Задерживаемый запрос системы	0x0000_0038
—	6	устан.	SysTick	Системный таймер	0x0000_003C
0	7	устан.	WWDG	Оконный сторожевой таймер	0x0000_0040
1	8	устан.	PVD	PVD через EXTI Line детектор	0x0000_0044
2	9	устан.	TAMP_STAMP	Вмешательство и метки времени по EXTI	0x0000_0048
3	10	устан.	RTC_WKUP	Прерывание побудки RTC по EXTI	0x0000_004C
4	11	устан.	FLASH	Глобальное прерывание FLASH	0x0000_0050
5	12	устан.	RCC	Глобальное прерывание RCC	0x0000_0054
6	13	устан.	EXTI0	EXTI Line0	0x0000_0058
7	14	устан.	EXTI1	EXTI Line1	0x0000_005C
8	15	устан.	EXTI2	EXTI Line2	0x0000_0060
9	16	устан.	EXTI3	EXTI Line3	0x0000_0064
10	17	устан.	EXTI4	EXTI Line4	0x0000_0068
11	18	устан.	DMA1_Stream0	Глобальное прерывание DMA1_Stream0	0x0000_006C
12	19	устан.	DMA1_Stream1	Глобальное прерывание DMA1_Stream1	0x0000_0070
13	20	устан.	DMA1_Stream2	Глобальное прерывание DMA1_Stream2	0x0000_0074
14	21	устан.	DMA1_Stream3	Глобальное прерывание DMA1_Stream3	0x0000_0078
15	22	устан.	DMA1_Stream4	Глобальное прерывание DMA1_Stream4	0x0000_007C
16	23	устан.	DMA1_Stream5	Глобальное прерывание DMA1_Stream5	0x0000_0080
17	24	устан.	DMA1_Stream6	Глобальное прерывание DMA1_Stream6	0x0000_0084
18	25	устан.	ADC	Глобальное прерывание ADC1, ADC2 и ADC3	0x0000_0088
19	26	устан.	CAN1_TX	Прерывания CAN1_TX	0x0000_008C
20	27	устан.	CAN1_RX0	Прерывания CAN1_RX0	0x0000_0090
21	28	устан.	CAN1_RX1	Прерывание CAN1_RX1	0x0000_0094
22	29	устан.	CAN1_SCE	Прерывание CAN1_SCE	0x0000_0098
23	30	устан.	EXTI9_5	Прерывания EXTI Line[9:5]	0x0000_009C
24	31	устан.	TIM1_BRK_TIM9	Прерывание TIM1_BRK и глобальное TIM9	0x0000_00A0
25	32	устан.	TIM1_UP_TIM10	Прерывание TIM1_UP и глобальное TIM10	0x0000_00A4
26	33	устан.	TIM1_TRG_COM_ TIM11	Прерывание TIM1 Trigger and Commutation и глобальное TIM11	0x0000_00A8
27	34	устан.	TIM1_CC	Прерывание TIM1 Capture Compare	0x0000_00AC
28	35	устан.	TIM2	Глобальное прерывание TIM2	0x0000_00B0
29	36	устан.	TIM3	Глобальное прерывание TIM3	0x0000_00B4
30	37	устан.	TIM4	Глобальное прерывание TIM4	0x0000_00B8
31	38	устан.	I2C1_EV	Событие I^2C1	0x0000_00BC

№	Pri	Тип приор.	Имя	Описание прерывания	Адрес
32	39	устан.	I2C1_ER	Прерывание ошибки I²C1	0x0000_00C0
33	40	устан.	I2C2_EV	Событие I²C2	0x0000_00C4
34	41	устан.	I2C2_ER	Прерывание ошибки I²C2	0x0000_00C8
35	42	устан.	SPI1	Глобальное прерывание SPI1	0x0000_00CC
36	43	устан.	SPI2	Глобальное прерывание SPI2	0x0000_00D0
37	44	устан.	USART1	Глобальное прерывание USART1	0x0000_00D4
38	45	устан.	USART2	Глобальное прерывание USART2	0x0000_00D8
39	46	устан.	USART3	Глобальное прерывание USART3	0x0000_00DC
40	47	устан.	EXTI15_10	Прерывания EXTI Line[15:10]	0x0000_00E0
41	48	устан.	RTCAlarm	Прерывание RTC alarm через EXTI	0x0000_00E4
42	49	устан.	OTG_FS_WKUP	Прерывание USB On-The-Go FS Wakeup через EXTI line	0x0000_00E8
43	50	устан.	TIM8_BRK_TIM12	TIM8 Break interrupt and TIM12 global interrupt	0x0000_00EC
44	51	устан.	TIM8_UP_TIM13	TIM8 Update interrupt and TIM13 global interrupt	0x0000 00F0
45	52	устан.	TIM8_TRG_COM_ TIM14	TIM8 Trigger and Commutation interrupts and TIM14 global interrupt	0x0000 00F4
46	53	устан.	TIM8_CC	TIM8 Capture Compare interrupt	0x0000 00F8
47	54	устан.	DMA1_Stream7	DMA1 Stream7 global interrupt	0x0000 00FC
48	55	устан.	FSMC	FSMC global interrupt	0x0000 0100
49	56	устан.	SDIO	SDIO global interrupt	0x0000 0104
50	57	устан.	TIM5	Глобальное прерывание TIM5	0x0000_0108
51	58	устан.	SPI3	Глобальное прерывание SPI3	0x0000_010C
52	59	устан.	UART4	Глобальное прерывание UART4	0x0000_0110
53	60	устан.	UART5	Глобальное прерывание UART5	0x0000_0114
54	61	устан.	TIM6_DAC	Глобальное прерывание TIM6 TIM6 global interrupt, DAC1 and DAC2 underrun error interrupts	0x0000_0118
55	62	устан.	TIM7	Глобальное прерывание TIM7	0x0000_011C
56	63	устан.	DMA2_Stream0	Глобальное прерывание DMA2_Stream0	0x0000_0120
57	64	устан.	DMA2_Stream1	Глобальное прерывание DMA2_Stream1	0x0000_0124
58	65	устан.	DMA2_Stream2	Глобальное прерывание DMA2_Stream2	0x0000_0128
59	66	устан.	DMA2_Stream3	Глобальное прерывание DMA2_Stream3	0x0000_012C
60	67	устан.	DMA2_Stream4	Глобальное прерывание DMA2_Stream4	0x0000_0130
61	68	устан.	ETH	Глобальное прерывание Ethernet	0x0000_0134
62	69	устан.	ETH_WKUP	Прерывание Ethernet Wakeup через EXTI line	0x0000_0138
63	70	устан.	CAN2_TX	Прерывания CAN2_TX	0x0000_013C
64	71	устан.	CAN2_RX0	Прерывания CAN2_RX0	0x0000_0140
65	72	устан.	CAN2_RX1	Прерывания CAN2_RX1	0x0000_0144
66	73	устан.	CAN2_SCE	Прерывание CAN2_SCE	0x0000_0148
67	74	устан.	OTG_FS	Глобальное прерывание USB On The Go FS	0x0000_014C

№	Pri	Тип приор.	Имя	Описание прерывания	Адрес
68	75	устан.	DMA2_Stream5	DMA2 Stream5 global interrupt	0x0000 0150
69	76	устан.	DMA2_Stream6	DMA2 Stream6 global interrupt	0x0000 0154
70	77	устан.	DMA2_Stream7	DMA2 Stream7 global interrupt	0x0000 0158
71	78	устан.	USART6	USART6 global interrupt	0x0000 015C
72	79	устан.	I2C3_EV	I^2C3 event interrupt	0x0000 0160
73	80	устан.	I2C3_ER	I^2C3 error interrupt	0x0000 0164
74	81	устан.	OTG_HS_EP1_OUT	USB On The Go HS End Point 1 Out global interrupt	0x0000 0168
75	82	устан.	OTG_HS_EP1_IN	USB On The Go HS End Point 1 In global interrupt	0x0000 016C
76	83	устан.	OTG_HS_WKUP	USB On The Go HS Wakeup through EXTI interrupt	0x0000 0170
77	84	устан.	OTG_HS	USB On The Go HS global interrupt	0x0000 0174
78	85	устан.	DCMI	DCMI global interrupt	0x0000 0178
79	86	устан.	CRYP	CRYP crypto global interrupt	0x0000 017C
80	87	устан.	HASH_RNG	Hash and Rng global interrupt	0x0000 0180
81	88	устан.	FPU	FPU global interrupt	0x0000 0184

Таблица 62. Таблица векторов для STM32F42xxx и STM32F43xxx

№	Pri	Тип приор.	Имя	Описание прерывания	Адрес
—	—	—	—	Резерв	0x0000_0000
—	-3	фикс.	Сброс	Сброс	0x0000_0004
—	-2	фикс.	NMI	Не маскируемое прерывание. Здесь же и CSS	0x0000_0008
—	-1	фикс.	HardFault	Все классы сбоев	0x0000_000C
—	0	устан.	MemManage	Управление памятью	0x0000_0010
—	1	устан.	BusFault	Сбой предвыборки и доступа к памяти	0x0000_0014
—	2	устан.	UsageFault	Неопределённая команда или недопустимое состояние	0x0000_0018
—	—	—	—	Резерв	0x0000_001C - 0x0000_002B
—	3	устан.	SVCall	Вызов Супервизора	0x0000_002C
—	4	устан.	Debug Monitor	Монитор отладка	0x0000_0030
—	—	—	—	Резерв	0x0000_0034
—	5	устан.	PendSV	Задерживаемый запрос системы	0x0000_0038
—	6	устан.	SysTick	Системный таймер	0x0000_003C
0	7	устан.	WWDG	Оконный сторожевой таймер	0x0000_0040
1	8	устан.	PVD	PVD через EXTI Line детектор	0x0000_0044
2	9	устан.	TAMP_STAMP	Взлом и метки времени по линии EXTI	0x0000_0048
3	10	устан.	RTC_WKUP	Побудка RTC по линии EXTI	0x0000_004C
4	11	устан.	FLASH	Глобальное прерывание FLASH	0x0000_0050
5	12	устан.	RCC	Глобальное прерывание RCC	0x0000_0054
6	13	устан.	EXTI0	EXTI Line0	0x0000_0058

№	Pri	Тип приор.	Имя	Описание прерывания	Адрес
7	14	устан.	EXTI1	EXTI Line1	0x0000_005C
8	15	устан.	EXTI2	EXTI Line2	0x0000_0060
9	16	устан.	EXTI3	EXTI Line3	0x0000_0064
10	17	устан.	EXTI4	EXTI Line4	0x0000_0068
11	18	устан.	DMA1_Stream0	Глобальное прерывание DMA1_Stream0	0x0000_006C
12	19	устан.	DMA1_Stream1	Глобальное прерывание DMA1_Stream1	0x0000_0070
13	20	устан.	DMA1_Stream2	Глобальное прерывание DMA1_Stream2	0x0000_0074
14	21	устан.	DMA1_Stream3	Глобальное прерывание DMA1_Stream3	0x0000_0078
15	22	устан.	DMA1_Stream4	Глобальное прерывание DMA1_Stream4	0x0000_007C
16	23	устан.	DMA1_Stream5	Глобальное прерывание DMA1_Stream5	0x0000_0080
17	24	устан.	DMA1_Stream6	Глобальное прерывание DMA1_Stream6	0x0000_0084
18	25	устан.	ADC	Глобальное прерывание ADC1, ADC2 и ADC3	0x0000_0088
19	26	устан.	CAN1_TX	CAN1_TX	0x0000_008C
20	27	устан.	CAN1_RX0	CAN1_RX0	0x0000_0090
21	28	устан.	CAN_RX1	Прерывание CAN1_RX1	0x0000_0094
22	29	устан.	CAN_SCE	Прерывание CAN1_SCE	0x0000_0098
23	30	устан.	EXTI9_5	Прерывания EXTI Line[9:5]	0x0000_009C
24	31	устан.	TIM1_BRK_TIM9	Прерывание TIM1_BRK и TIM9	0x0000_00A0
25	32	устан.	TIM1_UP_TIM10	Прерывание TIM1 Update и TIM10 глобальное	0x0000_00A4
26	33	устан.	TIM1_TRG_COM_ TIM11	Прерывание TIM1 Trigger и TIM11 глобальное	0x0000_00A8
27	34	устан.	TIM1_CC	Прерывание TIM1 Capture Compare	0x0000_00AC
28	35	устан.	TIM2	Глобальное прерывание TIM2	0x0000_00B0
29	36	устан.	TIM3	Глобальное прерывание TIM3	0x0000_00B4
30	37	устан.	TIM4	Глобальное прерывание TIM4	0x0000_00B8
31	38	устан.	I2C1_EV	Событие I^2C1	0x0000_00BC
32	39	устан.	I2C1_ER	Прерывание ошибки I^2C1	0x0000_00C0
33	40	устан.	I2C2_EV	Событие I^2C2	0x0000_00C4
34	41	устан.	I2C2_ER	Прерывание ошибки I^2C2	0x0000_00C8
35	42	устан.	SPI1	Глобальное прерывание SPI1	0x0000_00CC
36	43	устан.	SPI2	Глобальное прерывание SPI2	0x0000_00D0
37	44	устан.	USART1	Глобальное прерывание USART1	0x0000_00D4
38	45	устан.	USART2	Глобальное прерывание USART2	0x0000_00D8
39	46	устан.	USART3	Глобальное прерывание USART3	0x0000_00DC
40	47	устан.	EXTI15_10	Прерывания EXTI Line[15:10]	0x0000_00E0
41	48	устан.	RTCAlarm	Прерывание RTC alarm (A и B) через EXTI	0x0000_00E4
42	49	устан.	OTG_FS_WKUP	Прерывание USB wakeup через EXTI line	0x0000_00E8
43	50	устан.	TIM8_BRK_TIM12	Прерывание TIM8 Break TIM12 глобальное	0x0000_00EC

№	Pri	Тип приор.	Имя	Описание прерывания	Адрес
44	51	устан.	TIM8_UP_TIM13	Прерывание TIM8 Update и TIM13 глобальное	0x0000_00F0
45	52	устан.	TIM8_TRG_COM_ TIM14	Прерывание TIM8 Trigger и TIM14 глобальное	0x0000_00F4
46	53	устан.	TIM8_CC	Прерывание TIM8 Capture Compare	0x0000_00F8
47	54	устан.	DMA1_Stream7	DMA1 Stream7 global interrupt	0x0000_00FC
48	55	устан.	FSMC	Глобальное прерывание FSMC	0x0000_0100
49	56	устан.	SDIO	Глобальное прерывание SDIO	0x0000_0104
50	57	устан.	TIM5	Глобальное прерывание TIM5	0x0000_0108
51	58	устан.	SPI3	Глобальное прерывание SPI3	0x0000_010C
52	59	устан.	UART4	Глобальное прерывание UART4	0x0000_0110
53	60	устан.	UART5	Глобальное прерывание UART5	0x0000_0114
54	61	устан.	TIM6_DAC	Глобальное прерывание TIM6TIM6 global interrupt, DAC1 and DAC2 underrun error interrupts	0x0000_0118
55	62	устан.	TIM7	Глобальное прерывание TIM7	0x0000_011C
56	63	устан.	DMA2_Stream0	DMA2 Stream0 global interrupt	0x0000 0120
57	64	устан.	DMA2_Stream1	DMA2 Stream1 global interrupt	0x0000 0124
58	65	устан.	DMA2_Stream2	DMA2 Stream2 global interrupt	0x0000 0128
59	66	устан.	DMA2_Stream3	DMA2 Stream3 global interrupt	0x0000 012C
60	67	устан.	DMA2_Stream4	DMA2 Stream4 global interrupt	0x0000 0130
61	68	устан.	ETH	Ethernet global interrupt	0x0000 0134
62	69	устан.	ETH_WKUP	Ethernet Wakeup through EXTI line interrupt	0x0000 0138
63	70	устан.	CAN2_TX	CAN2 TX interrupts	0x0000 013C
64	71	устан.	CAN2_RX0	CAN2 RX0 interrupts	0x0000 0140
65	72	устан.	CAN2_RX1	CAN2 RX1 interrupt	0x0000 0144
66	73	устан.	CAN2_SCE	CAN2 SCE interrupt	0x0000 0148
67	74	устан.	OTG_FS	USB On The Go FS global interrupt	0x0000 014C
68	75	устан.	DMA2_Stream5	DMA2 Stream5 global interrupt	0x0000 0150
69	76	устан.	DMA2_Stream6	DMA2 Stream6 global interrupt	0x0000 0154
70	77	устан.	DMA2_Stream7	DMA2 Stream7 global interrupt	0x0000 0158
71	78	устан.	USART6	USART6 global interrupt	0x0000 015C
72	79	устан.	I2C3_EV	I^2C3 event interrupt	0x0000 0160
73	80	устан.	I2C3_ER	I^2C3 error interrupt	0x0000 0164
74	81	устан.	OTG_HS_EP1_OUT	USB On The Go HS End Point 1 Out global interrupt	0x0000 0168
75	82	устан.	OTG_HS_EP1_IN	USB On The Go HS End Point 1 In global interrupt	0x0000 016C
76	83	устан.	OTG_HS_WKUP	USB On The Go HS Wakeup through EXTI interrupt	0x0000 0170
77	84	устан.	OTG_HS	USB On The Go HS global interrupt	0x0000 0174
78	85	устан.	DCMI	DCMI global interrupt	0x0000 0178
79	86	устан.	CRYP	CRYP crypto global interrupt	0x0000 017C

№	Pri	Тип приор.	Имя	Описание прерывания	Адрес
80	87	устан.	HASH_RNG	Hash and Rng global interrupt	0x0000 0180
81	88	устан.	FPU	FPU global interrupt	0x0000 0184
82	89	устан.	UART7	UART 7 global interrupt	0x0000 0188
83	90	устан.	UART8	UART 8 global interrupt	0x0000 018C
84	91	устан.	SPI4	SPI 4 global interrupt	0x0000 0190
85	92	устан.	SPI5	SPI 5 global interrupt	0x0000 0194
86	93	устан.	SPI6	SPI 6 global interrupt	0x0000 0198
87	94	settable	SAI1	SAI1 global interrupt	0x0000 019C
88	95	устан.	LCD-TFT	LTDC global interrupt	0x0000 01A0
89	96	устан.	LCD-TFT	LTDC global Error interrupt	0x0000 01A4
90	97	устан.	DMA2D	DMA2D global interrupt	0x0000 01A8

12.2. Контроллер внешних прерываний/событий (EXTI)

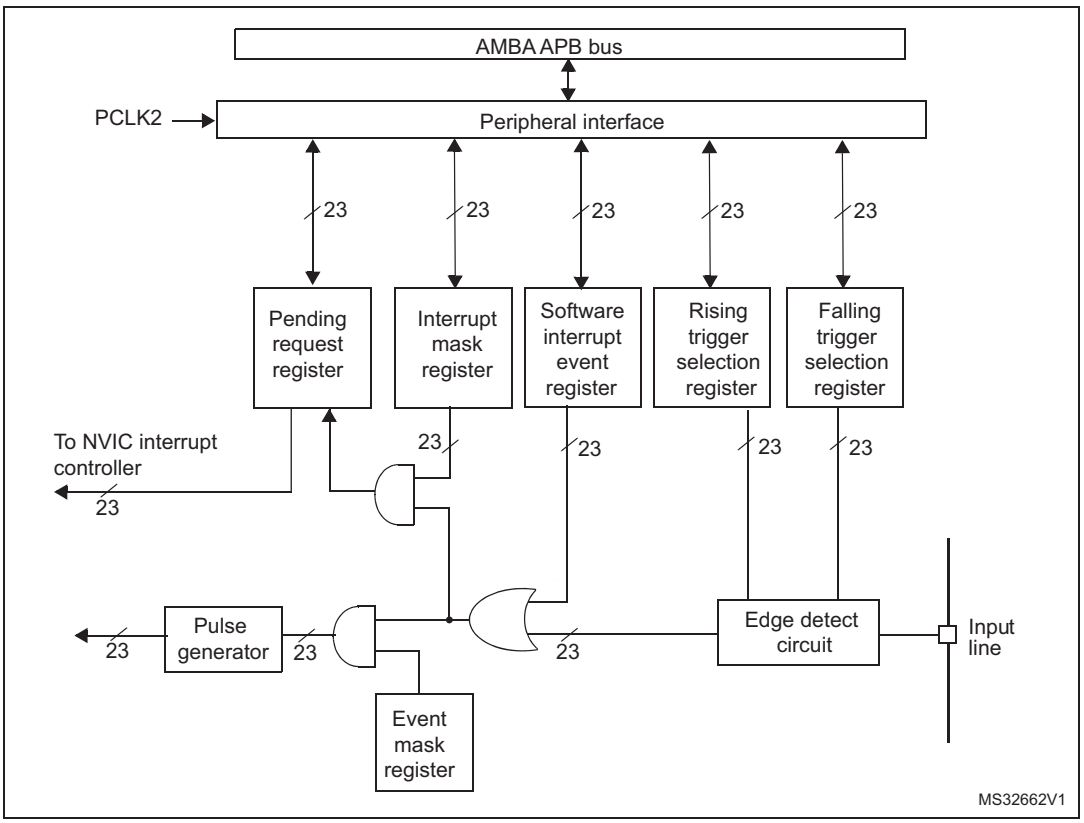
Контроллер внешних прерываний/событий содержит до 23 детекторов фронта запроса событий/прерываний. Каждая входная линия может быть независимо установлена на выбор типа (событие или прерывание) и тип фронта (передний, задний или оба). Маскируются линии независимо. Для запросов прерывания есть регистр удержания.

12.2.1. Основные свойства

Вот они:

- Независимый триггер и маска для каждой линии
- Специальный бит статуса для каждой линии прерывания
- Возбуждение до 23 программных запросов событий/прерываний
- Обнаружение внешних сигналов короче такта APB2. См. описания устройств.

12.2.2. Блок-схема



12.2.3. Событие побудки

Разбудить ядро STM32F4xx из состояния WFE можно изнутри и снаружи:

- разрешив прерывание в регистре управления периферии (но не в NVIC) и разрешив бит SEVONPEND в регистре управления системы. При выходе из WFE надо очистить биты удержания в периферии и в NVIC IRQ.
- поставив внешнюю или внутреннюю линию EXTI в режим события. При выходе из WFE биты удержания в периферии и в NVIC IRQ чистить не надо (их там и нет).

12.2.4. Функциональное описание

Если захотелось прерывания, то его надо сконфигурировать и разрешить двумя регистрами триггеров, определив желаемый фронт и записав "1" в маску. При появлении на внешней линии прерывания нужного фронта генерируется прерывание и ставится бит удержания нужной линии прерывания. Запрос сбрасывается записью ‘1’ в регистр удержания.

Если захотелось события, то его линию надо сконфигурировать и разрешить двумя регистрами триггеров, определив желаемый фронт и записав "1" в регистр маски события. При появлении на линии события нужного фронта генерируется импульс события, но бит удержания линии события не ставится.

Программно событие/прерывание можно вызвать записью ‘1’ в регистр программного прерывания/события.

Аппаратное прерывание.

Дабы получить прерывание от 23 линий внешних источников надо:

- Поставить биты маски 23 линий прерывания (EXTI_IMR)
- Поставить биты Выбора Запуска линий прерывания (EXTI_RTSR и EXTI_FTSR)
- Поставить биты разрешения и маски канала NVIC IRQ, отведённого для контроллера EXTI, чтобы поступающее по одной из 23 линий прерывание было правильно обработано.

Аппаратное событие.

Чтобы получить событие от 23 линий внешних источников надо:

- Поставить биты маски 23 линий события (EXTI_EMR)
- Поставить биты Выбора Запуска линий события (EXTI_RTSR и EXTI_FTSR)

Программное событие/прерывание.

По этим 23 линиям можно получить и программное событие/прерывание:

- Поставить биты маски 23 линий события/прерывания (EXTI_IMR, EXTI_EMR)
- Поставить нужный бит в регистре программного прерывания (EXTI_SWIER)

12.2.5. Подключение линий внешних событий/прерываний

К 16 лииям можно подключить до 140 GPIO (STM32F405xx/07xx и STM32F415xx/17xx) и 168 GPIO (STM32F42xxx и STM32F43xxx):

Рис. 42. Подключение GPIO для STM32F405xx/07xx и STM32F415xx/17xx

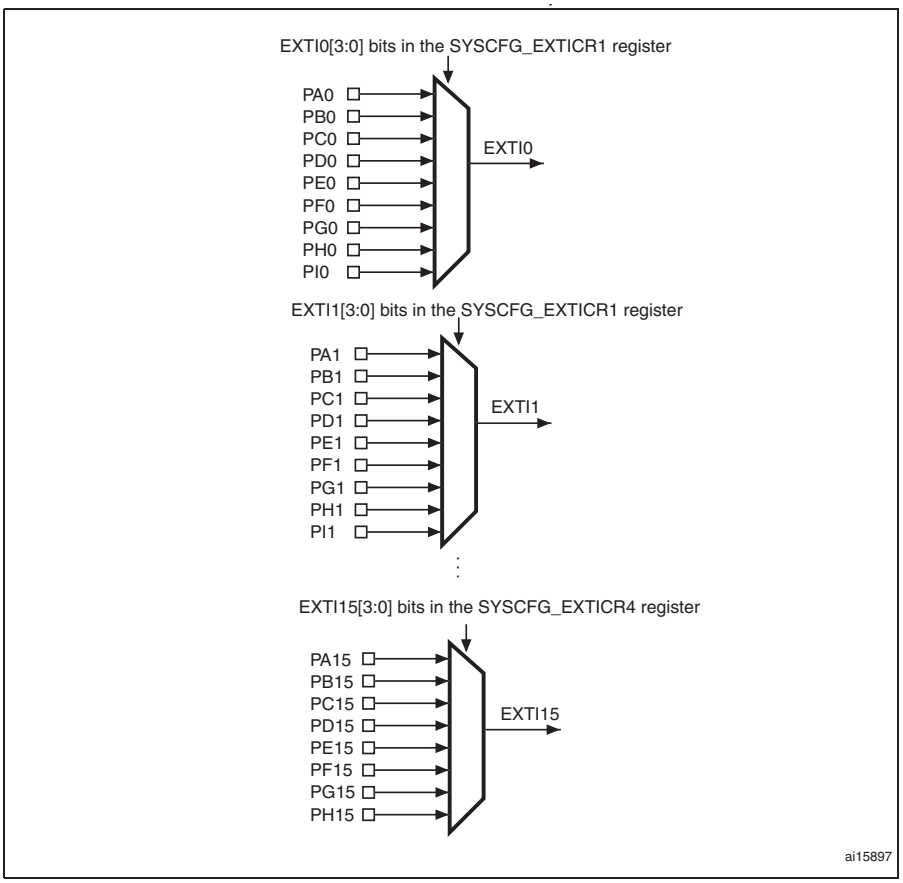
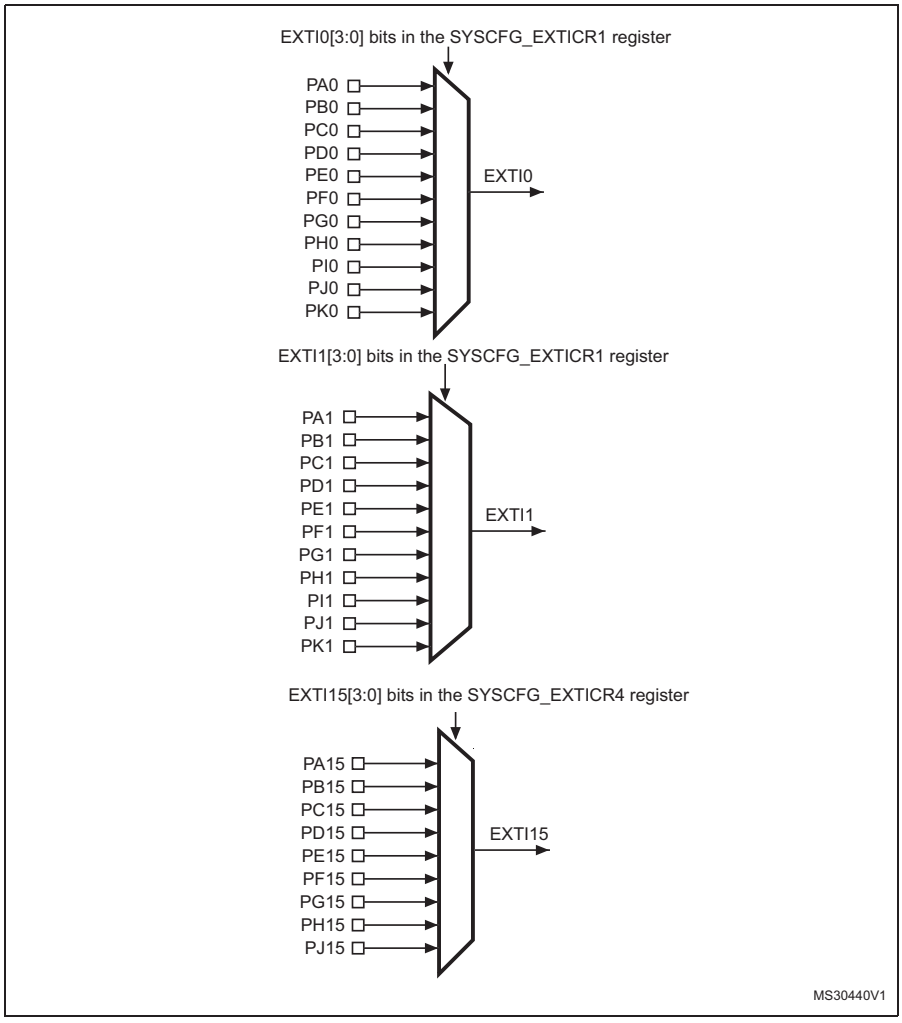


Рис. 43. Подключение GPIO для STM32F42xxx и STM32F43xxx



Семь остальных линий EXTI подключены так:

- EXTI line 16 подключена к выходу PVD
- EXTI line 17 подключена к событию RTC Alarm
- EXTI line 18 подключена к событию USB OTG FS Wakeup
- EXTI line 19 подключена к событию Ethernet Wakeup
- EXTI line 20 подключена к событию USB OTG HS (в режиме in FS) Wakeup
- EXTI line 21 подключена к событиям RTC Tamper и TimeStamp
- EXTI line 22 подключена к событию RTC Wakeup

12.3. Регистры EXTI

12.3.1. Регистр маски прерывания (EXTI_IMR)

Смещение адреса: 0x00

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									MR22	MR21	MR20	MR19	MR18	MR17	MR16
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:23 Резерв, не трогать.

— Биты 22:0 **MRx:** Маска прерывания линии x

0: Запрос прерывания на Line x закрыт

1: Запрос прерывания на Line x открыт

12.3.2. Регистр маски события (EXTI_EMR)

Смещение адреса: 0x04

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									MR22	MR21	MR20	MR19	MR18	MR17	MR16
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:23 Резерв, не трогать.

— Биты 22:0 **MRx:** Маска события на линии x

0: Запрос события на Line x закрыт

1: Запрос события на Line x открыт

12.3.3. Регистр выбора переднего фронта запуска (EXTI_RTSR)

Смещение адреса: 0x08

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									TR22	TR21	TR20	TR19	TR18	TR17	TR16
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:23 Резерв, не трогать.

— Биты 22:0 **TRx:** Маска события/прерывания по переднему фронту на линии **x**

0: Запрос события/прерывания на линии **x** закрыт

1: Запрос события/прерывания на линии **x** открыт

NB: Внешние линии побудки запускаются фронтом, так что глитчи здесь недопустимы. Если передний фронт появляется во время записи в EXTI_RTSR, то бит удержания не ставится.

Одна линия может срабатывать по обоим фронтам сигнала, регистров разрешения то два.

12.3.4. Регистр выбора заднего фронта запуска (EXTI_RTSR)

Смещение адреса: 0x0C

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									TR22	TR21	TR20	TR19	TR18	TR17	TR16
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:23 Резерв, не трогать.

— Биты 22:0 **TRx:** Маска события/прерывания по заднему фронту на линии **x**

0: Запрос события/прерывания на линии **x** закрыт

1: Запрос события/прерывания на линии **x** открыт

NB: Внешние линии побудки запускаются фронтом, так что глитчи здесь недопустимы. Если задний фронт появляется во время записи в EXTI_RTSR, то бит удержания не ставится.

Одна линия может срабатывать по обоим фронтам сигнала, регистров разрешения то два.

12.3.5. Регистр программного события/прерывания (EXTI_SWIER)

Смещение адреса: 0x10

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									SWIER 22	SWIER 21	SWIER 20	SWIER 19	SWIER 18	SWIER 17	SWIER 16
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWIER 15	SWIER 14	SWIER 13	SWIER 12	SWIER 11	SWIER 10	SWIER 9	SWIER 8	SWIER 7	SWIER 6	SWIER 5	SWIER 4	SWIER 3	SWIER 2	SWIER 1	SWIER 0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:23 Резерв, не трогать.

— Биты 22:0 **SWIERx:** Программное прерывание на линии **x**

Если прерывание на линии разрешено в регистре EXTI_IMR, то запись '1' в ранее чистый бит этого регистра (передний фронт) ставит соответствующий бит в регистре EXTI_PR и соответственно просит о прерывании. Бит снимается чисткой соответствующего бита в регистре EXTI_PR.

12.3.6. Регистр удержания (EXTI_PR)

Смещение адреса: 0x14

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									PR22	PR21	PR20	PR19	PR18	PR17	PR16
									rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PR15	PR14	PR13	PR12	PR11	PR10	PR9	PR8	PR7	PR6	PR5	PR4	PR3	PR2	PR1	PR0
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

— Биты 31:20 Резерв, не трогать.

— Биты 19:0 **PRx:** Бит удержания на линии **x**

0: Запроса не было.

1: Запрос был.

Стирается записью "1" в соответствующий бит.

12.3.7.Карта регистров EXTI

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**EXTI_IMR**	Reserved									MR[22:0]																						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**EXTI_EMR**	Reserved									MR[22:0]																						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**EXTI_RTSR**	Reserved									TR[22:0]																						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**EXTI_FTSR**	Reserved									TR[22:0]																						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**EXTI_SWIER**	Reserved									SWIER[22:0]																						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**EXTI_PR**	Reserved									PR[22:0]																						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

13. Аналого-цифровой преобразователь (ADC)

13.1. Введение в АЦП

АЦП последовательного приближения имеет до 19 мультиплексируемых каналов для 16 внешних, 2 внутренних сигналов и V_{BAT} . A/D преобразование разных каналов может быть однократным, непрерывным, сканирующим или прерывным. Результат преобразования пишется в 16-бит регистр, выравненным вправо или влево.

Аналоговый сторож может отслеживать входной аналоговый сигнал на выход за заданные верхний и нижний пределы.

Входные такты АЦП получаются из PCLK2 после предделителя. Частота не должна превышать 14MHz.

13.2. Основные свойства АЦП

- Разрешение 12-бит, 10-бит, 8-бит или 6-бит
- Прерывание по Концу преобразования, Концу вставного преобразования и событию от Аналогового сторожа или переполнения
- Однократный и непрерывный режимы преобразования
- Автоматический режим сканирования каналов от 0 до ‘n’
- Выравнивание данных с встроенной когеррентностью.
- Программируемое время выборки между каналами
- Опция внешнего запуска регулярного и вставного преобразования
- Прерывное преобразование
- Парный/Тройной режим (в устройствах с 2 АЦП или более)
- Запись данных через DMA в Парном/Тройном режиме
- Программируемая задержка между преобразованиями в Парном/Тройном чересстрочном режиме
- Тип преобразования ADC (см. описания)
- Питание АЦП: 2.4 V до 3.6 V на полной скорости и до 1.8 V на малой
- Входной диапазон: $V_{REF-} \leq V_{IN} \leq V_{REF+}$
- Запрос DMA во время регулярного преобразования канала

13.3. Функциональное описание АЦП

Блок-схема АЦП

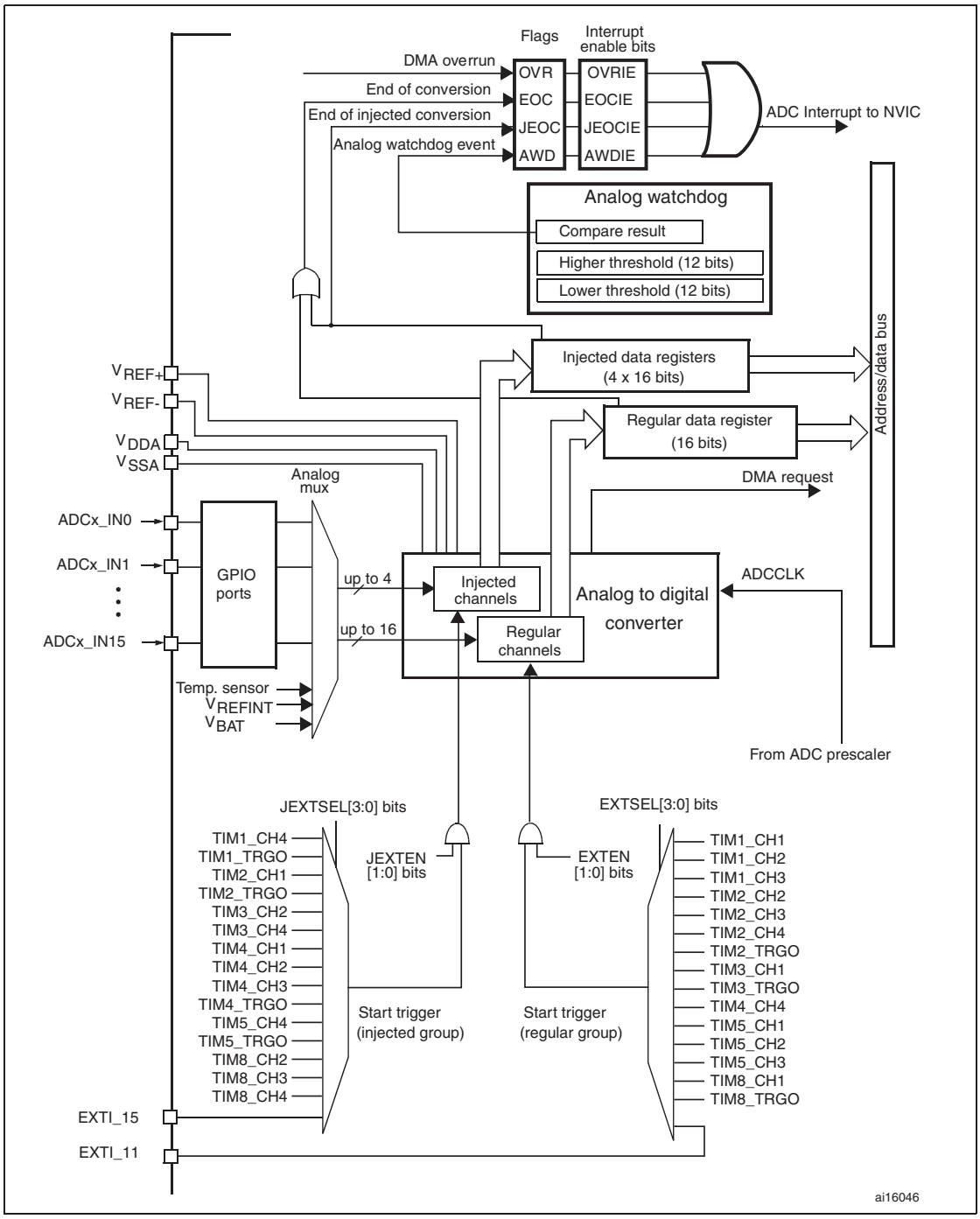


Таблица 65. Ножки АЦП.

Имя	Тип сигнала	Заметки
V_{REF+}	Положительное опорное напряжение	1.8 V ≤V_{REF+} ≤V_{DDA}
V_{DDA}	Аналоговое питание	V_{DD}, 2.4 V ≤V_{DDA} ≤3.6 V на полной скорости 1.8 V ≤V_{DDA} ≤3.6 V на малой скорости
V_{REF-}	Отрицательное опорное напряжение	V_{REF-} = V_{SSA}
V_{SSA}	Аналоговая земля	Эквивалентна V_{SS}
ADCx_IN[15:0]	Аналоговые сигналы	До 16 аналоговых каналов

13.3.1. Вкл./Выкл. АЦП

АЦП включается установкой бита ADON в регистре ADC_CR2. При установке ADON в первый раз АЦП просыпается из режима Power Down.

Преобразование начинается установкой бита SWSTART или JSWSTART.

Преобразование можно остановить сбросом бита ADON. В этом режиме АЦП почти не кушает (несколько μA).

13.3.2. Такты АЦП

Есть две схемы тактирования:

- Для аналоговой части: ADCCLK, общая для всех ADC
  Получают делением тактов APB2 на предделитель (f_{PCLK2}/2, /4, /6 или /8. см. описания).
- Для цифровой части (чтение/запись регистров)
  Это такты APB2. Включают ADC отдельно в регистре RCC_APB2ENR.

13.3.3. Выбор канала

Есть 16 мультиплексируемых каналов. Можно организовать преобразования в две группы: регулярную и вставную. Группа состоит из последовательности преобразований любых каналов в любом порядке. Например, ADC_IN3, ADC_IN8, ADC_IN2, ADC_IN2, ADC_IN0, ADC_IN2, ADC_IN2, ADC_IN15.

- **Регулярная группа** содержит до 16 преобразований. Каналы и их порядок выбираются в регистрах ADC_SQRx. Общее число преобразований пишется в биты L[3:0] регистра ADC_SQR1.
- **Вставная группа** содержит до 4 преобразований. Каналы и их порядок выбираются в регистре ADC_JSQR. Общее число преобразований пишется в биты L[1:0] регистра ADC_JSQR.

Если регистры ADC_SQRx или ADC_JSQR изменяются во время преобразования, то текущее преобразование прерывается и выдаётся импульс запуска преобразования новой группы.

Внутренние каналы Датчик температуры, внутренние каналы V_{REFINT} и V_{BAT}

- В STM32F40x и STM32F41x датчик температуры подключён к ADCx_IN16, а внутреннее опорное напряжение V_{REFINT} к ADCx_IN17.
- В STM32F42x и STM32F43x датчик температуры подключён к ADC1_IN18, общему с V_{BAT}. Так что одновременно можно мерять только один из них. При одновременной установке обоих меряется только V_{BAT}. Напряжение V_{REFINT} подключено к ADC1_IN17.

V_{BAT} можно измерять как вставной или регулярный канал.
NB: Датчик температуры, V_{REFINT} и V_{BAT} доступны только на ведущем ADC1.

13.3.4. Одинарное преобразование

Запускается при бите CONT равном 0 с помощью:

- установкой бита SWSTART в регистре ADC_CR2 (только для регулярных каналов)
- установкой бита JSWSTART
- внешним запуском (для регулярных и вставных каналов).

При завершении преобразования:

- Регулярного канала:
  - Результат пишется в 16-бит регистр ADC_DR
  - Ставится флаг EOC (Конец Преобразования)
  - Если бит EOCIE стоит, то генерируется прерывание.
- Вставного канала:
  – Результат пишется в 16-бит регистр ADC_JDR1
  – Ставится флаг JEOC (Конец Вставного Преобразования)
  – Если бит JEOCIE стоит, то генерируется прерывание.

АЦП останавливается.

13.3.5. Непрерывное преобразование

В этом режиме преобразования выполняются одно за другим. Запускается внешним сигналом или установкой бита SWSTRT в регистре ADC_CR2, а бит CONT равен 1 (только регулярных каналов).

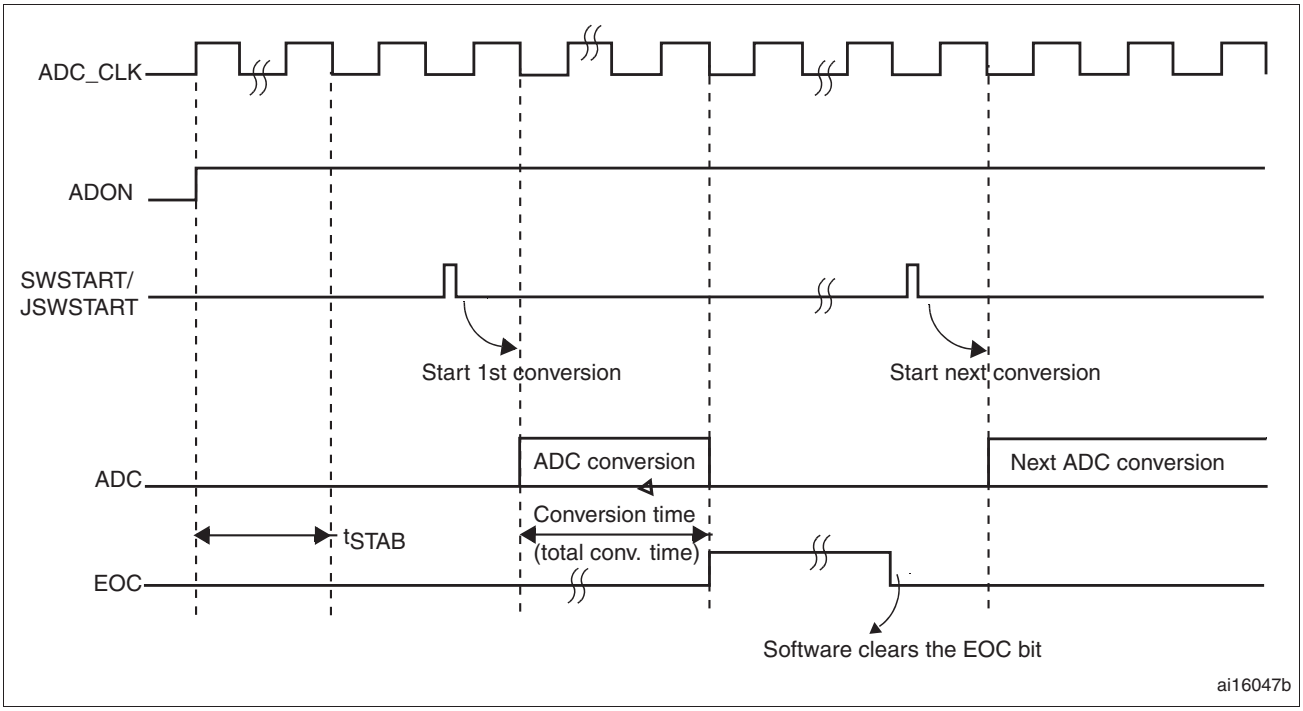
После каждого преобразования:

- Регулярного канала:
  - Результат пишется в 16-бит регистр ADC_DR
  - Ставится флаг EOC (Конец Преобразования)
  - Если бит EOCIE стоит, то генерируется прерывание.

NB: Вставные каналы не могут преобразовываться непрерывно. Исключение: битом JAUTO вставной канал можно добавить в конец непрерывного преобразования регулярных каналов, см. секцию Авто-вставка.

13.3.6. Временная диаграмма

Перед преобразованием АЦП нуждается в стабилизации (t_{STAB}). Через 15 тактов после запуска преобразования ставится флаг EOC и результат пишется в регистр ADC_DR.



13.3.7. Аналоговый сторож

Если результат АЦП выходит за верхний и нижний порог, то аналоговый сторож AWD ставит бит состояния. Пороги указываются в младших 12 битах 16-бит регистров ADC_HTR и ADC_LTR. Прерывание разрешается битом AWDIE в регистре ADC_CR1.

Значение порога не зависит от выравнивания, указанного в бите ALIGN регистра ADC_CR2. Сравнение делается раньше.

Аналоговый сторож может быть разрешён для одного или всех каналов в регистре ADC_CR1.

Охраняемая зона

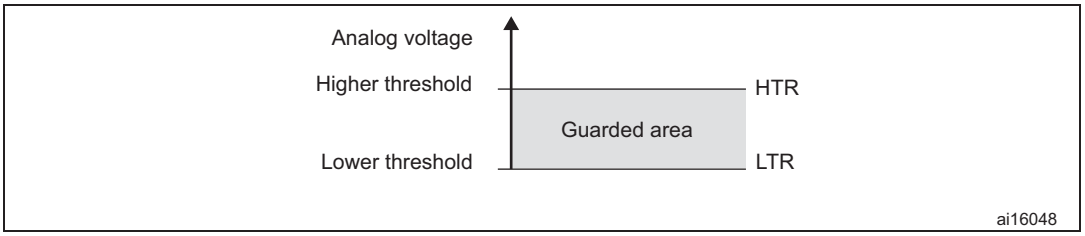


Таблица 66. Выбор каналов для охраны

Охраняемые каналы	Биты регистра ADC_CR1 (x = без разницы)		
	AWDSGL	AWDEN	JAWDEN
Нету	x	0	0
Все вставные	0	0	1
Все регулярные	0	1	0
Все регулярные и вставные	0	1	1
Один(1) вставной	1	0	1
Один(1) регулярный	1	1	0
Один(1) регулярный или вставной	1	1	1

1. Выбирается битами AWDCH[4:0]

13.3.8. Режим сканирования

Сканируется группа аналоговых каналов.

Включается установкой бита SCAN в регистре ADC_CR1. Тогда АЦП сканирует все каналы, указанные в регистрах ADC_SQRx (для регулярных) или ADC_JSQR (для вставных). Преобразование последовательно выполняется для всех каналов группы. Если стоит бит CONT, то преобразование выполняется снова и снова для всей группы.

В этом режиме можно ставить бит DMA для записи результатов в SRAM после каждого обновления регистра ADC_DR через контроллер DMA.

Бит EOC в регистре ADC_SR ставится:

- При снятом бите EOCS в конце преобразования каждой регулярной группы каналов
- При стоящем бите EOCS в конце преобразования каждого регулярного канала

Данные вставных каналов всегда пишутся в регистры ADC_JDRx.

13.3.9. Управление вставными каналами

Запускаемая вставка

В этом случае в регистре ADC_CR1 нужно очистить бит JAUTO.

1. Внешним сигналом или битом SWSTART в регистре ADC_CR2 запустить преобразование регулярной группы каналов.
2. Если во время обработки этой группы появляется внешний сигнал вставного запуска или ставится бит JSWSTART, то текущее преобразование останавливается и обрабатывается вставная последовательность в режиме однократного сканирования.
3. Затем восстанавливается регулярная последовательность с прерванного преобразования. Если во время вставной последовательности появляется запуск регулярной, то он не прерывает вставной, и регулярная выполняется после завершения вставной.

NB: При использовании запускаемой вставки интервал между событиями запуска должен быть больше вставляемой последовательности. Например, если длина последовательности равна 30 тактов ADC (2 преобразования и 3 такта времени выборки), то минимальный интервал между запусками должен быть не менее 31 тактов ADC.

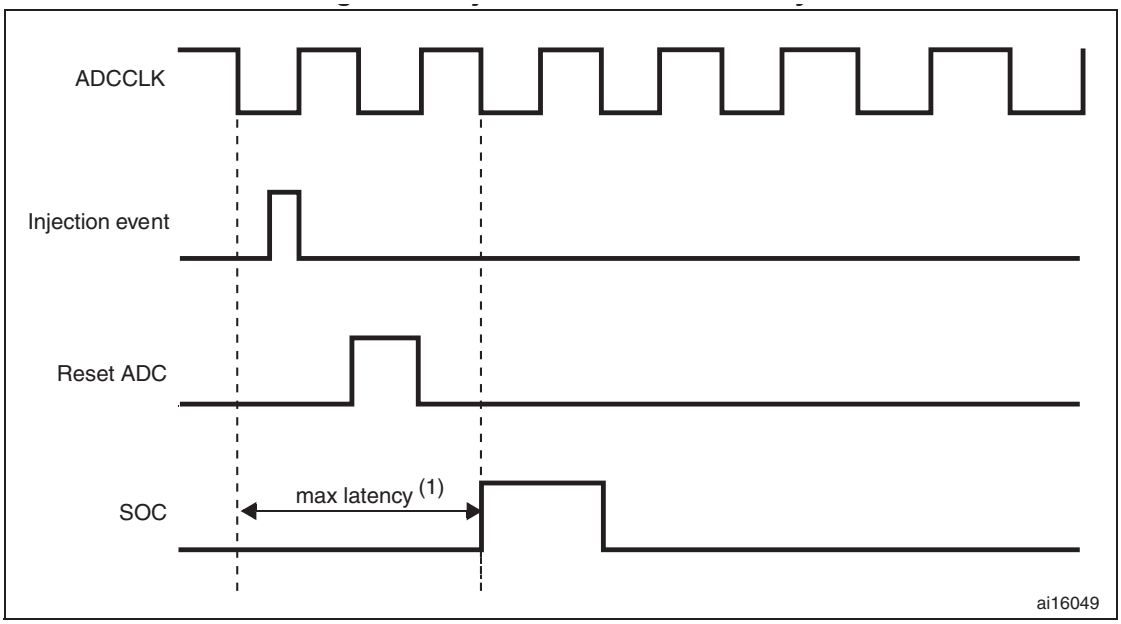
Автовставка

Если бит JAUTO стоит, то вставная группа каналов автоматически обрабатывается после завершения регулярной группы. Так можно выполнить до 20 преобразований, отмеченных в регистрах ADC_SQRx и ADC_JSQR. В этом режиме внешний запуск вставных каналов должен быть выключен. А если ещё стоит и бит CONT, то вся последовательность выполняется постоянно.

При значении прескалеров ADC от 4 до 8 для переключения между любыми последовательностями (регулярная/вставная) добавляется задержка в 1 такт. При предделителе ADC равном 2 и задержка равна 2 тактам ADC.

Автовставку и прерывный режим нельзя использовать одновременно.

Рис. 47. Задержки вставной последовательности.



1. Значение максимальной задержки приведено в описаниях STM32F40x и STM32F41x.

13.3.10.Прерывный режим

Регулярная группа

Разрешается установкой бита DISCEN в регистре ADC_CR1. Используется для обработки короткой последовательности преобразований (**n** <=8), записанной в регистрах ADC_SQRx. Значение **n** записано в битах DISCNUM[2:0] регистра ADC_CR1.

Внешний сигнал запускает следующие **n** преобразований последовательности из регистров ADC_SQRx пока не будет обработана вся последовательность. Общая длина последовательности определена в битах L[3:0] регистра ADC_SQR1.

Пример:

n = 3, преобразуемые каналы = 0, 1, 2, 3, 6, 7, 9, 10

1й запуск: преобразуются 0, 1, 2. Событие EOC выдаётся на каждое преобразование

2й запуск: преобразуются 3, 6, 7. Событие EOC выдаётся на каждое преобразование

3й запуск: преобразуются 9, 10. Событие EOC выдаётся на каждое преобразование

4й запуск: преобразуются 0, 1, 2. Событие EOC выдаётся на каждое преобразование

NB. При обработке регулярных групп заворота последовательности через начало не бывает и по завершению всей последовательности следующий запуск начинается с первого канала. См. пример ранее.

Вставная группа

Разрешается установкой бита JDISCEN в регистре ADC_CR1. Используется для обработки последовательности каналов, записанной в регистрах ADC_JSQR, один за другим после внешнего запуска.

Внешний сигнал запускает обработку следующего канала из регистров ADC_JSQR вплоть до конца. Общая длина последовательности определена в битах JL[1:0] регистра ADC_JSQR.

Пример:

n = 1, преобразуемые каналы = 1, 2, 3

1й запуск: канал 1

2й запуск: канал 2

3й запуск: канал 3, выдаётся событие JEOC

4й запуск: канал 1

NB. При завершении вставной последовательности очередной сигнал запускает преобразование первого в последовательности.

Прерывный режим и автовставку нельзя использовать одновременно.

Прерывный режим нужно использовать только для одной из групп (регулярной или вставной).

13.4. Выравнивание данных

Выравнивание данных вправо или влево выбирается битом ALIGN в регистре ADC_CR2.

Данные вставных каналов имеют знак относительно смещения, записанного в регистр ADC_JOFRx и разрядность уменьшается на бит знака. Бит SEXT это расширенный знак.

Данные регулярных каналов беззнаковые и 12-разрядные.

Рис. 48. Правое выравнивание 12-бит данных.

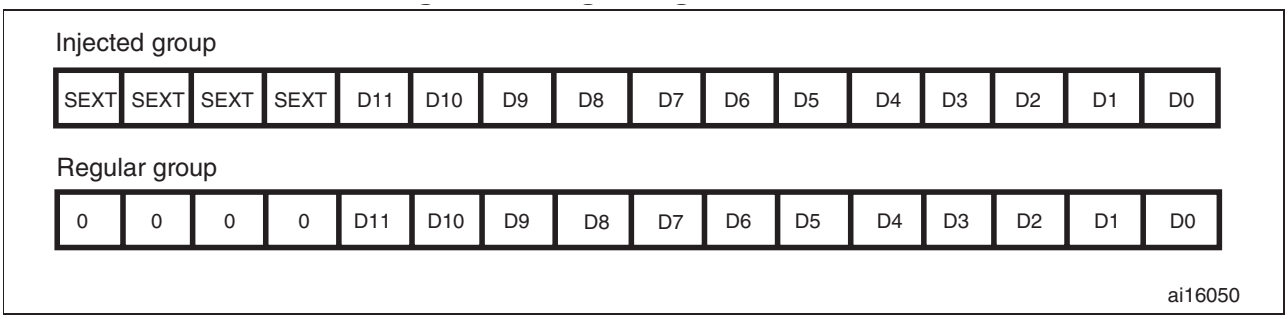
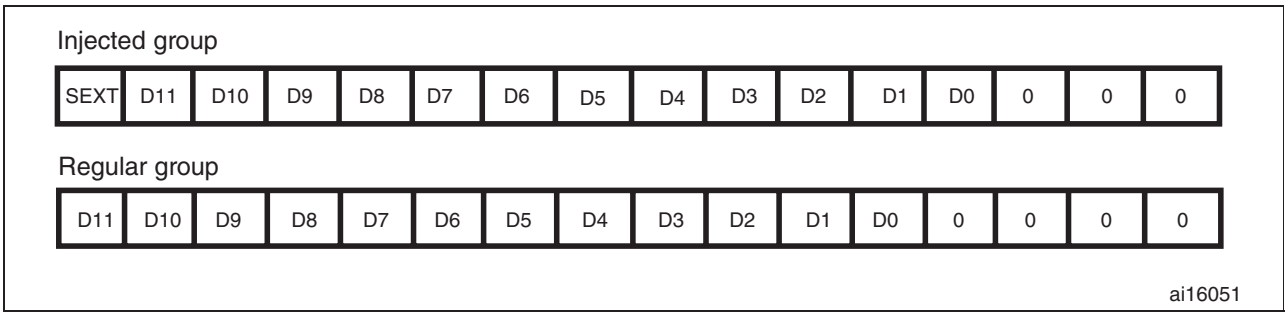
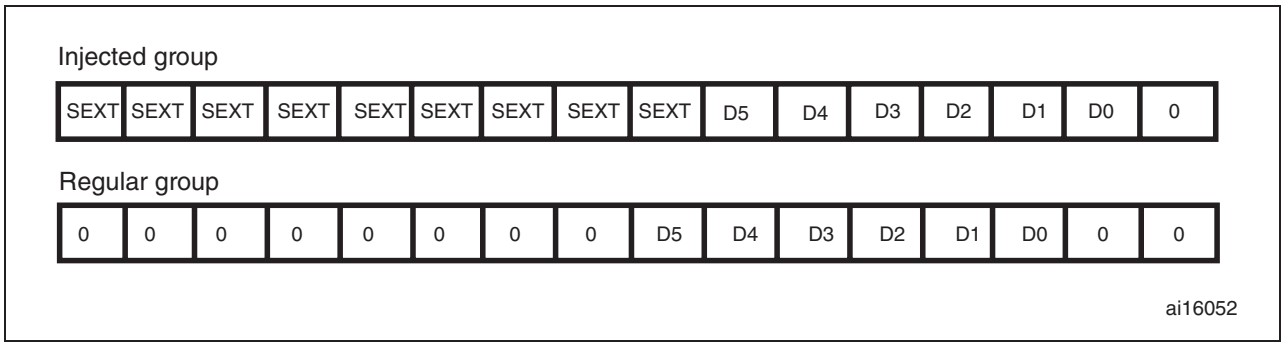


Рис. 49. Левое выравнивание 12-бит данных.



При левом выравнивании, данные выравниваются на основе полуслова, но при 6-бит разрешении он выравниваются на основе байта.

Рис. 50. Левое выравнивание 6-бит данных.



13.5. Время выборки каналов

Время выборки входного сигнала каждого канала АЦП задаётся числом тактов ADC_CLK , определяемых битами SMP[2:0] в регистрах ADC_SMPR1 и ADC_SMPR2.

Общее время преобразования равно:

T_{conv} = Время выборки + 12 тактов

Пример:

При ADCCLK = 30 MHz и времени выборки 3 такта:

T_{conv} = 3 + 12 = 15 тактов = 0.5 μs при APB2 на 60 MHz.

13.6. Внешний запуск преобразования и его полярность

Внешний запуск преобразования с заданной полярностью сигнала возможен если биты EXTEN[1:0] (для регулярного преобразования) или JEXTEN[1:0] (для вставного преобразования) отличны от “0b00”.

Таблица 67. Полярность запуска.

Источник	EXTEN[1:0] / JEXTEN[1:0]
Запуск запрещён	00
Передний фронт	01
Задний фронт	10
Оба фронта	11

NB: Полярность внешнего запуска можно менять налету.

Биты EXTSEL[3:0] и JEXTSEL[3:0] выбирают один из 16 возможных событий запуска преобразования регулярных и вставных групп.

Таблица 68. Внешний запуск регулярных каналов.

Источник	Тип	EXTSEL[3:0]
TIM1_CH1 event	Внутренний сигнал от встроенных таймеров	0000
TIM1_CH2 event		0001
TIM1_CH3 event		0010
TIM2_CH2 event		0011
TIM2_CH3 event		0100
TIM2_CH4 event		0101
TIM2_TRGO event		0110
TIM3_CH1 event		0111
TIM3_TRGO event		1000
TIM4_CH4 event		1001
TIM5_CH1 event		1010
TIM5_CH2 event		1011
TIM5_CH3 event		1100
TIM8_CH1 event		1101
TIM8_TRGO event		1110
EXTI line11	Внешняя ножка	1111

Таблица 69. Внешний запуск вставных каналов.

Источник	Тип	JEXTSEL[3:0]
TIM1_CH4 event	Внутренний сигнал от встроенных таймеров	0000
TIM1_TRGO event		0001
TIM2_CH1 event		0010
TIM2_TRGO event		0011
TIM3_CH2 event		0100
TIM3_CH4 event		0101
TIM4_CH1 event		0110
TIM4_CH2 event		0111
TIM4_CH3 event		1000
TIM4_TRGO event		1001
TIM5_CH4 event		1010
TIM5_TRGO event		1011
TIM8_CH2 event		1100
TIM8_CH3 event		1101
TIM8_CH4 event		1110
EXTI line15	Внешняя ножка	1111

Запуск может быть программным установкой битов SWSTART или JSWSTART в ADC_CR2.

Преобразование регулярной группы может быть прервано запуском вставной.

Выбор запуска можно менять налету. Но при изменении выбора есть задержка в 1 такт APB во время которой обнаружение запуска выключено во избежание ложного запуска при передаче.

13.7. Режим быстрого преобразования

Уменьшая разрешение битами RES можно ускорить преобразование. Вот времена:

- 12 бит: 3 + 12 = 15 тактов ADCCLK
- 10 бит: 3 + 10 = 13 тактов ADCCLK
- 8 бит: 3 + 8 = 11 тактов ADCCLK
- 6 бит: 3 + 6 = 9 тактов ADCCLK

13.8. Управление данными

13.8.1. Использование DMA

Поскольку результаты регулярного канала пишутся в один регистр, то при работе нескольких каналов надо бы использовать DMA. Так данные в ADC_DR не пропадут.

При включённом DMA (стоит бит DMA в регистре ADC_CR2) после преобразования каждого канала выдаётся запрос DMA. А он уже передаст из ADC_DR в память.

Не смотря не это при потере данных (переполнении) в регистре ADC_SR ставится бит OVR и при разрешении битом OVRIE выдаётся прерывание. Передачи DMA выключаются. В этом случае при появлении запроса DMA работающее регулярное преобразование прекращается и дальнейшие запуски игнорируются. Теперь необходимо очистить флаг OVR и бит DMAEN использованного потока DMA и заново инициализировать и DMA и ADC. Можно запускать снова. На преобразование вставных каналов переполнение не влияет.

При появлении OVR = 1 в режиме DMA его запросы блокируются после записи последнего полученные данного и всё данные, попавшие в память, достоверны.

По концу передач DMA (числа из регистра DMA_SxNTR):

- Если бит DDS в регистре ADC_CR2 снят, то запросы к DMA не выдаются. Но бит DMA аппаратно не снимается. Для возобновления передач его надо снять и поставить снова.
- При стоящем бите DDS запросы DMA продолжат приниматься двухбуферного режима для.

Для вывода ADC из состояния переполнения надо:

1. Снова инициализировать DMA (установить адрес получателя и счётчик NDTR)
2. Снять бит OVR в регистре ADC_SR
3. Запустить ADC.

13.8.2. Работа без DMA

При достаточно медленных преобразованиях их можно обслуживать программно. В этом случае надо ставить бит EOCS в регистре ADC_CR2 чтобы бит EOC вставал после каждого преобразования, а не только в конце последовательности. При EOCS = 1 автоматически включается обнаружение переполнения. По каждому появлению бита EOC надо читать регистр ADC_DR. Обработка переполнения такая же, как и с DMA.

Для вывода ADC из состояния переполнения надо:

1. Снять бит OVR в регистре ADC_SR
2. Запустить ADC.

13.8.3. Работа без DMA и переполнения

Иногда бывает полезно читать данные от ADC не по каждому преобразованию (например, сторожевой таймер). Для этого выключают DMA (DMA = 0) и бит EOC надо ставить в конце последовательности (EOCS = 0). Тогда переполнения не видят.

13.9. Режим нескольких АЦП

При наличии нескольких АЦП их можно запускать как ведущий ADC1 и ведомые ADC2 и ADC3 попеременно или одновременно в зависимости от битов MULTI[4:0] в регистре ADC_CCR.

NB: Если в парном режиме преобразование надо запускать внешним сигналом, то извне надо запускать только ведущий, а ведомые программно, во избежание ложного запуска ведомых.

Есть четыре возможных режима:

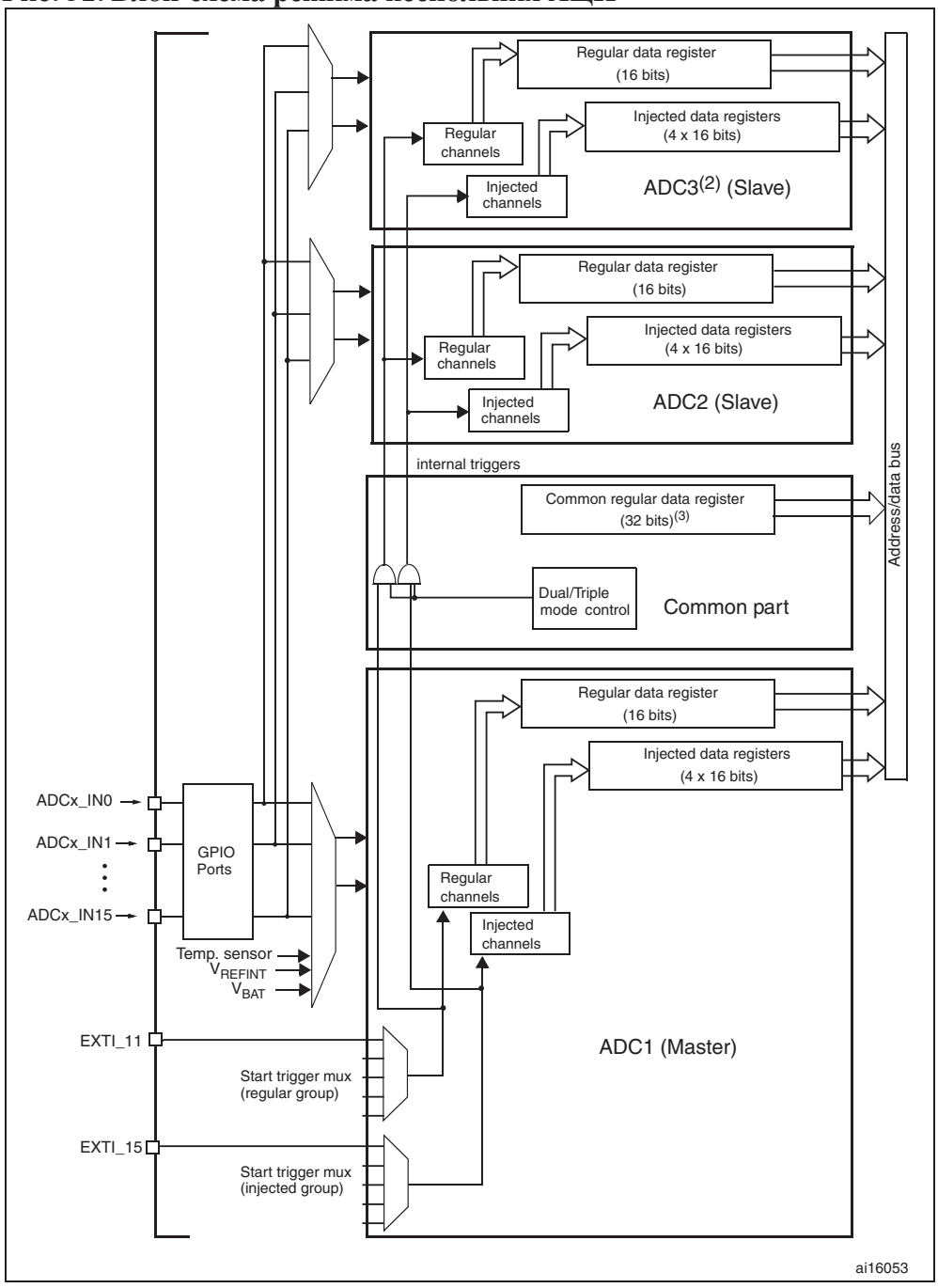
- Вставной одновременный
- Регулярный одновременный
- Чередующийся
- Попеременный запуск

Их можно сочетать:

- Вставной одновременный + Регулярный одновременный
- Регулярный одновременный + Переменный запуск

NB: В режиме нескольких АЦП читать данные надо из регистра ADC_CDR, а биты статуса из регистра ADC_CSR.

Рис. 51. Блок-схема режима нескольких АЦП(1)



1. Внешний запуск ADC2 и ADC3 на схеме не указаны.
2. В парном режиме ADC блока ведомого ADC3 нет.
3. В тройном режиме ADC регистр ADC_CDR) содержит данные регулярных ADC1, ADC2 и ADC3. Все 32 бита используются в соответствии с выбранным порядком. В парном режиме ADC_CDR содержит данные регулярных и ADC1 и ADC2. Используются все 32 бита регистра.

• Запросы DMA в режиме нескольких ADC:

Передача может быть в трёх режимах:

– **Режим DMA 1**: По каждому запросу DMA (есть одно данное) передаётся полуслово.

В тройном режиме по трём запросам данные передаются в порядке ADC1, ADC2 и ADC3 (ADC_CDR[31:0] = ADCx_DR[15:0]). Последовательность повторяется. **Режим DMA 1** используется только в регулярном тройном режиме.

– **Режим DMA 2**: По каждому запросу DMA (есть два данных) в регистре передаётся слово из двух полуслов.

В парном режиме ADC2 лежит в старшем полуслове, ADC1 в младшем.

В тройном режиме выдаются три запроса DMA. По первому запросу ADC2 лежит в старшем полуслове, ADC1 в младшем. По второму запросу ADC1 лежит в старшем полуслове, ADC3 в младшем. По третьему запросу ADC3 лежит в старшем полуслове, ADC2 в младшем.
Режим DMA 2 используется в чередующемся и регулярном одновременном режиме (только для двух ADC).

Пример:

a) Чередующийся парный режим: запрос DMA выдаётся на каждые 2 данных
1й запрос: ADC_CDR[31:0] = ADC2_DR[15:0] | ADC1_DR[15:0]
2й запрос: ADC_CDR[31:0] = ADC2_DR[15:0] | ADC1_DR[15:0]

b) Чередующийся тройной режим: запрос DMA выдаётся на каждые 2 данных
1й запрос: ADC_CDR[31:0] = ADC2_DR[15:0] | ADC1_DR[15:0]
2й запрос: ADC_CDR[31:0] = ADC1_DR[15:0] | ADC3_DR[15:0]
3й запрос: ADC_CDR[31:0] = ADC3_DR[15:0] | ADC2_DR[15:0]
4й запрос: ADC_CDR[31:0] = ADC2_DR[15:0] | ADC1_DR[15:0]

– **Режим DMA 3**: он подобен режиму DMA 2, но в младшем полуслове передаются данные двух DMA. Порядок передачи подобен режиму DMA 2.

Режим DMA 3 используется в чередующемся режиме с 6- и 8-бит разрешением (парный и тройной режимы).

Пример:

a) Чередующийся парный режим: запрос DMA выдаётся на каждые 2 данных
1й запрос: ADC_CDR[15:0] = ADC2_DR[7:0] | ADC1_DR[7:0]
2й запрос: ADC_CDR[15:0] = ADC2_DR[7:0] | ADC1_DR[7:0]

b) Чередующийся тройной режим: запрос DMA выдаётся на каждые 2 данных
1й запрос: ADC_CDR[15:0] = ADC2_DR[7:0] | ADC1_DR[7:0]
2й запрос: ADC_CDR[15:0] = ADC1_DR[7:0] | ADC3_DR[7:0]
3й запрос: ADC_CDR[15:0] = ADC3_DR[7:0] | ADC2_DR[7:0]
4й запрос: ADC_CDR[15:0] = ADC2_DR[7:0] | ADC1_DR[7:0]

Переполнение: При появлении переполнения в одном из работающих ADC запросы DMA больше не выдаются и переданные в RAM данные достоверны. Бит EOC в правильно сработавших ADC может остаться стоять.

13.9.1. Вставной одновременный режим

Внешний сигнал запуска подаётся через мультиплексор вставной группы ADC1 (биты JEXTSEL[3:0] в регистр ADC1_CR2). Одновременно запуск поступает и на ADC2 и ADC3.

NB: Не преобразуйте один канал на двух/трёх АЦП (тогда времена выборки не совпадают).

В одновременном режиме АЦП должны обрабатывать последовательности равной длины, или интервал между запусками должен быть больше самой длинной из 2 (парный режим) или 3 (тройной режим) последовательностей. Иначе ADC с самой короткой последовательностью стартует до окончания преобразования соседней.

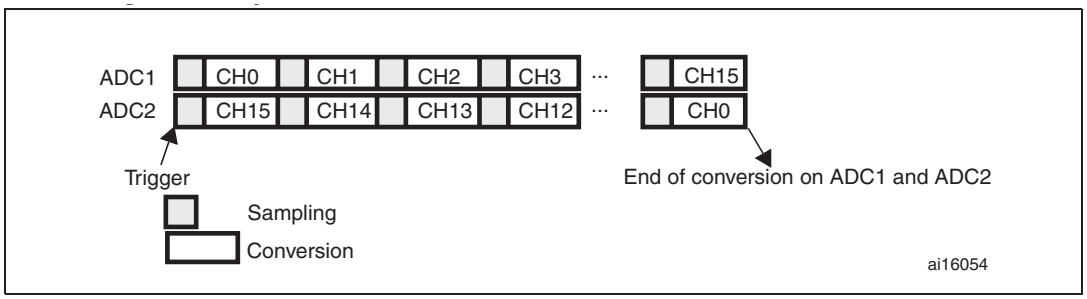
Регулярные преобразования могут делаться на одном или всех ADC. В этом случае они независимы друг от друга и прерываются при появлении вставной последовательности. По завершению вставной группы их работа возобновляется.

Парный режим ADC

По концу преобразования на ADC1 или ADC2:

- Результат пишется в свои регистры ADC_JDRx.
- Прерывание JEOC выставляется (если разрешено на одном из двух ADC) когда ADC1/ADC2 закончат преобразование всех вставных каналов.

Рис. 52. Вставной одновременный режим на 4 каналах: Парный режим ADC

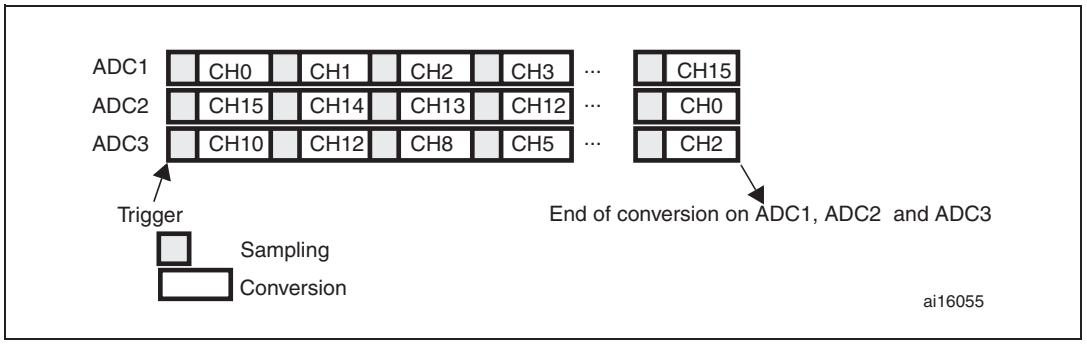


Тройной режим ADC

По концу преобразования на ADC1, ADC2 или ADC3:

- Результат пишется в свои регистры ADC_JDRx.
- Прерывание JEOC выставляется (если разрешено на одном из трёх ADC) когда ADC1/ADC2/ ADC3 закончат преобразование всех вставных каналов.

Рис. 53. Вставной одновременный режим на 4 каналах: Тройной режим ADC



13.9.2. Регулярный одновременный режим

Внешний сигнал запуска подаётся через мультиплексор регулярной группы ADC1 (биты EXTSEL[3:0] в регистр ADC1_CR2). Одновременно запуск поступает и на ADC2 и ADC3.

NB: Не преобразуйте один канал на двух/трёх АЦП (тогда времена выборки не совпадают).

По концу преобразования на ADC1 или ADC2:

- Генерируется запрос 32-бит DMA передачи (если стоит бит DMA) регистра ADC1_DR в SRAM с результатом ADC2 в старшем полуслове и ADC1 в младшем.
- При получении обоих результатов генерируется прерывание EOC (если разрешено в одном из АЦП).

В регулярном одновременном режиме АЦП должны обрабатывать последовательности равной длины, или интервал между запусками должен быть больше самой длинной из 2 (парный режим) или 3 (тройной режим) последовательностей. Иначе ADC с самой короткой последовательностью стартует до окончания преобразования соседней.

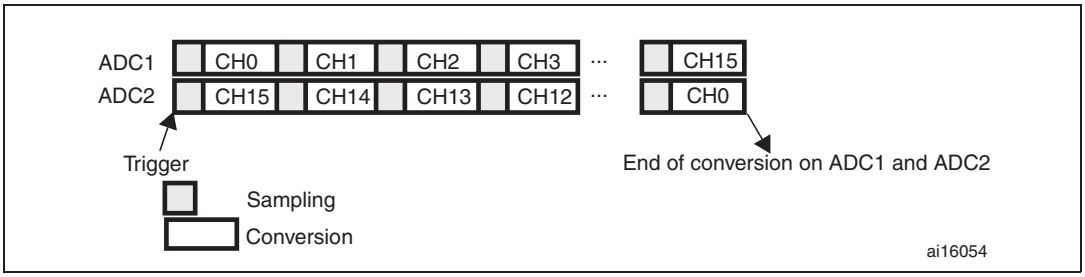
NB: Вставные преобразования должны быть выключены.

Парный режим ADC

По концу преобразования на ADC1 или ADC2:

- Выдаётся 32-бит запрос DMA (если биты DMA[1:0] в ADC_CCR равны 0b10). По нему результат ADC2 из старшего полуслова ADC_CDR пишется в SRAM и затем результат ADC1 из младшего полуслова ADC_CCR идёт в SRAM.
- Прерывание EOC выставляется (если разрешено на одном из двух ADC) когда ADC1/ADC2 закончат преобразование всех вставных каналов.

Figure 54. Одновременный регулярный режим на 16 каналах: Парный ADC mode

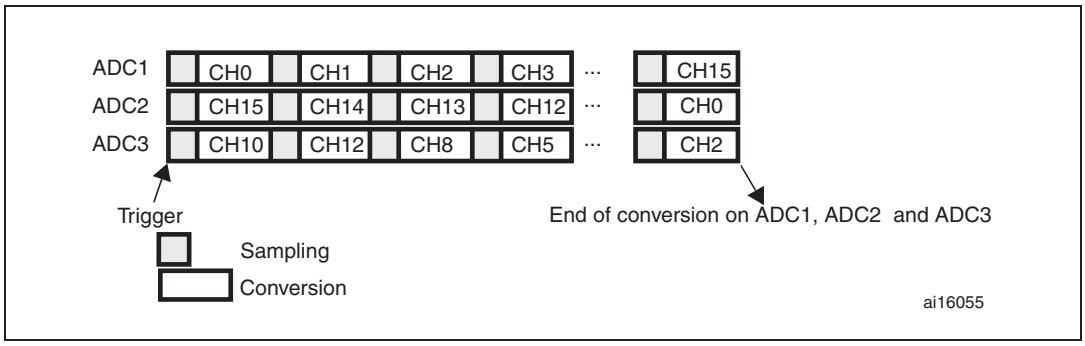


Тройной режим ADC

По концу преобразования на ADC1, ADC2 или ADC3:

- Выдаются три 32-бит запроса DMA (если биты DMA[1:0] в ADC_CCR равны 0b01). Порядок передачи из ADC_CDR в SRAM: ADC1, ADC2 и ADC3. Процесс повторяется для всех трёх новых преобразований.
- Прерывание EOC выставляется (если разрешено на одном из трёх ADC) когда ADC1/ADC2/ ADC3 закончат преобразование всех вставных каналов.

Рис. 55. Вставной одновременный режим на 4 каналах: Тройной режим ADC



13.9.3. Чередующийся режим

Этот режим предназначен только для регулярных групп (обычно один канал). Внешний сигнал запуска подаётся через мультиплексор регулярной группы ADC1.

Парный режим

После запуска:

- ADC2 стартует немедленно
- ADC1 стартует после задержки в несколько тактов ADC.

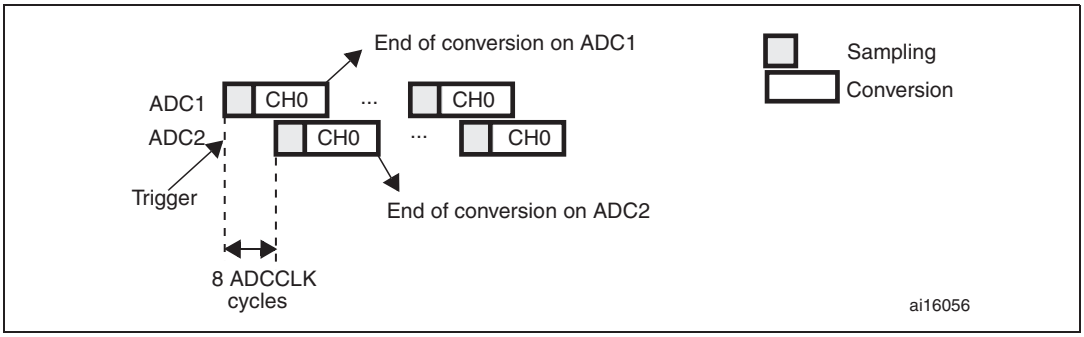
  Минимальную задержку между двумя преобразованиями в чередующемся режиме задают в битах DELAY регистра ADC_CCR. Но ADC не может начать преобразование если второй ADC ещё в стадии выборки (в одно и то же время выборку может делать только один ADC).

Если в ADC1 и ADC2 стоит бит CONT, то АЦП работают постоянно.

NB: Если последовательность преобразований прерывается (например, по концу передачи DMA), то секвенсор нескольких ADC надо сбросить в независимый режим (биты DUAL[4:0] = 00000) перед повторным переключением в чередующийся режим.

После прерывания EOC от ADC2 (разрешается битом EOCIE) выдаётся запрос 32-бит передачи DMA (если биты DMA[1:0] в ADC_CCR равны 0b10). Он сначала передаёт в SRAM результат ADC2 в старшем полуслове регистра ADC_CDR, Затем результат ADC1 в младшем полуслове.

Рис. 56. Постоянный чередующийся режим с 1 каналом: Парный режим ADC



Тройной режим ADC

- ADC1 стартует немедленно
- ADC2 стартует после задержки в несколько тактов ADC.
- ADC3 стартует после задержки в несколько тактов ADC относительно преобразования ADC2

Минимальную задержку между двумя преобразованиями в чередующемся режиме задают в битах DELAY регистра ADC_CCR. Но ADC не может начать преобразование если другой ADC ещё в стадии выборки (в одно и то же время выборку может делать только один ADC).

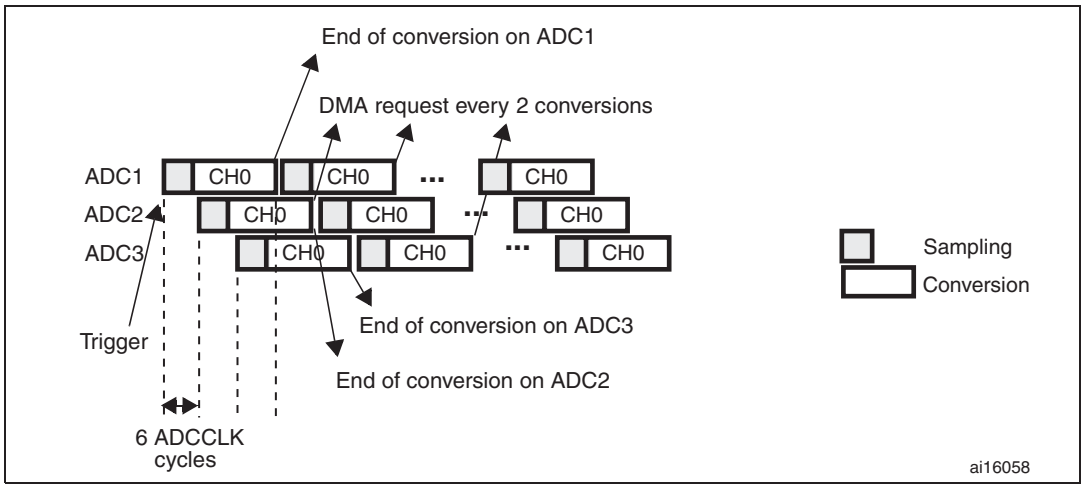
Если в ADC1, ADC2 и ADC3 стоит бит CONT, то АЦП работают постоянно.

NB: Если последовательность преобразований прерывается (например, по концу передачи DMA), то секвенсор нескольких ADC надо сбросить в независимый режим (биты DUAL[4:0] = 00000) перед повторным переключением в чередующийся режим.

В этом режиме запрос DMA выдаётся при готовности 2 результатов (если биты DMA[1:0] в ADC_CCR равны 0b10). После прерывания EOC от ADC2 (разрешается битом EOCIE) выдаётся запрос 32-бит передачи DMA (если биты DMA[1:0] в ADC_CCR равны 0b10). Он сначала передаёт в SRAM первый результат в младшем полуслове регистра ADC_CDR, затем второй результат в старшем полуслове. Последовательность такая:

- 1й запрос: ADC_CDR[31:0] = ADC2_DR[15:0] | ADC1_DR[15:0]
- 2й запрос: ADC_CDR[31:0] = ADC1_DR[15:0] | ADC3_DR[15:0]
- 3й запрос: ADC_CDR[31:0] = ADC3_DR[15:0] | ADC2_DR[15:0]
- 4й запрос: ADC_CDR[31:0] = ADC2_DR[15:0] | ADC1_DR[15:0], ...

Рис. 57. Постоянный чередующийся режим с 1 каналом: Тройной режим ADC



13.9.4. Режим попеременного запуска

Этот режим предназначен только для вставных групп. Внешний сигнал запуска подаётся через мультиплексор вставной группы ADC1.

NB: На одном или всех ADC может быть включено регулярное прерывание. В этом случае регулярные прерывания независимы друг от друга. Когда надо выполнить вставное преобразование, регулярное преобразование прерывается. По завершению вставного регулярное преобразование восстанавливается.

Если последовательность преобразований прерывается (например, по концу передачи DMA), то секвенсор нескольких ADC надо сбросить в независимый режим (биты DUAL[4:0] = 00000) перед повторным переключением в чередующийся режим.

Интервал времени между 2 запусками должен быть больше или равен 1 такту ADC.

Минимальный интервал времени между 2 запусками одного и того же ADC такой же как и в режиме одного ADC.

Парный режим

После запуска:

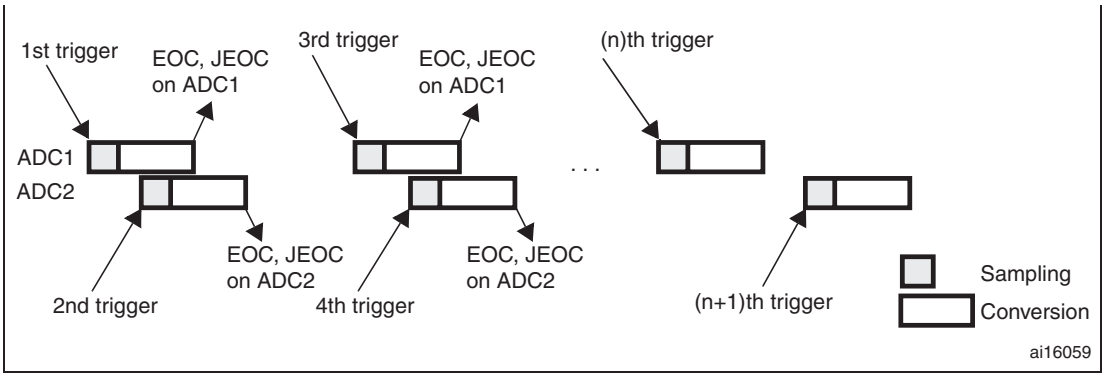
- После первого запуска ADC1 преобразует все каналы своей вставной группы.
- После второго запуска ADC2 преобразует все каналы своей вставной группы.
- И так далее.

Прерывание JEOC, если разрешено, генерируется после обработки всей вставной группы ADC1.

Прерывание JEOC, если разрешено, генерируется после обработки всей вставной группы ADC2.

Следующий сигнал запуска после завершения обеих групп возобновляет обработку вставных каналов с ADC1.

Рис. 58. Попеременный режим с вставными группами на каждом АЦП



В прерывном вставном режиме обоих АЦП:

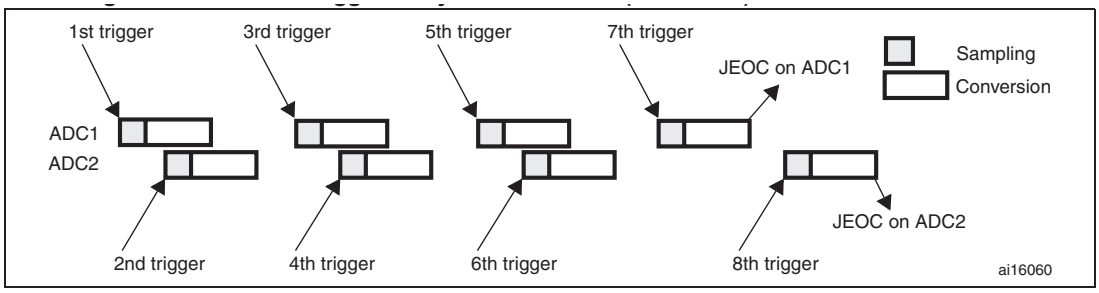
- После первого запуска ADC1 преобразует первый канал своей вставной группы.
- После второго запуска ADC2 преобразует первый канал своей вставной группы.
- И так далее.

Прерывание JEOC, если разрешено, генерируется после обработки всей вставной группы ADC1.

Прерывание JEOC, если разрешено, генерируется после обработки всей вставной группы ADC2.

Следующий сигнал запуска после завершения обеих групп возобновляет обработку вставных каналов с ADC1.

Рис. 59. Попеременный режим: 4 вставных канала (на каждом ADC) в прерывном режиме



Тройной режим ADC

- После первого запуска ADC1 преобразует все каналы своей вставной группы.
- После второго запуска ADC2 преобразует все каналы своей вставной группы.
- После третьего запуска ADC3 преобразует все каналы своей вставной группы.
- И так далее.

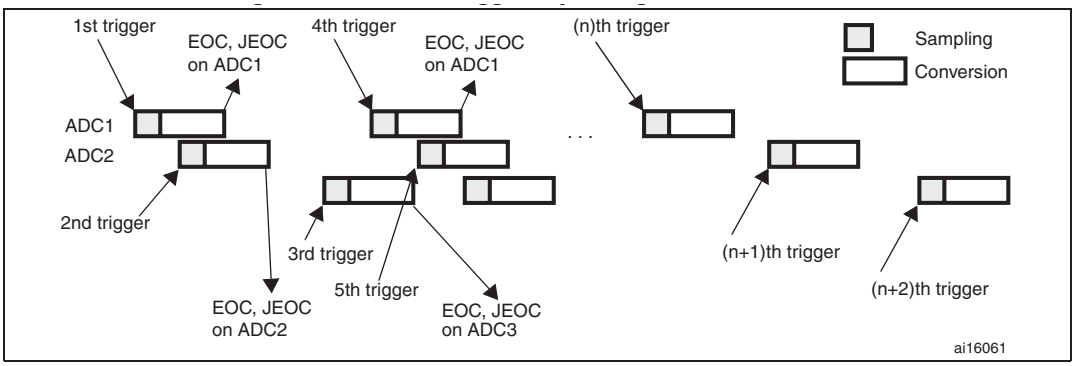
Прерывание JEOC, если разрешено, генерируется после обработки всей вставной группы ADC1.

Прерывание JEOC, если разрешено, генерируется после обработки всей вставной группы ADC2.

Прерывание JEOC, если разрешено, генерируется после обработки всей вставной группы ADC3.

Следующий сигнал запуска после завершения всех групп возобновляет обработку вставных каналов с ADC1.

Рис. 60. Попеременный режим: вставная группа на каждом ADC



13.9.5. Комбинированный регулярно/вставной одновременный режим

Есть возможность прерывать одновременное преобразование регулярной группы одновременным преобразованием вставной.

NB: В этом режиме обрабатывать надо последовательности одинаковой длины или время между запусками должно быть больше времени обработки самой длинной последовательности из 2 (Парный режим ADC) или 3 (Тройной режим ADC). Иначе, ADC с самой короткой последовательностью может перезапуститься пока другие ещё обрабатывают предыдущую.

13.9.6. Регулярный одновременный + Попеременный режим

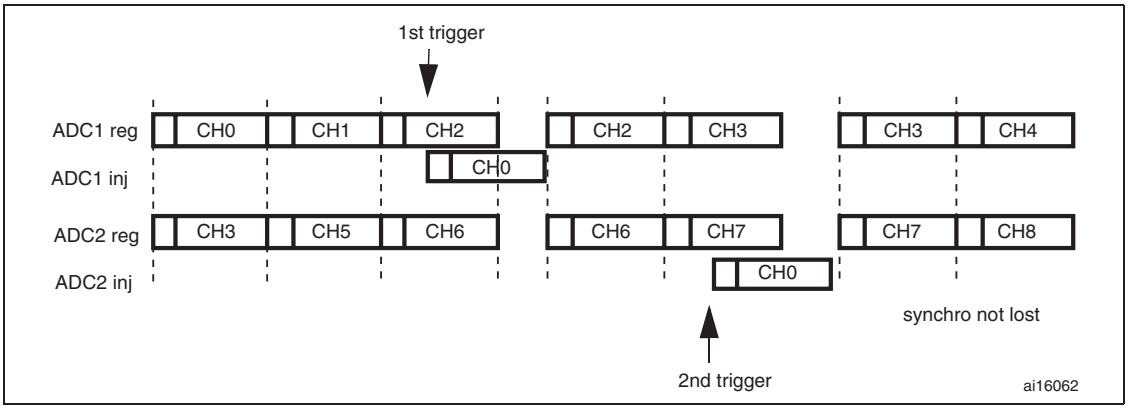
Есть возможность прерывать одновременное преобразование регулярной группы запуском попеременного преобразования вставной.

Попеременное вставное преобразование начинается сразу после сигнала запуска. Если регулярное преобразование уже работает, то оно останавливается на всех (ведущий/ведомый) ADC и синхронно восстанавливается после завершения вставной группы.

NB: В этом режиме обрабатывать надо последовательности одинаковой длины или время между запусками должно быть больше времени обработки самой длинной последовательности из 2 (Парный режим ADC) или 3 (Тройной режим ADC). Иначе, ADC с самой короткой последовательностью может перезапуститься пока другие ещё обрабатывают предыдущую.

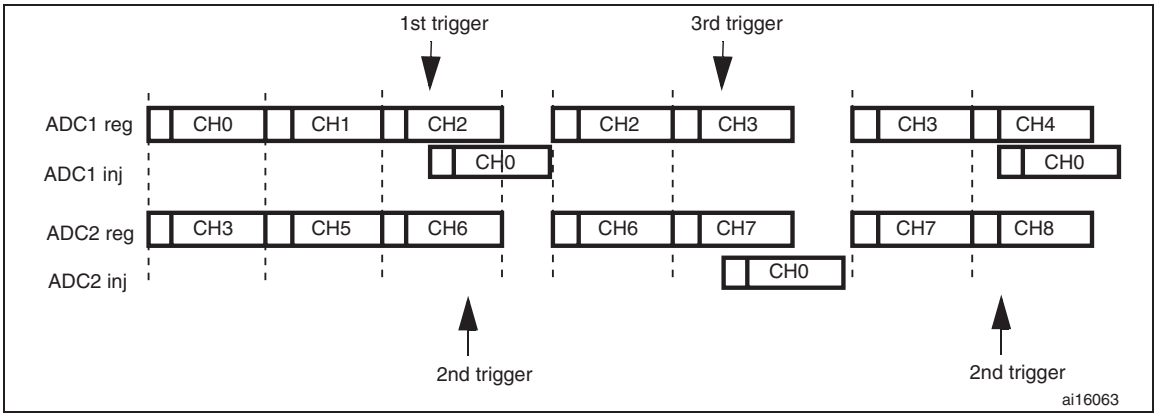
Если последовательность преобразований прерывается (например, по концу передачи DMA), то секвенсор нескольких ADC надо сбросить в независимый режим (биты DUAL[4:0] = 00000) перед повторным переключением в чередующийся режим.

Рис. 61. Попеременный + Регулярный одновременный



Сигнал запуска, пришедший во время обработки вставной последовательности, прервавшей регулярную, игнорируется. На следующей картинке игнорируется второй запуск.

Рис. 62. Запуск во время обработки вставной группы



13.10. Датчик температуры

Встроенный датчик температуры измеряет температуру окружающей среды (T_A) устройства.

- На STM32F40x и STM32F41x датчик внутренне подключён к входному каналу ADCx_IN16.
- На STM32F42x и STM32F43x датчик вместе с V_{BAT} внутренне подключены к входному каналу ADCx_IN18. Одновременно можно измерять только один источник. Если выбраны оба, то измеряется V_{BAT}.

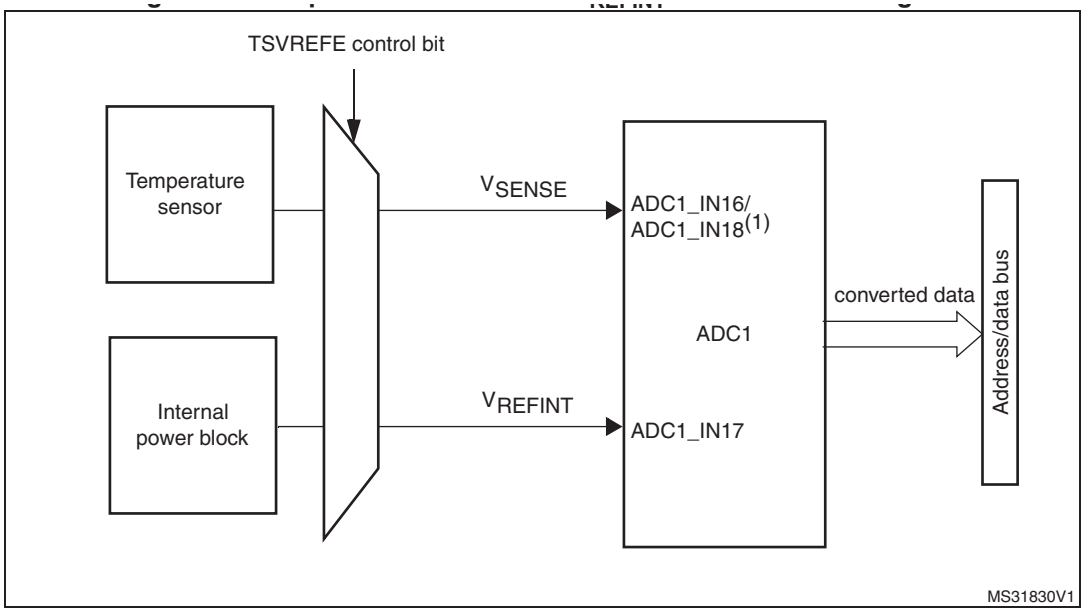
Не измеряемому датчику надо отключать питание.

NB: Бит TSVREFE надо ставить для разрешения обоих внутренних каналов: ADCx_IN16 или ADCx_IN18 (датчик температуры) и ADCx_IN17 (V_{REFINT}).

Основные характеристики

- Диапазон температуры: –40 to 125 °C
- Точность: ±1.5 °C

Рис. 63. Блок-схема датчика температуры и V_{REFINT}



1. V_{SENSE} это вход ADC1_IN16 для STM23F40x и STM32F41x и ADC1_IN18 для STM32F42x и STM32F43x.

Чтение температуры

Надо:

1. Выбрать канал ADCx_IN16 или ADCx_IN18.
2. Выбрать время выборки большее указанного в описании.
3. Установить бит TSVREFE в регистре ADC_CR2 для включения датчика.
4. Запустить АЦП битом SWSTART (или снаружи).
5. Прочитать данные V_{SENSE} из регистра АЦП
6. Получить температуру по формуле:

$$T\ (°C) = \{(V_{25} - VS_{ENSE}) / Avg\_Slope\} + 25.$$

где,

V_{25} = значение V_{SENSE} для 25° C и

Avg_Slope = Средний наклон кривой температуры (в mV/°C или µV/°C).

Действительные значения V_{25} and Avg_Slope см. в Электрических характеристиках.

NB: Для уменьшения задержки включения биты SWSTART и TSVREFE ставят одновременно.

Выход датчика температуры зависит линейно от температуры. Смещение зависит от конкретной микросхемы (до 45 °C).

Внутренний датчик удобен для определения изменений температуры, точно надо мерять внешним.

13.11. Контроль заряда батареи

Напряжение V_{BAT} может быть больше V_{DDA}. Поэтому ножка V_{BAT} подключена через делитель, который включается битом VBATE в ADC_CCR.

Если VBATE стоит, то подключается:

- Для STM32F40xx и STM32F41xx $V_{BAT}/2$ к каналу ADCx_IN16,
- Для STM32F42xx и STM32F43xx $V_{BAT}/4$ к каналу ADCx_IN18

NB: На STM32F42x и STM32F43x датчик вместе с V_{BAT} внутренне подключены к входному каналу ADCx_IN18. Одновременно можно измерять только один источник. Если выбраны оба, то измеряется V_{BAT}.

13.12.Прерывания АЦП

Разрешённые прерывания генерируются для конца преобразования групп, аналогового сторожа и переполнения.

В регистре ADC_SR есть не связанные с прерываниями флаги:

- JSTRT (Старт преобразования вставных групп)
- STRT (Старт преобразования регулярных групп)

Таблица 70. Прерывания АЦП.

Прерывание	Флаг	Бит разрешения
Конец преобразования регулярной группы	EOC	EOCIE
Конец преобразования вставной группы	JEOC	JEOCIE
Бит статуса Аналогового сторожа встал	AWD	AWDIE
Переполнение	OVR	OVRIE

13.13.Регистры АЦП

Доступны байтами, полусловами и словами.

13.13.1.Регистр состояния АЦП (ADC_SR)

Смещение адреса: 0x00

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										OVR	STRT	JSTRT	JEOC	EOC	AWD
										rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

— Биты 31:6 Резерв, не трогать.

— Бит 5 **OVR**: Переполнение

Ставится аппаратно при потере данных во всех режимах. Снимается программно. Определение переполнения включено только при DMA = 1 или EOCS = 1.

0: Не было

1: Было

— Бит 4 **STRT:** Флаг старта регулярных каналов.

Ставится аппаратно, снимается программно записью 0.

0: Не стартовали

1: Стартовали

— Бит 3 **JSTRT:** Флаг старта вставных каналов.

Ставится аппаратно, снимается программно записью 0.

0: Не стартовали

1: Стартовали

— Бит 2 **JEOC:** Конец преобразования вставных каналов.

Ставится аппаратно, снимается программно записью 0.

0: Нет конца

1: Есть конец

— Бит 1 **EOC:** Конец преобразования.

Ставится аппаратно, снимается программно записью 0.

0: Нет конца

1: Есть конец

— Бит 0 **AWD:** Флаг аналогового сторожа.

Ставится аппаратно, снимается программно записью 0.

0: Не случалось

1: Случилось

13.13.2.Регистр 1 управления АЦП (ADC_CR1)

Смещение адреса: 0x04

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					OVRIE	RES		AWDEN	JAWDEN	Reserved					
					rw	rw	rw	rw	rw						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DISCNUM[2:0]			JDISCEN	DISCEN	JAUTO	AWDSGL	SCAN	JEOCIE	AWDIE	EOCIE	AWDCH[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:27 Резерв, не трогать.

— Бит 26 **OVRIE:** Разрешение прерывания переполнения.
Ставится и снимается программно.
0: Нельзя
1: Можно

— Биты 25:24 **RES[1:0]**: Разрешающая способность
Пишется программно.
00: 12-бит (15 тактов ADCCLK)
01: 10-бит (13 тактов ADCCLK)
10: 8-бит (11 тактов ADCCLK)
11: 6-бит (9 тактов ADCCLK)

— Бит 23 **AWDEN:** Разрешение сторожа для регулярных каналов.
Ставится и снимается программно.
0: Нельзя
1: Можно

— Бит 22 **JAWDEN:** Разрешение сторожа для вставных каналов.
Ставится и снимается программно.
0: Нельзя
1: Можно

— Биты 21:16 Резерв, не трогать.

— Биты 15:13 **DISCNUM[2:0]**: Счётчик каналов Прерывного режима
Ставится и снимается программно.
000: 1 канал
001: 2 канала
…….
111: 8 каналов

— Бит 12 **JDISCEN**: Прерывный режим вставных каналов
Ставится и снимается программно.
0: Нельзя
1: Можно

— Бит 11 **DISCEN**: Прерывный режим регулярных каналов.
Ставится и снимается программно.
0: Нельзя
1: Можно

— Бит 10 **JAUTO:** Автоматическая обработка вставной группы.
Ставится и снимается программно.
0: Нельзя
1: Можно

— Бит 9 **AWDSGL:** Разрешение сторожа для всех или одного канала из битов AWDCH[4:0].
Ставится и снимается программно.
0: Все каналы
1: Один канал

— Бит 8 **SCAN:** Режим Сканирования каналов из регистров ADC_SQRx или ADC_JSQRx.
Ставится и снимается программно.

0: Нельзя
1: Можно

NB: Разрешённое битом EOCIE прерывание EOC возникает при:

- При снятом бите EOCS по концу каждой регулярной последовательности
- При стоящем бите EOCS по концу каждого регулярного канала

Разрешённое битом JEOCIE прерывание JEOC возникает по концу последнего канала.

— Бит 7 **JEOCIE**: Разрешение прерывания вставных каналов по флагу JEOC.
Ставится и снимается программно.
0: Нельзя
1: Можно

— Бит 6 **AWDIE**: Разрешение прерывания Аналогового сторожа.
Ставится и снимается программно.
0: Нельзя
1: Можно

— Бит 5 **EOCIE:** Разрешение прерывания для EOC.
Ставится и снимается программно.
0: Нельзя
1: Можно

— Биты 4:0 **AWDCH[4:0]:** Выбор каналов АЦП для Аналогового сторожа.
Ставится и снимается программно.
00000: Канал 0
00001: Канал 1
….
01111: Канал 15
10000: Канал 16
10001: Канал 17
10010: Канал 18
Остальные значения в резерве.

13.13.3.Регистр 2 управления АЦП (ADC_CR2)

Смещение адреса: 0x08

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved	SWSTART	EXTEN		EXTSEL[3:0]				reserved	JSWSTART	JEXTEN		JEXTSEL[3:0]			
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved				ALIGN	EOCS	DDS	DMA	Reserved						CONT	ADON
				rw	rw	rw	rw							rw	rw

— Бит 31 Резерв, не трогать.

— Бит 30 **SWSTART:** Старт преобразования регулярных каналов
Ставится программно, снимается аппаратно сразу после старта.
0: Состояние сброса
1: Поехали.

NB: Можно ставить только при ADON = 1, иначе не пускается.

— Биты 29:28 **EXTEN**: Включение внешнего запуска регулярных каналов
Пишутся программно.
00: Запрет
01: По переднему фронту
10: По заднему фронту
11: По обоим фронтам

— Биты 27:24 **EXTSEL[3:0]**: Выбор внешнего события для регулярной группы
0000: Timer 1 CC1 event
0001: Timer 1 CC2 event
0010: Timer 1 CC3 event
0011: Timer 2 CC2 event

0100: Timer 2 CC3 event
0101: Timer 2 CC4 event
0110: Timer 2 TRGO event
0111: Timer 3 CC1 event
1000: Timer 3 TRGO event
1001: Timer 4 CC4 event
1010: Timer 5 CC1 event
1011: Timer 5 CC2 event
1100: Timer 5 CC3 event
1101: Timer 8 CC1 event
1110: Timer 8 TRGO event
1111: EXTI line11

— Бит 23 Резерв, не трогать.

— Бит 22 **JSWSTART:** Старт обработки группы вставных каналов

Ставится программно, снимается аппаратно сразу после запуска.

0: Сброшено
1: Запуск

NB: Можно ставить только при ADON = 1, иначе не пускается.

— Бит 21:20 **JEXTEN:** Включение внешнего запуска вставных каналов

Пишутся программно.

00: Запрет
01: По переднему фронту
10: По заднему фронту
11: По обоим фронтам

— Биты 19:16 **JEXTSEL[3:0]** Выбор внешнего события для вставной группы

0000: Timer 1 CC4 event
0001: Timer 1 TRGO event
0010: Timer 2 CC1 event
0011: Timer 2 TRGO event
0100: Timer 3 CC2 event
0101: Timer 3 CC4 event
0110: Timer 4 CC1 event
0111: Timer 4 CC2 event
1000: Timer 4 CC3 event
1001: Timer 4 TRGO event
1010: Timer 5 CC4 event
1011: Timer 5 TRGO event
1100: Timer 8 CC2 event
1101: Timer 8 CC3 event
1110: Timer 8 CC4 event
1111: EXTI line15

— Биты 15:12 Резерв, не трогать.

— Бит 11 **ALIGN**: Выравнивание результата.

Ставится и снимается программно.

0: Вправо
1: Влево

— Бит 10 **EOCS**: Выбор конца преобразования

Ставится и снимается программно.

0: Бит EOC встаёт по концу каждой последовательности регулярных преобразований. Переполнение ищут только при DMA=1.
1: Бит EOC встаёт по концу каждого регулярного преобразования. Переполнение ищут всегда.

— Бит 9 **DDS**: Выбор выключения DMA (одинарный режим ADC)

Ставится и снимается программно.

0: Новый запрос DMA за последней передачей не выдаётся (как стоит в контроллере DMA)
1: Запросы DMA пока есть преобразования и DMA=1

— Бит 8 **DMA:** Режим DMA (одинарный режим ADC).
Ставится и снимается программно.
0: Нельзя
1: Можно

— Биты 7:2 Резерв, не трогать.

— Бит 1 **CONT:** Разрешение непрерывного преобразования.
Ставится и снимается программно. Снятие бита останавливает непрерывное преобразование.
0: Однократное преобразование
1: Непрерывное преобразование

— Бит 0 **ADON:** Вкл./Выкл. ADC.
Ставится и снимается программно.
0: Стоп преобразования и выключает питание ADC.
1: Включает ADC.

13.13.4.Регистр 1 времени выборки АЦП (ADC_SMPR1)

Смещение адреса: 0x0C

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					SMP18[2:0]			SMP17[2:0]			SMP16[2:0]			SMP15[2:1]	
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP15_0	SMP14[2:0]			SMP13[2:0]			SMP12[2:0]			SMP11[2:0]			SMP10[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:27 Резерв, не трогать.

— Биты 26:0 **SMPx[2:0]**: Время выборки канала **x**.
Ставится и снимается программно. Во время цикла выборки эти биты изменять нельзя.
000: 3 такта
001: 15 тактов
010: 28 тактов
011: 56 тактов
100: 84 такта
101: 112 тактов
110: 144 тактов
111: 480 тактов

13.13.5.Регистр 2 времени выборки АЦП (ADC_SMPR2)

Смещение адреса: 0x10

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		SMP9[2:0]			SMP8[2:0]			SMP7[2:0]			SMP6[2:0]			SMP5[2:1]	
Res.		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP 5_0	SMP4[2:0]			SMP3[2:0]			SMP2[2:0]			SMP1[2:0]			SMP0[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:24 Резерв, не трогать.

— Биты 19:0 **SMPx[2:0]**: Время выборки канала **x**.
Ставится и снимается программно. Во время цикла выборки эти биты изменять нельзя.
000: 3 такта
001: 15 тактов
010: 28 тактов
011: 56 тактов
100: 84 такта
101: 112 тактов
110: 144 тактов
111: 480 тактов

13.13.6.Смещение данных вставных каналов АЦП (ADC_JOFRx) (x=1..4)

Смещение адреса: 0x14 - 0x20

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				JOFFSETx[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:12 Резерв, не трогать.

— Биты 11:0 **JOFFSETx[11:0]**: Смещение данных вставного канала x

Пишется программно. Вычитается из результата преобразования вставных каналов, итог пишется в регистр ADC_JDRx.

13.13.7.Верхний порог аналогового сторожа (ADC_HTR)

Смещение адреса: 0x24

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				HT[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:12 Резерв, не трогать.

— Биты 11:0 **HT[11:0]:** Верхний порог аналогового сторожа

Пишется программно.

NB: Писать можно во время цикла преобразования АЦП. Значение начнёт действовать после завершения следующего преобразования. Запись делается с задержкой, что может создать неопределённость в начало работы нового значения.

13.13.8.Нижний порог аналогового сторожа (ADC_LTR)

Смещение адреса: 0x28

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				LT[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:12 Резерв, не трогать.

— Биты 11:0 **LT[11:0]:** Нижний порог аналогового сторожа

Пишется программно.

NB: Писать можно во время цикла преобразования АЦП. Значение начнёт действовать после завершения следующего преобразования. Запись делается с задержкой, что может создать неопределённость в начало работы нового значения.

13.13.9.Регистр 1 регулярной последовательности (ADC_SQR1)

Смещение адреса: 0x2C

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								L[3:0]				SQ16[4:1]			
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ16_0	SQ15[4:0]					SQ14[4:0]					SQ13[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:24 Резерв, не трогать.

— Биты 23:20 **L[3:0]**. Длина регулярной последовательности.

Пишется программно.

— Биты 19:15, 14:10, 9:5, 4:0
SQx[4:0]: Номер аналогового канала (0-18) на месте **x** (x=16-13) последовательности.
Пишется программно.

13.13.10.Регистр 2 регулярной последовательности (ADC_SQR2)

Смещение адреса: 0x30

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		SQ12[4:0]					SQ11[4:0]					SQ10[4:1]			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ10_0	SQ9[4:0]					SQ8[4:0]					SQ7[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:30 Резерв, не трогать.
— Биты 29:25, 24:20, 19:15,14:10, 9:5, 4:0
SQx[4:0]: Номер аналогового канала (0-18) на месте **x** (x=12-7) последовательности.
Пишется программно.

13.13.11.Регистр 3 регулярной последовательности (ADC_SQR3)

Смещение адреса: 0x34

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		SQ6[4:0]					SQ5[4:0]					SQ4[4:1]			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ4_0	SQ3[4:0]					SQ2[4:0]					SQ1[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:30 Резерв, не трогать.
— Биты 29:25
— Биты 24:20
— Биты 19:15
— Биты 14:10
— Биты 9:5
— Биты 4:0 **SQx[4:0]:** Номер аналогового канала (0-18) на месте **x** (x=6-1) последовательности.
Пишется программно.

13.13.12.Регистр вставной последовательности (ADC_JSQR)

Смещение адреса: 0x38

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										JL[1:0]		JSQ4[4:1]			
										rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JSQ4_0	JSQ3[4:0]					JSQ2[4:0]					JSQ1[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:22 Резерв, не трогать.
— Биты 21:20 **JL[2:0]**. Длина вставной последовательности.
Пишется программно.
00: 1 преобразование
01: 2 преобразования
10: 3 преобразования
11: 4 преобразования
— Биты 19:15 **JSQ4[4:0]**: Номер аналогового канала (0..17) 4-го преобразования (при JL[1:0]=3)(1)
Пишется программно.

— Биты 14:10 **JSQ3[4:0]**: 3е преобразование (при JL[1:0]=3)
— Биты 9:5 **JSQ2[4:0]**: 2е преобразование (при JL[1:0]=3)
— Биты 4:0 **JSQ1[4:0]**: 1е преобразование (при JL[1:0]=3).

NB: При JL=3 (4 канала) порядок обработки такой: JSQ1[4:0] >> JSQ2[4:0] >> JSQ3[4:0] >> JSQ4[4:0]
При JL=2 (3 канала) порядок обработки такой: JSQ2[4:0] >> JSQ3[4:0] >> JSQ4[4:0]
При JL=1 (2 канала) порядок обработки такой: JSQ3[4:0] >> JSQ4[4:0]
При JL=0 (1 канал) преобразуется только канал JSQ4[4:0]

13.13.13.Регистры x данных вставных каналов (ADC_JDRx) (x=1..4)

Смещение адреса: 0x3C - 0x48

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JDATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

— Биты 31:16 Резерв, не трогать.
— Биты 15:0 **JDATA[15:0]**. Выравненные влево или вправо данные вставного канала.
Только чтение.

13.13.14.Регистр данных регулярных каналов (ADC_DR)

Смещение адреса: 0x4C

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

— Биты 31:16 Резерв, не трогать.
— Биты 15:0 **DATA[15:0]**: Выравненные влево или вправо регулярные данные
Только чтение.

13.13.15.Общий регистр состояния (ADC_CSR)

Смещение адреса: 0x00 (относительно базового адреса ADC1 + 0x300)

По сбросу: 0x0000 0000

Это просто читаемая копия соответствующих битов регистров состояния разных ADC, и всё.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										OVR3	STRT3	JSTRT3	JEOC 3	EOC3	AWD3
										ADC3					
										r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		OVR2	STRT2	JSTRT 2	JEOC2	EOC2	AWD2	Reserved		OVR1	STRT1	JSTRT1	JEOC 1	EOC1	AWD1
		ADC2								ADC1					
		r	r	r	r	r	r			r	r	r	r	r	r

— Биты 31:22 Резерв, не трогать.
— Бит 21 **OVR3**: Флаг переполнения ADC3.
— Бит 20 **STRT3**: Флаг старта регулярного канала ADC3.
— Бит 19 **JSTRT3**: Флаг старта вставного канала ADC3.
— Бит 18 **JEOC3**: Конец преобразования вставного канала ADC3.
— Бит 17 **EOC3**: Конец преобразования регулярного канала ADC3.
— Бит 16 **AWD3**: Флаг аналогового сторожа ADC3.
— Биты 15:14 Резерв, не трогать.
— Бит 13 **OVR2**: Флаг переполнения ADC2.
— Бит 12 **STRT2**: Флаг старта регулярного канала ADC2.

— Бит 11 **JSTRT2**: Флаг старта вставного канала ADC2.
— Бит 10 **JEOC2**: Конец преобразования вставного канала ADC2.
— Бит 9 **EOC2**: Конец преобразования регулярного канала ADC2.
— Бит 8 **AWD2**: Флаг аналогового сторожа ADC2.
— Биты 7:6 Резерв, не трогать.
— Бит 5 **OVR1**: Флаг переполнения ADC1.
— Бит 4 **STRT1**: Флаг старта регулярного канала ADC1.
— Бит 3 **JSTRT1**: Флаг старта вставного канала ADC1.
— Бит 2 **JEOC1**: Конец преобразования вставного канала ADC1.
— Бит 1 **EOC1**: Конец преобразования регулярного канала ADC1.
— Бит 0 **AWD1**: Флаг аналогового сторожа ADC1.

13.13.16.Общий регистр управления (ADC_CCR)

Смещение адреса: 0x04 (относительно базового адреса ADC1 + 0x300)

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								TSVREFE	VBATE	Reserved				ADCPRE	
								rw	rw					rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMA[1:0]		DDS	Res.	DELAY[3:0]				Reserved			MULTI[4:0]				
rw	rw	rw		rw	rw	rw	rw				rw	rw	rw	rw	rw

— Биты 31:24 Резерв, не трогать.
— Бит 23 **TSVREFE**: Включение датчика температуры и V_{REFINT}
Ставится и снимается программно.
0: Выкл.
1: Вкл.
Note: На STM32F42x и STM32F43x бит VBATE нужно снимать при стоящем TSVREFE . Если стоят оба, то измеряется только V_{BAT}.
— Бит 22 **VBATE**: Включение V_{BAT}.
Ставится и снимается программно.
0: Выкл.
1: Вкл.
— Биты 21:18 Резерв, не трогать.
— Биты 17:16 **ADCPRE**: Общий предделитель для всех ADC.
Ставится и снимается программно.
00: PCLK2 / 2
01: PCLK2 / 4
10: PCLK2 / 6
11: PCLK2 / 8
— Биты 15:14 **DMA[1:0]**: Режим DMA для нескольких ADC
Ставится и снимается программно.
00: DMA выключен
01: DMA режим 1 (2 / 3 полуслова чередой - 1, 2 затем 3)
10: DMA режим 2 (2 / 3 полуслова парами - 2&1, 1&3 затем 3&2)
11: DMA режим 3 (2 / 3 байта парами - 2&1, 1&3 затем 3&2)
— Бит 13 **DDS**: Режим выключения DMA для нескольких ADC
Ставится и снимается программно.
0: По концу последней передачи новый запрос DMA не выдаётся (как в контроллере DMA). Бит DMA аппаратно не снимается, перед новыми передачами надо чистить и ставить программно.
1: Запросы DMA выдаются пока идут преобразования и are issued as long as data are converted and DMA[1;0] = 01, 10 или 11.
— Бит 12 Резерв, не трогать.
— Биты 11:8 **DEJAY[3:0]**: Задержка между двумя фазами выборки
Ставится и снимается программно. Для парного и тройного чередующегося режима.
0000: 5 * T_{ADCCLK}

…

1111: 20 * T_{ADCCLK}

— Биты 7:5 Резерв, не трогать.

— Биты 4:0 **MULTI[4:0]**: Выбор режима нескольких ADC

Ставится и снимается программно.

– Все ADC независимы:

00000: Независимый режим

– 00001 до 01001: Парный режим, ADC1 и ADC2 работают вместе, ADC3 независимо

00001: Одновременный регулярный + Одновременный вставной

00010: Одновременный регулярный + Попеременный запуск

00011: Резерв

00101: Одновременный вставной

00110: Одновременный регулярный

00111: Чередующийся

01001: Попеременный запуск

– 10001 до 11001: Тройной режим: ADC1, 2 и 3 работают вместе

10001: Одновременный регулярный + Одновременный вставной

10010: Одновременный регулярный + Попеременный запуск

10011: Резерв

10101: Одновременный вставной

10110: Одновременный регулярный

10111: Чередующийся

11001: Попеременный запуск

Всё остальное зарезервировано

NB: В режиме нескольких ADC изменение конфигурации каналов приводит к аборту и может привести к потере синхронизации. Надо заранее его выключить.

13.13.17.Общий регистр регулярных данных (ADC_CDR)

Смещение адреса: 0x08 (относительно базового адреса ADC1 + 0x300)

По сбросу: 0x0000 0000

Для парного и тройного регулярных режимов.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA2[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA1[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

— Биты 31:16 **DATA2[15:0]**: Второе данное пары регулярных преобразований

– В парном режиме это регулярное данное ADC2.

– В тройном режиме это чередующееся данное ADC2, ADC1 и ADC3.

— Биты 15:0 **DATA1[15:0]**: Первое данное пары регулярных преобразований

– В парном режиме это регулярное данное ADC1.

– В тройном режиме это чередующееся данное ADC1, ADC3 и ADC2.

13.13.18.Карта регистров АЦП

Таблица 71. Глобальная карта регистров ADC

Offset	Register
0x000 - 0x04C	ADC1
0x050 - 0x0FC	Резерв
0x100 - 0x14C	ADC2
0x118 - 0x1FC	Резерв
0x200 - 0x24C	ADC3
0x250 - 0x2FC	Резерв
0x300 - 0x308	Общие регистры

Таблица 72. Отдельные регистры

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**ADC_SR**	Reserved																										OVR	STRT	JSTRT	JEOC	EOC	AWD
	Reset value																											0	0	0	0	0	0
0x04	**ADC_CR1**	Reserved					OVRIE	RES[1:0]		AWDEN	JAWDEN	Reserved						DISC NUM [2:0]			JDISCEN	DISCEN	JAUTO	AWD SGL	SCAN	JEOCIE	AWDIE	EOCIE	AWDCH[4:0]					
	Reset value						0	0	0	0	0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**ADC_CR2**	Reserved	SWSTART	EXTEN[1:0]		EXTSEL [3:0]				Reserved	JSWSTART	JEXTEN[1:0]		JEXTSEL [3:0]				Reserved				ALIGN	EOCS	DDS	DMA	Reserved						CONT	ADON
	Reset value		0	0	0	0	0	0	0		0	0	0	0	0	0	0					0	0		0							0	0
0x0C	**ADC_SMPR1**	Sample time bits SMPx_x																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**ADC_SMPR2**	Sample time bits SMPx_x																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**ADC_JOFR1**	Reserved																				JOFFSET1[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x18	**ADC_JOFR2**	Reserved																				JOFFSET2[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**ADC_JOFR3**	Reserved																				JOFFSET3[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x20	**ADC_JOFR4**	Reserved																				JOFFSET4[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x24	**ADC_HTR**	Reserved																				HT[11:0]											
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1
0x28	**ADC_LTR**	Reserved																				LT[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**ADC_SQR1**	Reserved								L[3:0]								Regular channel sequence SQx_x bits															
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	**ADC_SQR2**	Reserved	Regular channel sequence SQx_x bits																														
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x34	**ADC_SQR3**	Reserved	Regular channel sequence SQx_x bits																														
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	**ADC_JSQR**	Reserved										JL[1:0]		Injected channel sequence JSQx_x bits																			
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	**ADC_JDR1**	Reserved																JDATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	**ADC_JDR2**	Reserved																JDATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	**ADC_JDR3**	Reserved																JDATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	**ADC_JDR4**	Reserved																JDATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	**ADC_DR**	Reserved																Regular DATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Таблица 73. Общие регистры

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**ADC_CSR**	Reserved										OVR	STRT	JSTRT	JEOC	EOC	AWD	Reserved		OVR	STRT	JSTRT	JEOC	EOC	AWD	Reserved		OVR	STRT	JSTRT	JEOC	EOC	AWD
	Reset value											0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
												ADC3								ADC2								ADC1					
0x04	**ADC_CCR**	Reserved								TSVREFE	VBATE	Reserved				ADCPRE[1:0]		DMA[1:0]		DDS	Reserved	DELAY [3:0]				Reserved		MULTI [4:0]					
	Reset value									0	0					0	0	0	0	0		0	0	0	0			0	0	0	0	0	0
0x08	**ADC_CDR**	Regular DATA2[15:0]																Regular DATA1[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

14. Цифро-аналоговый преобразователь (DAC)

14.1. Введение в ЦАП

ЦАП можно включать в 8- или 12-бит режим и использовать совместно с ПДП. В 12-бит режиме данные могут быть выравненными влево или вправо. ЦАП имеет два канала со своими конверторами. В парном режиме преобразование может выполняться независимо или одновременно, когда оба канала группируются вместе для синхронного обновления операции. Ножка опорного напряжения V_{REF+} общая для АЦП и ЦАП.

14.2. Основные свойства ЦАП

- Два конвертора ЦАП со своими выходными каналами
- Правое или левое выравнивание данных в 12-бит режиме
- Возможность синхронного обновления
- Генерация шумоподобного сигнала
- Генерация треугольного сигнала
- Независимое или одновременное парное преобразование
- Возможность ПДП для обоих каналов
- Определение исчерпания DMA
- Внешний запуск преобразования
- Вход опорного напряжения V_{REF+}

Рис 64. Блок-схема ЦАП.

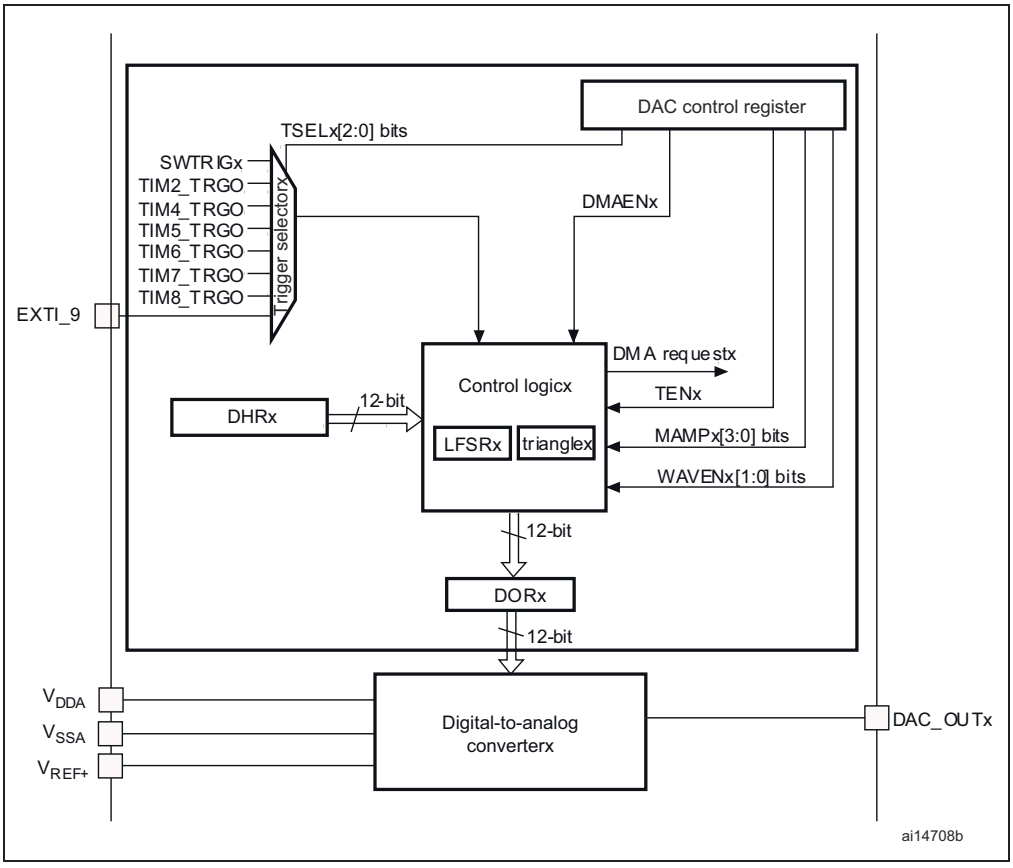


Таблица 74. Ножки ЦАП

Имя	Тип	Заметки
V_{REF+}	Опорное напряжение	1.8 V ≤V_{REF+} ≤V_{DDA}
V_{DDA}	Аналоговое питание	Аналоговое питание
V_{SSA}	Аналоговая земля	Аналоговая земля
DAC_OUTx	Аналоговый выход	Аналоговый выход

NB. После включения канала ЦАП к его выходу (DAC_OUTx) автоматически подключается соответствующая ножка GPIO (PA4 или PA5). Во избежание паразитных утечек ножки PA4 или PA5 нужно включить как аналоговый вход (AIN).

14.3. Основные свойства ЦАП

14.3.1. Включение канала ЦАП

Каналы ЦАП включаются раздельно битами ENx в регистре DAC_CR. Разрешается канал по истечении времени запуска t_{WAKEUP}.

NB: Бит ENx включает только аналоговую часть канала (DAC channelx), цифровой интерфейс включён даже при снятом бите ENx.

14.3.2. Включение выходных каскадов ЦАП

Для уменьшения выходного сопротивления и ради прямого управления внешней нагрузкой имеются два выходных каскада, которые включаются битами BOFFx в регистре DAC_CR.

14.3.3. Формат данных ЦАП

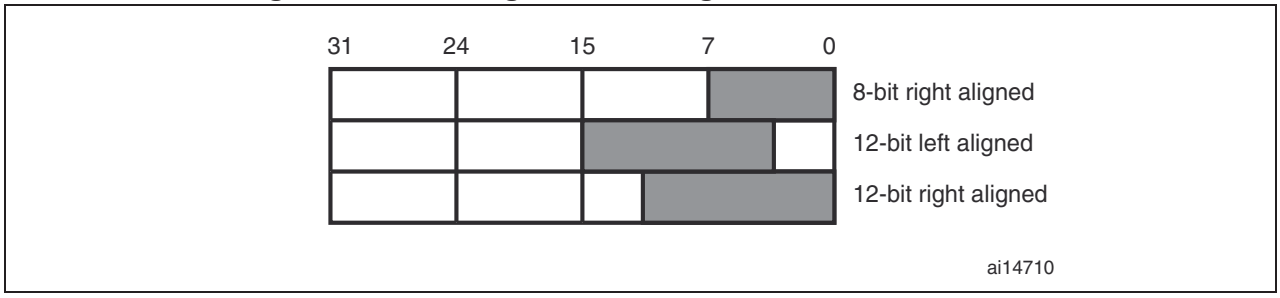
В зависимости от режима данные надо писать в определённые регистры:

• В одиночном режиме канала DAC channelx:

- 8-бит правое выравнивание: данные пишутся в биты DAC_DHR8Rx[7:0], хранятся в DHRx[11:4]
- 12-бит левое выравнивание: данные пишутся в биты DAC_DHR12Lx[15:4], хранятся в DHRx[11:0]
- 12-бит правое выравнивание: данные пишутся в биты DAC_DHR12Rx[11:0], хранятся в DHRx[11:0]

Данные из регистров DAC_DHRyyyx сдвигаются и пишутся в регистры DHRx. Регистры DHRx пишутся в регистры DORx программно или по внешнему сигналу.

Рис. 65. Данные в одинарном режиме

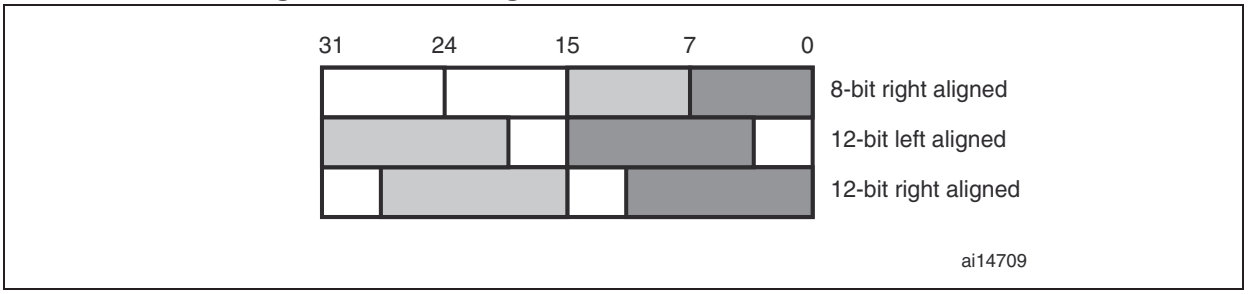


• В парном режиме каналов DAC:

- 8-бит правое выравнивание: DAC channel1 пишется в биты DAC_DHR8RD[7:0] (хранится в DHR1[11:4]), DAC channel2 пишется в биты DAC_DHR8RD[15:8] (хранится в DHR2[11:4])
- 12-бит левое выравнивание: DAC channel1 пишется в биты DAC_DHR12LD[15:4] (хранится в DHR1[11:0]), DAC channel2 пишется в биты DAC_DHR12LD[31:20] (хранится в DHR2[11:0])
- 12-бит правое выравнивание: DAC channel1 пишется в биты DAC_DHR12RD[11:0] (хранится в DHR1[11:0]), DAC channel2 пишется в биты DAC_DHR12LD[27:16] (хранится в DHR2[11:0])

Данные из регистра DAC_DHRyyyD сдвигаются и пишутся в регистры DHR1 и DHR2. Регистры DHR1 и DHR2 пишутся в регистры DOR1 и DOR2 программно или по внешнему сигналу.

Рис. 66. Данные в парном режиме



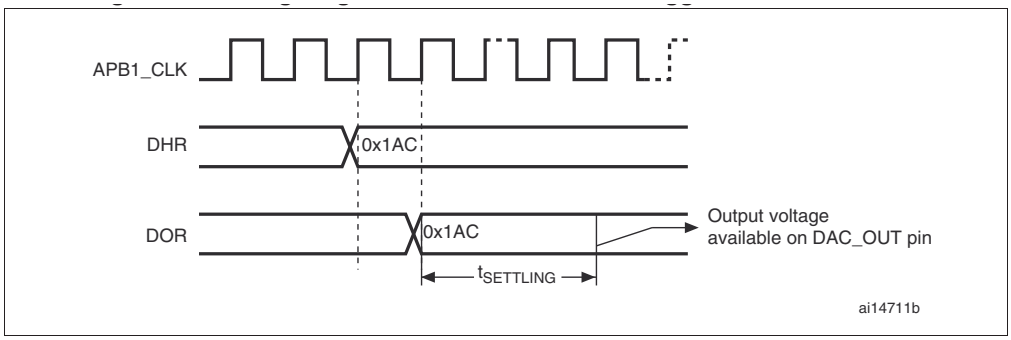
14.3.4. Преобразование ЦАП

Регистры DAC_DORx напрямую недоступны и писать для DAC channelx нужно через регистр DAC_DHRx (в DAC_DHR8Rx, DAC_DHR12Lx, DAC_DHR12Rx, DAC_DHR8RD, DAC_DHR12LD или DAC_DHR12LD).

Регистр DAC_DHRx автоматически передаётся в регистр DAC_DORx через один цикл APB1, если не выбран аппаратный запуск (бит TENx в регистре DAC_CR сброшен). Но, при аппаратном запуске (бит TENx в регистре DAC_CR) и появлении сигнала запуска данные передаются на три такта APB1 позже.

После перезаписи DAC_DHRx в DAC_DORx аналоговый сигнал появляется через $t_{SETTLING}$, зависящее от напряжения питания и выходного напряжения.

Рис. 67. Временная диаграмма с выключенным запуском TEN=0



14.3.5. Выходное напряжение ЦАП

Преобразование между 0 и V_{REF+} линейное. Выходное напряжение вычисляется по формуле:

$$\text{DACoutput} = V_{REF} \times \frac{DOR}{4096}$$

14.3.6. Выбор запуска ЦАП

При стоящем бите TENx источник запуска выбирается битами TSELx[2:0].

Таблица 75. Внешний запуск.

Источник	Тип	TSEL[2:0]
Timer 6 TRGO	Встроенные таймеры	000
Timer 8 TRGO		001
Timer 7 TRGO		010
Timer 5 TRGO		011
Timer 2 TRGO		100
Timer 4 TRGO		101
EXTI line9	Внешняя линия	110
SWTRIG	Программный запуск	111

Через три цикла APB1 после появления переднего фронта сигнала TRGO от выбранного таймера или EXTI line 9 данные из регистра DAC_DHRx передаются в регистр DAC_DORx.

При программном запуске преобразование запускается сразу после установки бита SWTRIG. SWTRIG сбрасывается аппаратно сразу после записи DAC_DORx из регистра DAC_DHRx.

NB: Биты TSELx[2:0] нельзя менять при стоящем бите ENx.

Программный запуск занимает только один такт APB1 на передачу DAC_DHRx —> DAC_DORx.

14.3.7. Запрос DMA

Для двух каналов ЦАП используются два канала ПДП.

Запрос DMA генерируется только при внешнем запуске (не программном) и установленном бите DMAENx. Затем значение регистра DAC_DHRx передаётся в регистр DAC_DORx.

В парном режиме, если установлены оба бита DMAENx, то генерируются два запроса DMA. Если нужен только один запрос DMA, то и ставить нужно один бит DMAENx. Так в парном режиме можно два канала DAC передавать одним каналом DMA по одному запросу.

Исчерпание DMA

Запросы DAC DMA не имеют очереди, так что если второй внешний запуск появляется до получения подтверждения предыдущего, то нового запроса не выдается, ставится флаг исчерпания DMA channelx (бит DMAUDRx в регистре DAC_SR). Передачи DMA прекращаются и запросы не удовлетворяются. DAC channelx продолжает выдавать старое данное.

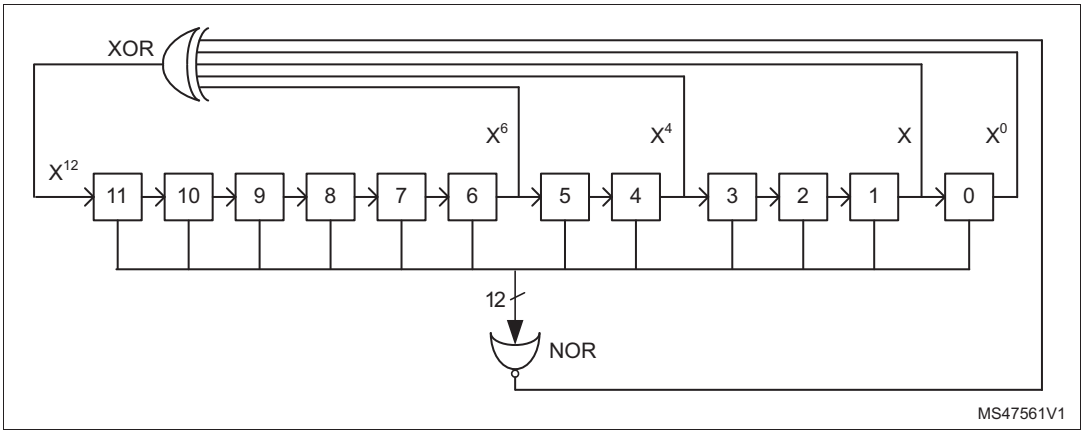
Тут надо снять флаг DMAUDRx записью “1”, очистить бит DMAEN потока DMA и заново инициализировать DMA и DAC channelx. А частоту запуска DAC надо бы пересмотреть.

Для каждого DAC channelx выдаётся прерывание, если разрешено соответствующим битом DMAUDRIEx в регистре DAC_CR.

14.3.8. Генерация шума

Шумоподобный сигнал переменной амплитуды использует Регистр Сдвига с Линейной обратной связью (LFSR). Генератор шума DAC выбирается установкой WAVEx[1:0] в “01”. Предзагруженное значение регистра LFSR равно 0xAAA. Этот регистр обновляется каждые три цикла APB1 после запуска следуя особому алгоритму.

Рис. 68. Алгоритм вычисления LFSR.

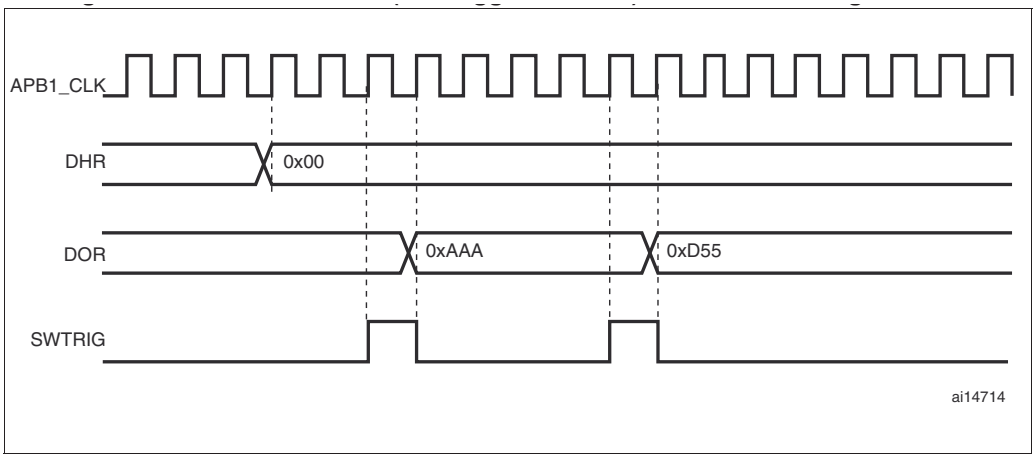


Значение LFSR можно частично или полностью маскировать с помощью битов MAMPx[3:0] в регистре DAC_CR. Оно добавляется к регистру DAC_DHRx без переполнения и затем пересылается в регистр DAC_DORx.

Если LFSR равен 0x0000, то в него вставляется ‘1’ (антиблокировка).

Сбрасывается генератор LFSR сбросом битов WAVEx[1:0].

Рис. 69. Программный запуск ЦАП с включённым LFSR.



NB. Запуск генератора шума нужно разрешать битом TENx в регистре DAC_CR.

14.3.9. Генерация треугольного сигнала

К постоянному или медленно меняющемуся выходному сигналу ЦАП можно добавлять треугольный сигнал малой амплитуды. Генератор треугольника выбирается записью “10” в WAVEx[1:0]. Амплитуда назначается с помощью битов MAMPx[3:0] в регистре DAC_CR. Внутренний

реверсивный счётчик инкрементируется каждые три такта APB1 после каждого запуска. Его значение добавляется к регистру DAC_DHRx без переполнения и сумма пишется в DAC_DORx. Счётчик инкрементируется до максимальной амплитуды, заданной битами MAMPx[3:0] и затем инкрементируется до 0, затем снова инкрементируется и т. д.

Останавливается генератор сбросом битов WAVEx[1:0].

Рис. 70. Генерация треугольника.

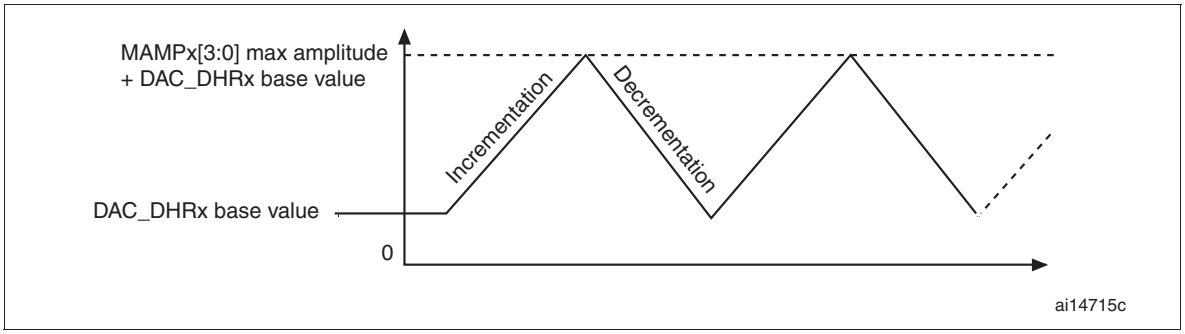
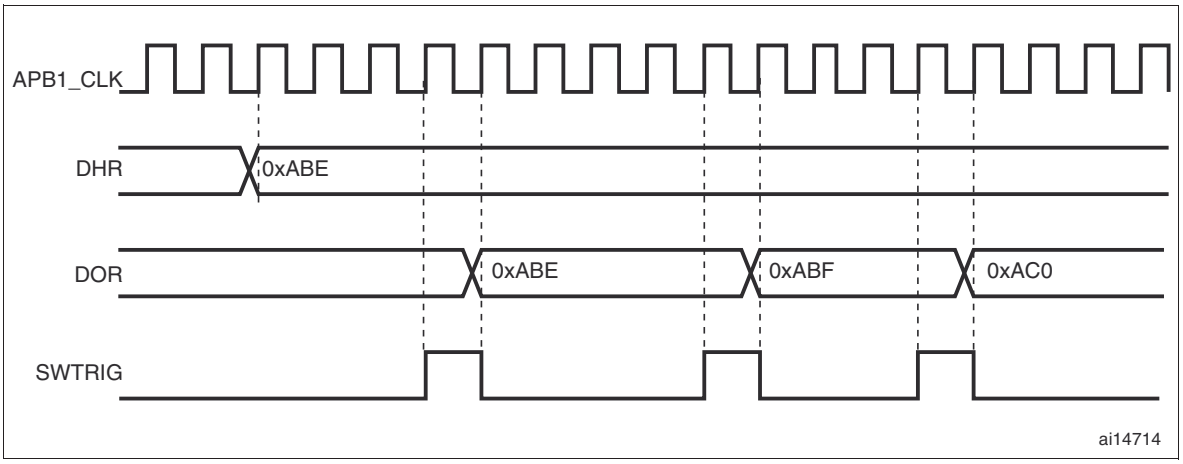


Рис. 71. Программный запуск ЦАП с включённым генератором треугольника.



NB. Запуск генератора шума нужно разрешать битом TENx в регистре DAC_CR.
Биты MAMPx[3:0] нужно писать до включение ЦАП, иначе их не изменить.

14.4. Парный режим ЦАП

Для одновременной работы двух каналов DAC есть три парных регистра: DHR8RD, DHR12RD и DHR12LD. Для управления обоими каналами DAC нужен доступ к одному регистру. Есть одиннадцать режимов преобразования для двух каналов и парных регистров. Но можно пользоваться и раздельными регистрами DHRx.

14.4.1. Независимый запуск ЦАП без генераторов

Последовательность запуска:

- Разрешить запуск обоих ЦАП битами TEN1 and TEN2
- Выбрать разные источники запуска в битах TSEL1[2:0] и TSEL2[2:0]
- Записать парные данные каналов в нужный регистр DHR (DAC_DHR12RD, DAC_DHR12LD или DAC_DHR8RD)

Через три такта APB1 после сигнала запуска DAC channel1 регистр DHR1 пишется в DAC_DOR1.

Через три такта APB1 после сигнала запуска DAC channel2 регистр DHR2 пишется в DAC_DOR2.

14.4.2. Независимый запуск ЦАП с одинаковым LFSR

Последовательность запуска:

- Разрешить запуск обоих ЦАП битами TEN1 and TEN2
- Выбрать разные источники запуска в битах TSEL1[2:0] и TSEL2[2:0]
- Записать “01” в биты WAVEx[1:0] обоих каналов и одинаковую маску LFSR в биты MAMPx[3:0]

- Записать парные данные каналов в нужный регистр DHR (DAC_DHR12RD, DAC_DHR12LD или DAC_DHR8RD)

Через три такта APB1 после сигнала запуска DAC channel1 счётчик LFSR1 прибавляется к регистру DHR1 и он пишется в DAC_DOR1. LFSR1 обновляется.

Через три такта APB1 после сигнала запуска DAC channel2 счётчик LFSR2 прибавляется к регистру DHR2 и он пишется в DAC_DOR2. LFSR2 обновляется.

14.4.3. Независимый запуск ЦАП с разными LFSR

Последовательность запуска:

- Разрешить запуск обоих ЦАП битами TEN1 and TEN2
- Выбрать разные источники запуска в битах TSEL1[2:0] и TSEL2[2:0]
- Записать “01” в биты WAVEx[1:0] обоих каналов и разные маски LFSR в битах MAMP1[3:0] и MAMP2[3:0]
- Записать парные данные каналов в нужный регистр DHR (DAC_DHR12RD, DAC_DHR12LD или DAC_DHR8RD)

Через три такта APB1 после сигнала запуска DAC channel1 счётчик LFSR1 с маской из MAMP1[3:0] прибавляется к регистру DHR1 и он пишется в DAC_DOR1. LFSR1 обновляется.

Через три такта APB1 после сигнала запуска DAC channel2 счётчик LFSR2 с маской из MAMP2[3:0] прибавляется к регистру DHR2 и он пишется в DAC_DOR2. LFSR2 обновляется.

14.4.4. Независимый запуск ЦАП с одинаковым треугольным сигналом

Последовательность запуска:

- Разрешить запуск обоих ЦАП битами TEN1 and TEN2
- Выбрать разные источники запуска в битах TSEL1[2:0] и TSEL2[2:0]
- Записать “1x” в биты WAVEx[1:0] обоих каналов и одну максимальную амплитуду в биты MAMPx[3:0]
- Записать парные данные каналов в нужный регистр DHR (DAC_DHR12RD, DAC_DHR12LD или DAC_DHR8RD)

Через три такта APB1 после сигнала запуска DAC channel1 счётчик треугольника с одинаковой амплитудой прибавляется к регистру DHR1 и он пишется в DAC_DOR1. Счётчик обновляется.

Через три такта APB1 после сигнала запуска DAC channel2 счётчик треугольника с одинаковой амплитудой прибавляется к регистру DHR2 и он пишется в DAC_DOR2. Счётчик обновляется.

14.4.5. Независимый запуск ЦАП с разным треугольным сигналом

Последовательность запуска:

- Разрешить запуск обоих ЦАП битами TEN1 and TEN2
- Выбрать разные источники запуска в битах TSEL1[2:0] и TSEL2[2:0]
- Записать “1x” в биты WAVEx[1:0] обоих каналов и разную максимальную амплитуду в биты MAMP1[3:0] и MAMP2[3:0]
- Записать парные данные каналов в нужный регистр DHR (DAC_DHR12RD, DAC_DHR12LD или DAC_DHR8RD)

Через три такта APB1 после сигнала запуска DAC channel1 счётчик треугольника с амплитудой из MAMP1[3:0] прибавляется к регистру DHR1 и он пишется в DAC_DOR1. Счётчик обновляется.

Через три такта APB1 после сигнала запуска DAC channel2 счётчик треугольника с амплитудой из MAMP2[3:0] прибавляется к регистру DHR2 и он пишется в DAC_DOR2. Счётчик обновляется.

14.4.6. Одновременный программный запуск ЦАП

Последовательность запуска:

- Записать парные данные каналов в нужный регистр DHR (DAC_DHR12RD, DAC_DHR12LD или DAC_DHR8RD)

Через один такт APB1 регистры DHR1 и DHR2 пишутся в регистры DAC_DOR1 и DAC_DOR2.

14.4.7. Одновременный запуск ЦАП без генераторов

Последовательность запуска:

- Разрешить запуск обоих ЦАП битами TEN1 and TEN2
- Выбрать один источник запуска обоих каналов в битах TSEL1[2:0] и TSEL2[2:0]
- Записать парные данные каналов в нужный регистр DHR (DAC_DHR12RD, DAC_DHR12LD или DAC_DHR8RD)

Через три такта APB1 регистры DHR1 и DHR2 пишутся в регистры DAC_DOR1 и DAC_DOR2.

14.4.8. Одновременный запуск ЦАП с одинаковым LFSR

Последовательность запуска:

- Разрешить запуск обоих ЦАП битами TEN1 and TEN2
- Выбрать один источник запуска обоих каналов в битах TSEL1[2:0] и TSEL2[2:0]
- Записать “01” в биты WAVEx[1:0] обоих каналов и одинаковую маску LFSR в биты MAMPx[3:0]
- Записать парные данные каналов в нужный регистр DHR (DAC_DHR12RD, DAC_DHR12LD или DAC_DHR8RD)

Через три такта APB1 счётчик LFSR1 прибавляется к регистру DHR1 и он пишется в DAC_DOR1. LFSR1 обновляется. Одновременно счётчик LFSR2 прибавляется к регистру DHR2 и он пишется в DAC_DOR2. LFSR2 обновляется.

14.4.9. Одновременный запуск ЦАП с разными LFSR

Последовательность запуска:

- Разрешить запуск обоих ЦАП битами TEN1 and TEN2
- Выбрать один источник запуска обоих каналов в битах TSEL1[2:0] и TSEL2[2:0]
- Записать “01” в биты WAVEx[1:0] обоих каналов и разные маски LFSR в битах MAMP1[3:0] и MAMP2[3:0]
- Записать парные данные каналов в нужный регистр DHR (DAC_DHR12RD, DAC_DHR12LD или DAC_DHR8RD)

Через три такта APB1 после сигнала запуска счётчик LFSR1 с маской из MAMP1[3:0] прибавляется к регистру DHR1 и он пишется в DAC_DOR1. LFSR1 обновляется.

Одновременно счётчик LFSR2 с маской из MAMP2[3:0] прибавляется к регистру DHR2 и он пишется в DAC_DOR2. LFSR2 обновляется.

14.4.10.Одновременный запуск ЦАП с одинаковым треугольным сигналом

Последовательность запуска:

- Разрешить запуск обоих ЦАП битами TEN1 and TEN2
- Выбрать один источник запуска в битах TSEL1[2:0] и TSEL2[2:0]
- Записать “1x” в биты WAVEx[1:0] обоих каналов и одну максимальную амплитуду в биты MAMPx[3:0]
- Записать парные данные каналов в нужный регистр DHR (DAC_DHR12RD, DAC_DHR12LD или DAC_DHR8RD)

Через три такта APB1 после сигнала запуска DAC channel1 счётчик треугольника с одинаковой амплитудой прибавляется к регистру DHR1 и он пишется в DAC_DOR1. Счётчик обновляется.

Одновременно в DAC channel2 счётчик треугольника с одинаковой амплитудой прибавляется к регистру DHR2 и он пишется в DAC_DOR2. Счётчик обновляется.

14.4.11.Одновременный запуск ЦАП с разным треугольным сигналом

Последовательность запуска:

- Разрешить запуск обоих ЦАП битами TEN1 and TEN2
- Выбрать один источник запуска в битах TSEL1[2:0] и TSEL2[2:0]
- Записать “1x” в биты WAVEx[1:0] обоих каналов и разную максимальную амплитуду в биты MAMP1[3:0] и MAMP2[3:0]
- Записать парные данные каналов в нужный регистр DHR (DAC_DHR12RD, DAC_DHR12LD или DAC_DHR8RD)

Через три такта APB1 после сигнала запуска DAC channel1 счётчик треугольника с амплитудой из MAMP1[3:0] прибавляется к регистру DHR1 и он пишется в DAC_DOR1. Счётчик обновляется.

Одновременно в DAC channel2 счётчик треугольника с амплитудой из MAMP2[3:0] прибавляется к регистру DHR2 и он пишется в DAC_DOR2. Счётчик обновляется.

14.5. Регистры ЦАП

Доступны только словами.

14.5.1. Регистр управления АЦП (ADC_CR)

Смещение адреса: 0x00

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			DMA EN2	MAMP2[3:0]				WAVE2[1:0]		TSEL2[2:0]			TEN2	BOFF2	EN2
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			DMA EN1	MAMP1[3:0]				WAVE1[1:0]		TSEL1[2:0]			TEN1	BOFF1	EN1
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:29 Резерв, не трогать.

— Бит 28 **DMAEN2**: Разрешение DMA DAC channel2
Ставится и снимается программно.
0: Нельзя
1: Можно

— Биты 27:24 **MAMP2[3:0]**: Маска/амплитуда DAC channel2
Пишутся программно.
0000: Немаскированный бит 0 LFSR/ Амплитуда равны 1
0001: Немаскированные биты [1:0] LFSR/ Амплитуда равны 3
0010: Немаскированные биты[2:0] LFSR/ Амплитуда равны 7
0011: Немаскированные биты[3:0] LFSR/ Амплитуда равны 15
0100: Немаскированные биты[4:0] LFSR/ Амплитуда равны 31
0101: Немаскированные биты[5:0] LFSR/ Амплитуда равны 63
0110: Немаскированные биты[6:0] LFSR/ Амплитуда равны 127
0111: Немаскированные биты[7:0] LFSR/ Амплитуда равны 255
1000: Немаскированные биты[8:0] LFSR/ Амплитуда равны 511
1001: Немаскированные биты[9:0] LFSR/ Амплитуда равны 1023
1010: Немаскированные биты[10:0] LFSR/ Амплитуда равны to 2047
≥ 1011: Немаскированные биты[11:0] LFSR/ Амплитуда равны 4095

— Биты 23:22 **WAVE2[1:0]**: Включение генератора шума/треугольника DAC channel2
Пишутся программно.
00: Нету
01: Включён генератор шума
1x: Включён генератор треугольника
NB: Используется при TEN2 = 1 (запуск DAC channel2 разрешён)

— Биты 21:19 **TSEL2[2:0]**: Выбор запуска DAC channel2
Пишутся программно.
000: Timer 6 TRGO
001: Timer 3 TRGO в сетевых, Timer 8 TRGO в устройствах высокой и XL-плотности
010: Timer 7 TRGO
011: Timer 5 TRGO
100: Timer 2 TRGO
101: Timer 4 TRGO
110: External line9
111: Программный запуск
NB: Используется при TEN2 = 1 (запуск DAC channel2 разрешён)

— Бит 18 **TEN2**: Разрешение запуска DAC channel2

Пишется программно.

0: DAC channel2 запускается через один такт APB1 после записи в регистр DAC_DHRx передачей его в регистр DAC_DOR2.

1: DAC channel2 запускается через три такта APB1 после внешнего сигнала передачей DAC_DHRx в регистр DAC_DOR2.

NB: При программном запуске задержка передачи DAC_DHRx в DAC_DOR2 составляют 1 такт APB1.

— Бит 17 **BOFF2**: Выключение выходного буфера DAC channel2

Пишется программно.

0: Выходной буфер DAC channel2 включён

1: Выходной буфер DAC channel2 выключен

— Бит 16 **EN2**: Включение DAC channel2

Пишется программно.

0: DAC channel2 выключен

1: DAC channel2 включён

— Биты 15:13 Резерв, не трогать.

— Бит 12 **DMAEN1**: Разрешение DMA DAC channel1

Ставится и снимается программно.

0: Нельзя

1: Можно

— Биты 11:8 **MAMP1[3:0]**: Маска/амплитуда DAC channel1

Пишутся программно.

0000: Немаскированный бит 0 LFSR/ Амплитуда равны 1

0001: Немаскированные биты [1:0] LFSR/ Амплитуда равны 3

0010: Немаскированные биты[2:0] LFSR/ Амплитуда равны 7

0011: Немаскированные биты[3:0] LFSR/ Амплитуда равны 15

0100: Немаскированные биты[4:0] LFSR/ Амплитуда равны 31

0101: Немаскированные биты[5:0] LFSR/ Амплитуда равны 63

0110: Немаскированные биты[6:0] LFSR/ Амплитуда равны 127

0111: Немаскированные биты[7:0] LFSR/ Амплитуда равны 255

1000: Немаскированные биты[8:0] LFSR/ Амплитуда равны 511

1001: Немаскированные биты[9:0] LFSR/ Амплитуда равны 1023

1010: Немаскированные биты[10:0] LFSR/ Амплитуда равны to 2047

≥ 1011: Немаскированные биты[11:0] LFSR/ Амплитуда равны 4095

— Биты 7:6 **WAVE1[1:0]**: Включение генератора шума/треугольника DAC channel1

Пишутся программно.

00: Нету

01: Включён генератор шума

1x: Включён генератор треугольника

NB: Используется при TEN1 = 1 (запуск DAC channel1 разрешён)

— Биты 5:3 **TSEL1[2:0]**: Выбор запуска DAC channel1

Пишутся программно.

000: Timer 6 TRGO

001: Timer 3 TRGO в сетевых, Timer 8 TRGO в устройствах высокой и XL-плотности

010: Timer 7 TRGO

011: Timer 5 TRGO

100: Timer 2 TRGO

101: Timer 4 TRGO

110: External line9

111: Программный запуск

NB: Используется при TEN1 = 1 (запуск DAC channel1 разрешён)

— Бит 2 **TEN1**: Разрешение запуска DAC channel1

Пишется программно.

0: DAC channel1 запускается через один такт APB1 после записи в регистр DAC_DHRx передачей его в регистр DAC_DOR1.

1: DAC channel1 запускается через три такта APB1 после внешнего сигнала передачей DAC_DHRx в регистр DAC_DOR1.

NB: При программном запуске задержка передачи DAC_DHRx в DAC_DOR1 составляют 1 такт APB1.

— Бит 1 **BOFF1**: Выключение выходного буфера DAC channel1

Пишется программно.

0: Выходной буфер DAC channel1 включён

1: Выходной буфер DAC channel1 выключен

— Бит 0 **EN1**: Включение DAC channel1

Пишется программно.

0: DAC channel1 выключен

1: DAC channel1 включён

14.5.2. Регистр программного запуска АЦП (ADC_SWTRIGR)

Смещение адреса: 0x04

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														SWTRIG2	SWTRIG1
														w	w

— Биты 31:2 Резерв, не трогать.

— Бит 1 **SWTRIG2**: Разрешение программного запуска DAC channel2

Только запись.

0: Нельзя

1: Можно

NB: Сбрасывается аппаратно через один такт после записи DAC_DHR2 в DAC_DOR2.

— Бит 0 **SWTRIG1**: Разрешение программного запуска DAC channel1

Только запись.

0: Нельзя

1: Можно

NB: Сбрасывается аппаратно через один такт после записи DAC_DHR1 в DAC_DOR1.

14.5.3. Регистр хранения правых 12-бит данных канала 1 (DAC_DHR12R1)

Смещение адреса: 0x08

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:12 Резерв, не трогать.

— Биты 11:0 **DACC1DHR[11:0]**: Выравненные вправо 12-бит данные DAC channel1.

14.5.4. Регистр хранения левых 12-бит данных канала 1 (DAC_DHR12L1)

Смещение адреса: 0x0C

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]												Reserved			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

— Биты 31:16 Резерв, не трогать.

— Биты 15:4 **DACC1DHR[11:0]**: Выравненные влево 12-бит данные DAC channel1.

— Биты 3:0 Резерв, не трогать.

14.5.5. Регистр хранения правых 8-бит данных канала 1 (DAC_DHR8R1)

Смещение адреса: 0x10

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								DACC1DHR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:8 Резерв, не трогать.

— Биты 7:0 **DACC1DHR[7:0]**: Выравненные вправо 8-бит данные DAC channel1.

14.5.6. Регистр хранения правых 12-бит данных канала 2 (DAC_DHR12R2)

Смещение адреса: 0x14

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DACC2DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:12 Резерв, не трогать.

— Биты 11:0 **DACC2DHR[11:0]**: Выравненные вправо 12-бит данные DAC channel2.

14.5.7. Регистр хранения левых 12-бит данных канала 2 (DAC_DHR12L2)

Смещение адреса: 0x18

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[11:0]												Reserved			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

— Биты 31:16 Резерв, не трогать.

— Биты 15:4 **DACC2DHR[11:0]**: Выравненные влево 12-бит данные DAC channel2.

— Биты 3:0 Резерв, не трогать.

14.5.8. Регистр хранения правых 8-бит данных канала 2 (DAC_DHR8R2)

Смещение адреса: 0x1C

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								DACC2DHR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:8 Резерв, не трогать.

— Биты 7:0 **DACC2DHR[7:0]**: Выравненные вправо 8-бит данные DAC channel2.

14.5.9. Парный регистр правых 12-бит данных (DAC_DHR12RD)

Смещение адреса: 0x20

По сбросу: 0x0000 0000

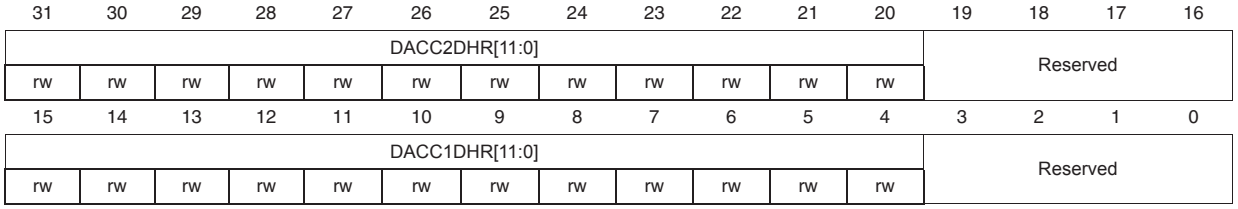
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				DACC2DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:28 Резерв, не трогать.
— Биты 27:16 **DACC2DHR[11:0]**: Выравненные вправо 12-бит данные DAC channel2.
— Биты 15:12 Резерв, не трогать.
— Биты 11:0 **DACC1DHR[11:0]**: Выравненные вправо 12-бит данные DAC channel1.

14.5.10.Парный регистр левых 12-бит данных (DAC_DHL12RD)

Смещение адреса: 0x24

По сбросу: 0x0000 0000

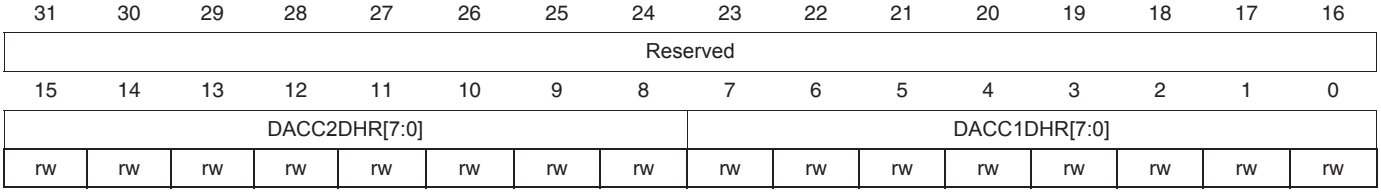


— Биты 31:20 **DACC2DHR[11:0]**: Выравненные вправо 12-бит данные DAC channel2.
— Биты 19:16 Резерв, не трогать.
— Биты 15:4 **DACC1DHR[11:0]**: Выравненные вправо 12-бит данные DAC channel1.
— Биты 3:0 Резерв, не трогать.

14.5.11.Парный регистр правых 8-бит данных (DAC_DHR8RD)

Смещение адреса: 0x28

По сбросу: 0x0000 0000

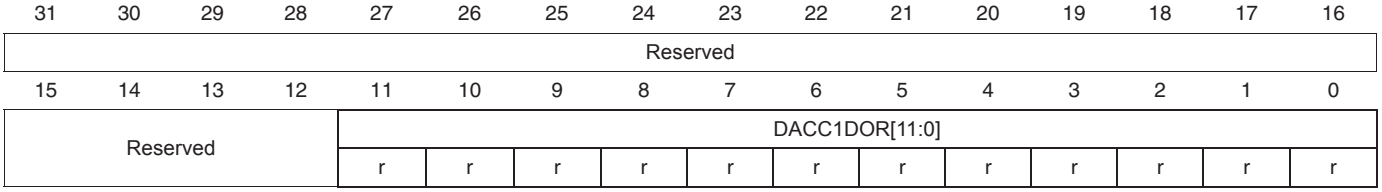


— Биты 31:16 Резерв, не трогать.
— Биты 15:8 **DACC2DHR[7:0]**: Выравненные вправо 8-бит данные DAC channel2.
— Биты 7:0 **DACC2DHR[7:0]**: Выравненные вправо 8-бит данные DAC channel1.

14.5.12.Выходной регистр данных канала 1 (DAC_DOR1)

Смещение адреса: 0x2C

По сбросу: 0x0000 0000

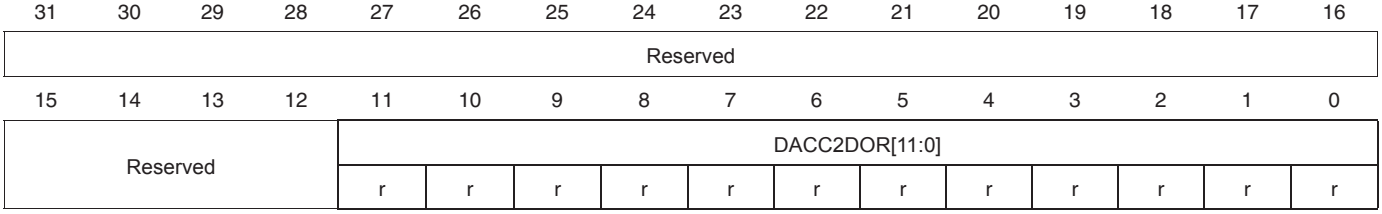


— Биты 31:12 Резерв, не трогать.
— Биты 11:0 **DACC1DOR[11:0]**: Выравненные вправо 12-бит данные DAC channel1.

14.5.13.Выходной регистр данных канала 2 (DAC_DOR2)

Смещение адреса: 0x30

По сбросу: 0x0000 0000



— Биты 31:12 Резерв, не трогать.
— Биты 11:0 **DACC2DOR[11:0]**: Выравненные вправо 12-бит данные DAC channel2.

14.5.14.Регистр статуса (DAC_SR)

Смещение адреса: 0x34

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		DMAUDR2 rc_w1	Reserved												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		DMAUDR1 rc_w1	Reserved												

— Биты 31:30 Резерв, не трогать.

— Бит 29 **DMAUDR2**: Флаг исчерпания DMA DAC channel2

Ставится программно, снимается записью.

0: Не было

1: Было

— Биты 28:14 Резерв, не трогать.

— Бит 13 **DMAUDR1**: Флаг исчерпания DMA DAC channel1

Ставится программно, снимается записью.

0: Не было

1: Было

— Биты 12:0 Резерв, не трогать.

14.5.15.Карта регистров ЦАП

Таблица 76. Карта регистров ЦАП

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	DAC_CR	Reserved		DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE 2[2:0]		TSEL2[2:0]			TEN2	BOFF2	EN2	Reserved		DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE 1[2:0]		TSEL1[2 :0]			TEN1	BOFF1	EN1
0x04	DAC_ SWTRIGR	Reserved																														SWTRIG2	SWTRIG1
0x08	DAC_ DHR12R1	Reserved																DACC1DHR[11:0]															
0x0C	DAC_ DHR12L1	Reserved																DACC1DHR[11:0]													Reserved		
0x10	DAC_ DHR8R1	Reserved																								DACC1DHR[7:0]							
0x14	DAC_ DHR12R2	Reserved																DACC2DHR[11:0]															
0x18	DAC_ DHR12L2	Reserved																DACC2DHR[11:0]													Reserved		
0x1C	DAC_ DHR8R2	Reserved																								DACC2DHR[7:0]							
0x20	DAC_ DHR12RD	Reserved		DACC2DHR[11:0]														Reserved		DACC1DHR[11:0]													
0x24	DAC_ DHR12LD	DACC2DHR[11:0]												Reserved				DACC1DHR[11:0]													Reserved		
0x28	DAC_ DHR8RD	Reserved																DACC2DHR[7:0]								DACC1DHR[7:0]							
0x2C	DAC_ DOR1	Reserved																		DACC1DOR[11:0]													
0x30	DAC_ DOR2	Reserved																		DACC2DOR[11:0]													
0x34	DAC_SR	Reserved		DMAUDR2	Reserved															DMAUDR1	Reserved												

15. Интерфейс цифровой камеры (DCMI)

15.1. Введение в DCMI

Это параллельный синхронный интерфейс для приёма высокоскоростного потока данных от внешнего 8-, 10-, 12- или 14-бит модуля CMOS камеры. Поддерживает форматы данных: YCbCr4:2:2/ RGB565 видео с прогрессивной развёрткой и сжатые данные (JPEG).

Работает с чёрно-белыми, X24 и X5 камерами. Вся предобработка делается в модуле камеры.

15.2. Основные свойства DCMI

Это:

- 8-, 10-, 12- или 14-бит параллельный интерфейс
- Встроенная/внешняя синхронизация строк и кадров
- Непрерывный или режим снимка
- Масштабирование
- Форматы данных:
  - 8/10/12/14- бит видео с прогрессивной развёрткой: монохромное или raw bayer
  - YCbCr4:2:2 видео с прогрессивной развёрткой
  - RGB 565 видео с прогрессивной развёрткой
  - Сжатые данные: JPEG

15.3. Ножки DCMI

Таблица 77. Ножки DCMI

Имя	Сигнал
D[0:13]	Вход данных
HSYNC	Вход горизонтальной синхронизации
VSYNC	Вход вертикальной синхронизации
PIXCLK	Вход тактов пикселей

15.4. Такты DCMI

Используется два домена тактов PIXCLK и HCLK. Сигналы от PIXCLK считываются по переднему фронту стабильных HCLK. Сигнал разрешения считывания выдаётся доменом HCLK при стабилизации данных от камеры. Минимальный период PIXCLK должен быть больше 2.5 HCLK.

15.5. Функциональное описание DCMI

Скорость интерфейса до 54 Мбайт/с. Состоит из 14 линий данных (D13-D0) и линии тактов пикселей (PIXCLK). У тактов PIXCLK программируемая полярность, так что данные можно читать по переднему или заднему их фронту. Данные пакуются в 32-бит регистр данных (DCMI_DR), откуда их берёт DMA. Данные от камеры могут быть строками/кадрами (режимы YUB/RGB/Bayer) или чередой JPEG изображений. Приём JPEG включают битом JPEG в DCMI_CR.

Поток данных синхронизируется либо аппаратно сигналами HSYNC (строчный) и VSYNC (кадровый), либо кодами синхронизации в потоке данных.

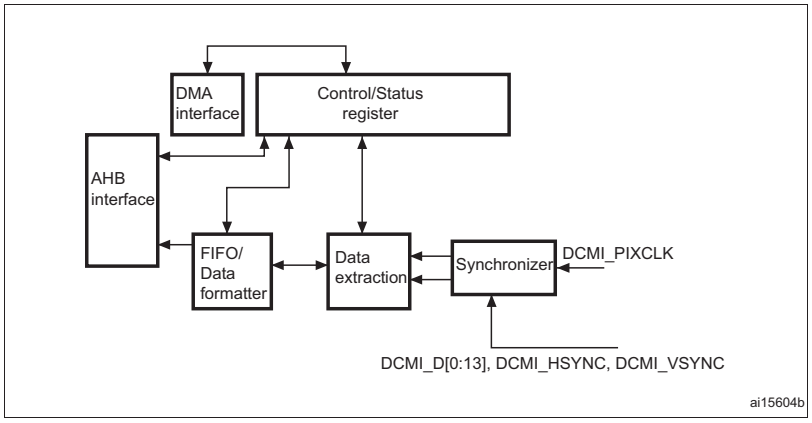


Рис. 72. Блок-схема DCMI

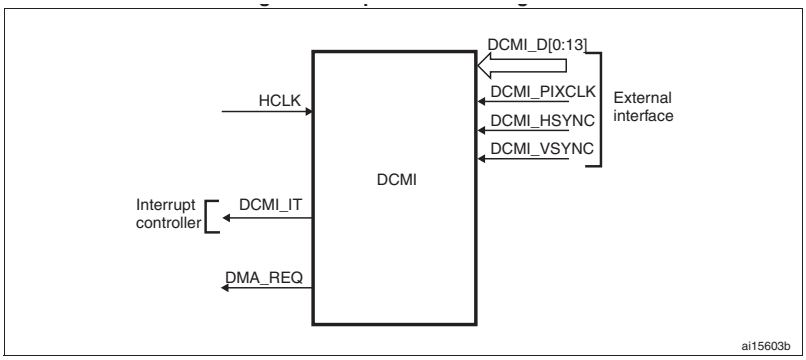


Рис. 73. Блок-схема верхнего уровня

15.5.1. Интерфейс DMA

Включается битом CAPTURE в регистре DCMI_CR. Запрос к DMA выдаётся по заполнению интерфейсом камеры 32-бит регистра данных.

15.5.2. Физический интерфейс DCMI

Состоит из 11/13/15/17 входов. Только режим ведомого.

Интерфейс камеры может считывать 8-, 10-, 12- или 14-бит данные в зависимости от битов EDM[1:0] в регистре DCMI_CR. Если ширина меньше 14 бит, то неиспользованные ножки не должны назначаться DCMI через альтернативные функции GPIO.

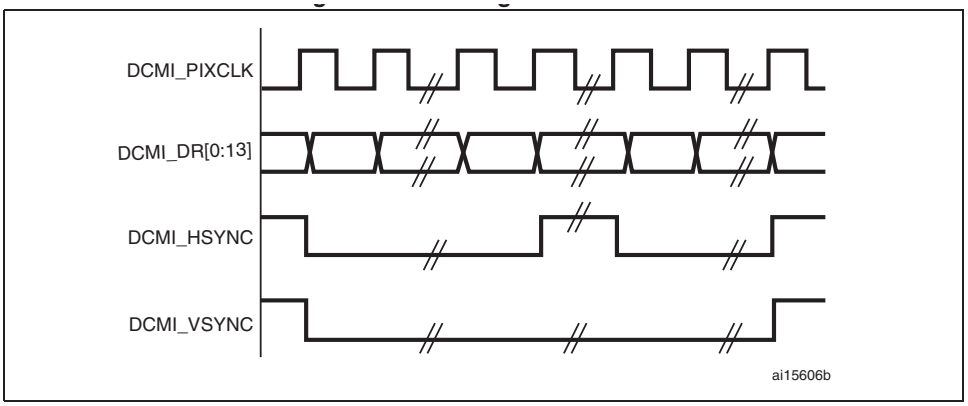
Таблица 78. Сигналы DCMI

Имя		Описание
8 bits 10 bits 12 bits 14 bits	D[0..7] D[0..9] D[0..11] D[0..13]	Данные
PIXCLK		Такты пикселей
HSYNC		Горизонтальная синхронизация / Данные верны
VSYNC		Вертикальная синхронизация

Данные синхронны с PIXCLK и меняются по переднему/заднему фронту согласно полярности.

HSYNC это начало/конец строки. VSYNC это начало/конец кадра.

Рис. 74. Сигналы DCMI



1. Считывание по заднему фронту DCMI_PIXCLK, активное состояние DCMI_HSYNC и DCMI_VSYNC — 1.
2. DCMI_HSYNC и DCMI_VSYNC могут меняться одновременно.

8-бит данные

При EDM[1:0] в DCMI_CR равных “00” берутся только 8 младших битов (D[0:7]), входы D[13:8] игнорируются. За 4 такта заполняется 32-бит слово, начиная с младшего байта.

Таблица 79. Позиции байтов данных (8-бит ширина)

Адрес байта	31:24	23:16	15:8	7:0
0	$D_{n+3}[7:0]$	$D_{n+2}[7:0]$	$D_{n+1}[7:0]$	$D_n[7:0]$
4	$D_{n+7}[7:0]$	$D_{n+6}[7:0]$	$D_{n+5}[7:0]$	$D_{n+4}[7:0]$

10-бит данные

При EDM[1:0] в DCMI_CR равных “01” берутся только 10 младших битов (D[0:9]) и пишутся в полуслово, оставшиеся биты (15:10) обнуляются. За 2 такта заполняется 32-бит слово, начиная с младшего полуслова.

Таблица 80. Позиции данных (10-бит ширина)

Адрес байта	31:26	25:16	15:10	9:0
0	0	D_{n+1}[9:0]	0	D_n[9:0]
4	0	D_{n+3}[9:0]	0	D_{n+2}[9:0]

12-бит данные

При EDM[1:0] в DCMI_CR равных “10” берутся только 12 младших битов (D[0:11]) и пишутся в полуслово, оставшиеся биты обнуляются. За 2 такта заполняется 32-бит слово, начиная с младшего полуслова.

Таблица 81. Позиции данных (12-бит ширина)

Адрес байта	31:28	27:16	15:12	11:0
0	0	D_{n+1}[11:0]	0	D_n[11:0]
4	0	D_{n+3}[11:0]	0	D_{n+2}[11:0]

14-бит данные

При EDM[1:0] в DCMI_CR равных “11” берутся только 14 младших битов (D[0:13]) и пишутся в полуслово, оставшиеся биты обнуляются. За 2 такта заполняется 32-бит слово, начиная с младшего полуслова.

Таблица 82. Позиции данных (14-бит ширина)

Адрес байта	31:30	29:16	15:14	13:0
0	0	D_{n+1}[13:0]	0	D_n[13:0]
4	0	D_{n+3}[13:0]	0	D_{n+2}[13:0]

15.5.3. Синхронизация

Синхронизация может быть встроенная или аппаратная (HSYNC & VSYNC). Встроенная синхронизация бывает только при 8-бит ширине (биты EDM[1:0] в DCMI_CR равны “00”) и хорошо образованный модуль камеры использует байты 0x00 и 0xFF ТОЛЬКО для синхронизации.

Для сжатых данных используется только аппаратная синхронизация. То есть, VSYNC это начало/конец картинки, а HSYNC это сигнал "Данные Достоверны".

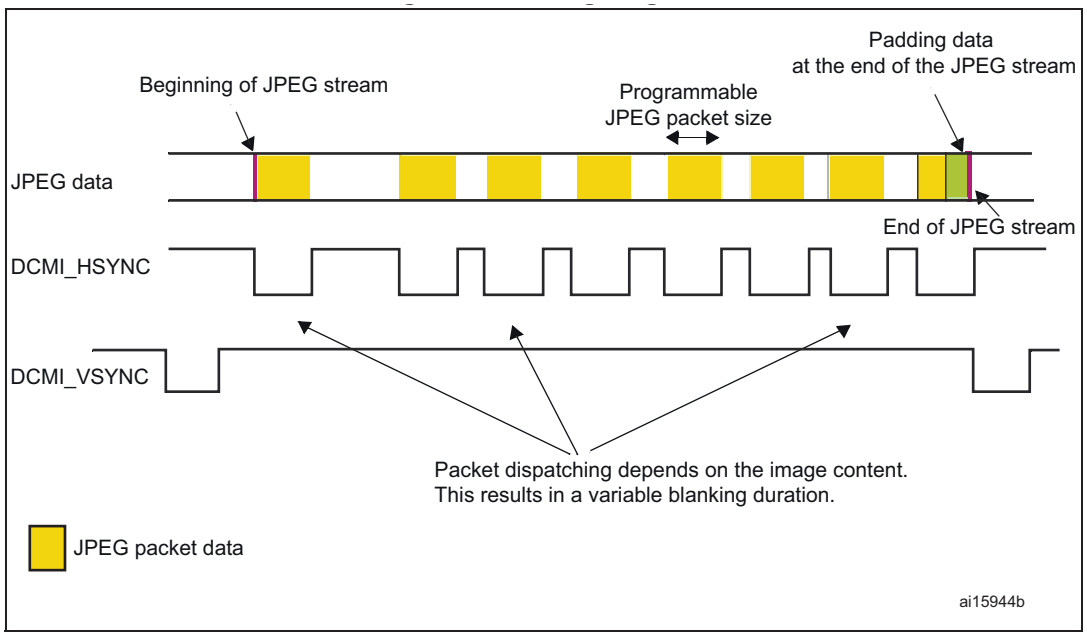


Рис. 75. Временная диаграмма

Аппаратная синхронизация

Используются сигналы синхронизации HSYNC и VSYNC. В зависимости от модуля камеры/режима данные передаются в период горизонтальной/вертикальной синхронизации. Во время активных HSYNC/VSYNC данные игнорируются.

При стоящем бите CAPTURE в регистре DCMI_CR передача данных DMA/RAM идёт по сигналу VSYNC. Снятие VSYNC это начало нового кадра. Успешные передачи могут быть непрерывными. По концу передачи кадра ставится флаг VSIF.

Встроенная синхронизация

Здесь в поток данных встраиваются 32-бит коды синхронизации с байтами 0x00/0xFF. Среди данных таких быть не должно. Есть 4 типа кодов в формате 0xFF0000XY. Такая синхронизация возможна только при 8-бит ширине данных (биты EDM[1:0] в регистре DCMI_CR равны, “00”).

При чересстрочной развёртке таких кодов 8, и интерфейс камеры её не поддерживает.

• Режим 2

Четыре кода обозначают:

- Начало кадра (FS)
- Конец кадра (FE)
- Начало строки (LS)
- Конец строки (LE)

Значения XY в формате 0xFF0000XY программируются, см. Секцию 15.8.7. Значение 0xFF в качестве “конца кадра” означает, что все неиспользованные коды рассматриваются как нормальный конец кадра.

После включения интерфейса камеры кадры начинают считываться после появления конца кадра (FE) с последующим началом кадра (FS).

• Режим 1

Он совместим с ITU656. Коды обозначают:

- SAV (активная строка) - начало строки
- EAV (активная строка) - конец строки
- SAV (гашение) - начало строки в период между кадрами
- EAV (гашение) - конец строки в период между кадрами

При этом:

- FS ≤ 0xFF
- FE ≤ 0xFF
- LS ≤ SAV (активная)
- LE ≤ EAV (активная)

В коды начала/конца кадра/строки встроены биты демаскирования. Они позволяют сравнивать только один бит для различения начала и конца.

Пример

FS = 0xA5

Демаскирование FS = 0x10

В этом случае в бит 4 встроен код начала кадра.

15.5.4. Режимы чтения

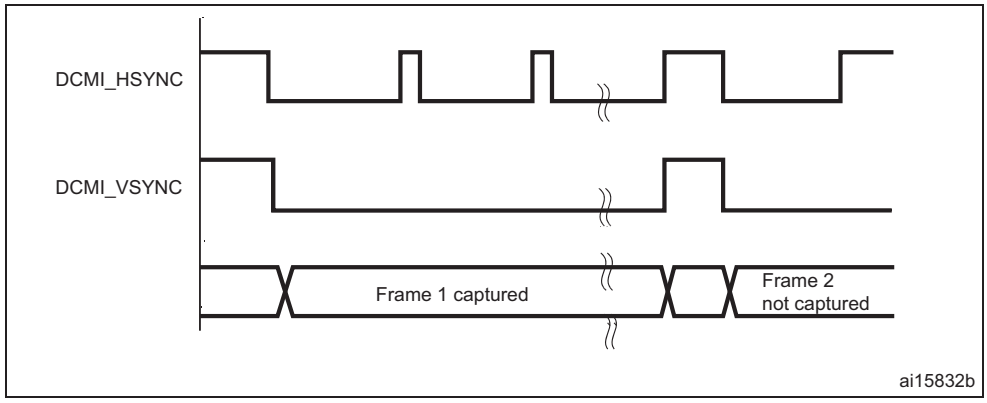
Есть два типа чтения: снимок (один кадр) и постоянный.

Снимок (один кадр)

Считывается один кадр (CM = ‘1’ в DCMI_CR). После установки бита CAPTURE в регистре DCMI_CR, интерфейс ждёт следующего начала кадра. После получения всего кадра интерфейс автоматически отключается и выдаётся прерывание IT_FRAME).

При переполнении кадр теряется и бит CAPTURE снимается.

Рис. 76. Сигналы получения снимка

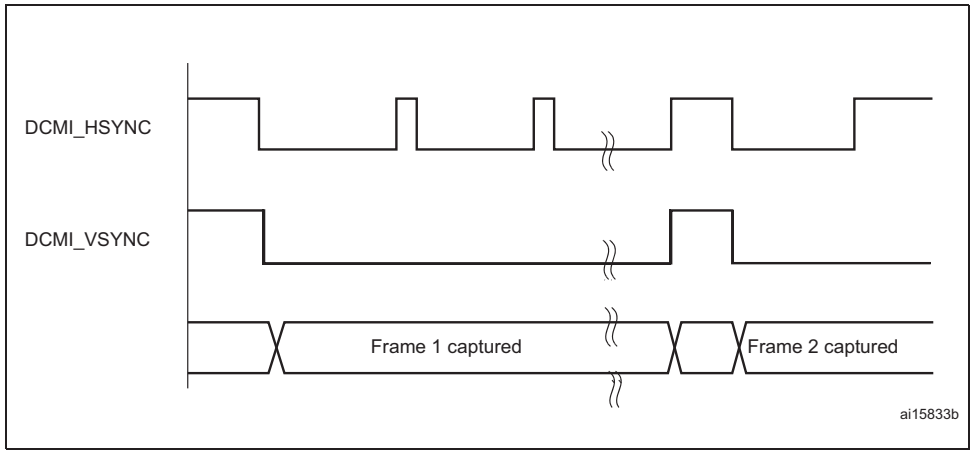


1. Здесь, активное состояние DCMI_HSYNC и DCMI_VSYNC равно 1.
2. DCMI_HSYNC и DCMI_VSYNC могут меняться одновременно.

Постоянное чтение

В этом режиме (бит CM = ‘0’ в DCMI_CR), после установки бита CAPTURE чтение кадров начинается после получения следующего VSYNC или кода начала кадра. Процесс продолжается вплоть до снятия бита CAPTURE в DCMI_CR. Приём текущего кадра завершается нормально.

Рис. 77. Сигналы получения постоянного потока



1. Здесь, активное состояние DCMI_HSYNC и DCMI_VSYNC равно 1.
2. DCMI_HSYNC и DCMI_VSYNC могут меняться одновременно.

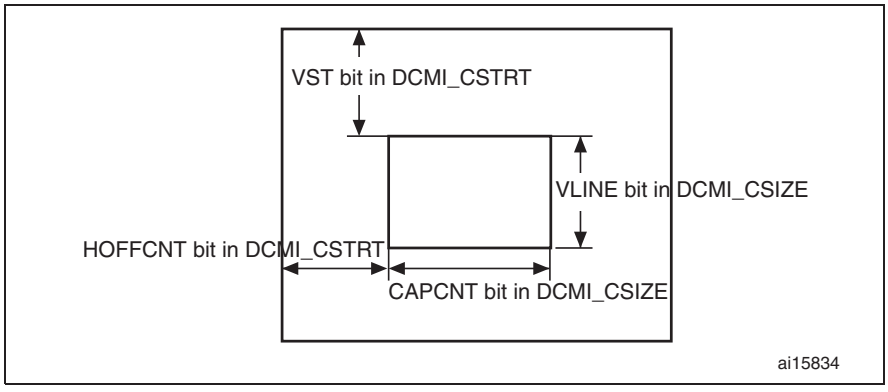
В постоянном режиме битами FCRC в регистре DCMI_CR можно задать чтение всех кадров, каждый второй и каждый четвёртый экономии для.

При аппаратной синхронизации (ESS = ‘0’ в DCMI_CR) прерывание IT_VSYNC выдаётся даже при CAPTURE = ‘0’ в DCMI_CR, так что пропускать можно желаемое только вами число кадров. При встроенной синхронизации такое невозможно.

15.5.5. Обрезание

Интерфейс камеры может вырезать прямоугольник из получаемого кадра. Координаты верхнего левого угла и размеры в строках (вертикаль) и пикселях (горизонталь) задаются в регистрах DCMI_CWSTRT и DCMI_CWSIZE. Отсчёт координат и размеров ведётся с 0.

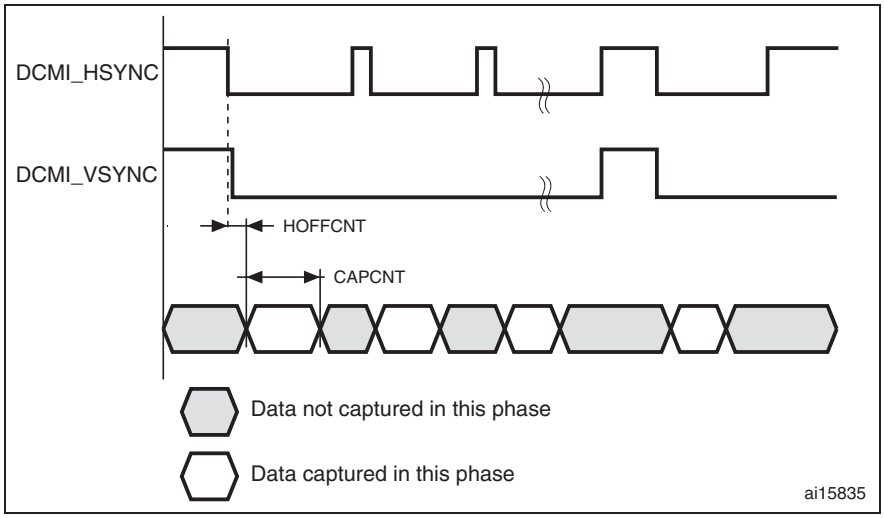
Рис. 78. Координаты и размеры окна



Значение CAPCNT должно быть кратно 4 из-за передач через DMA.

Если сигнал VSYNC возникает до достижения числа строк, указанных в регистре DCMI_CWSIZE, то приём прекращается и выдаётся прерывание IT_FRAME если разрешено.

Рис. 79. Сигналы приёма данных



1. Здесь, активное состояние DCMI_HSYNC и DCMI_VSYNC равно 1.
2. DCMI_HSYNC и DCMI_VSYNC могут меняться одновременно.

15.5.6. Формат JPEG

Приём разрешают установкой бита JPEG в регистре DCMI_CR. JPEG не содержит строк и кадров, так что сигнал VSYNC запускает приём, а HSYNC разрешает запись данных. Число байтов в строке может быть не кратно 4, так что по DMA надо передавать внимательно. Если по концу кадра принят не всё последнее 32-бит слово, то перед выдачей запроса DMA то оно дополняется нулями.

Обрезание и встроенная синхронизация в режиме JPEG невозможны.

15.5.7. FIFO

Для передач по AHB есть FIFO из четырёх слов с указателями чтения и записи. Защиты от переполнения нет и при его возникновении или ошибки в сигналах сигнализации FIFO сбрасывается и интерфейс DCMI ждут нового начала кадра.

15.6. Описание формата данных

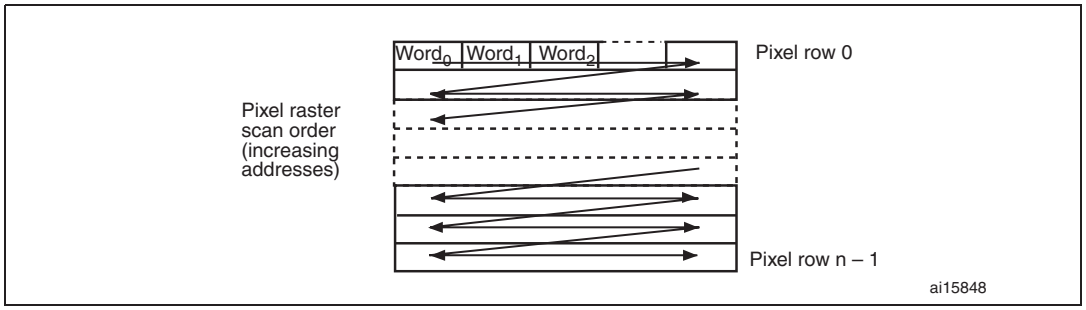
15.6.1. Форматы данных

Поддерживаются три типа данных:

- 8-бит, прогрессивная развёртка: монохром или raw Bayer
- YCbCr4:2:2, прогрессивная развёртка
- RGB565, прогрессивная развёртка. В пикселе 16 бит (5 - синий, 5 - красный, 6 - зелёный) два цикла на передачу.

Сжатые данные: JPEG. Для B&W, YCbCr и RGB, максимальный входной размер 2048 × 2048 пикселей. Без ограничений в режиме JPEG. В монохроме, RGB & YCbCr, буфер кадра в растровом режиме. 32-бит слова. Формат только little endian.

Рис. 80. Растровый порядок пикселей



15.6.2. Монохромный формат

Характеристики:

- Растровый формат
- 8 бит на пиксель

Таблица 83. Память данных в монохромном режиме

Адрес байта	31:24	23:16	15:8	7:0
0	n+3	n+2	n+1	n
4	n+7	n+6	n+5	n+4

15.6.3. Формат RGB

Характеристики:

- Растровый формат
- RGB
- Чересстрочный: один буфер: R, G & B чересстрочный: BRGBRGBRG, и т.д.
- Оптимизированный для вывода на дисплей

Планарный формат RGB совместим со стандартным форматом буфера кадра OS.

Только 16 BPP (бит на пиксель): поддерживается RGB565.

Форматы 24 BPP и шкала серого не поддерживаются. Пиксели хранятся в порядке растра. Компоненты пикселя: R (красный), G (зелёный) и B (синий). Разрешение всех компонентов одинаковое (формат 4:4:4). Кадр хранится одним куском с чередующимися компонентами на основе пикселей.

Таблица 84. Память данных в RGB прогрессивном режиме

Адрес байта	31:27	26:21	20:16	15:11	10:5	4:0
0	Red n + 1	Green n + 1	Blue n + 1	Red n	Green n	Blue n
4	Red n + 4	Green n + 3	Blue n + 3	Red n + 2	Green n + 2	Blue n + 2

15.6.4. Формат YCbCr

Характеристики:

- Растровый формат
- YCbCr 4:2:2
- Чересстрочный: один буфер: Y, Cb & Cr чересстрочный: CbYCrYCbYCr, и т.д.

Компоненты пикселя: Y (яркость или “luma”), Cb и Cr (цветность или “chroma” синий и красный). По 8 бит на компонент. Luma и chroma хранятся вместе (чередуясь).

Таблица 85. Память данных в YCbCr прогрессивном режиме

Адрес байта	31:24	23:16	15:8	7:0
0	Yn+1	Cr n	Yn	Cb n
4	Yn+3	Cr n + 2	Yn+2	Cb n + 2

15.7. Прерывания DCMI

Их пять. Они маскируемые. Глобальное прерывание (`IT_DCMI`) это OR всех отдельных.

Таблица 86. Прерывания DCMI

Имя	Описание
IT_LINE	Конец строки
IT_FRAME	Конец чтения кадра
IT_OVR	Переполнение
IT_VSYNC	Синхронизация кадра
IT_ERR	Ошибка встроенной синхронизации кадра
IT_DCMI	Логическое OR предыдущих прерываний

15.8. Регистры DCMI

Доступны только словами, иначе Ошибка шины.

15.8.1. Регистр управления 1 (DCMI_CR)

Смещение адреса: 0x00

По сбросу: 0x0000 0000

31–15	14	13–12	11–10	9–8	7	6	5	4	3	2	1	0
Reserved	ENABLE	Reserved	EDM	FCRC	VSPOL	HSPOL	PCKPOL	ESS	JPEG	CROP	CM	CAPTURE
	rw		rw rw	rw rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:15 Резерв, не трогать.

— Бит 14 **ENABLE**: Включение DCMI

0: Выкл.

1: Вкл.

NB: Включать надо правильно сконфигурированный DCMI

— Биты 13:12 Резерв, не трогать.

— Биты 11:10 **EDM[1:0]**: Расширенный режим данных

00: Приём 8-бит данного на такт пикселя

01: Приём 10-бит данного на такт пикселя

10: Приём 12-бит данного на такт пикселя

11: Приём 14-бит данного на такт пикселя

— Биты 9:8 **FCRC[1:0]**: Чтение кадров в Непрерывном режиме

00: Все кадры

01: Один из двух кадров (снижение 50% нагрузки)

10: Один из 4 кадров (снижение 75% нагрузки)

11: Резерв

— Бит 7 **VSPOL**: Полярность синхронизации кадров с недоступными данными

0: VSYNC активен низким

1: VSYNC активен высоким

— Бит 6 **HSPOL**: Полярность синхронизации строк с недоступными данными

0: **H**SYNC активен низким

1: **H**SYNC активен высоким

— Бит 5 **PCKPOL**: Полярность такта пикселя

0: Задний фронт

1: Передний фронт

— Бит 4 **ESS**: Выбор синхронизации

0: Аппаратная, сигналами HSYNC/VSYNC.

1: Встроенная, кодами в потоке данных.

Note: Работает только с 8-бит данными. При стоящем ESS сигналы HSPOL/VSPOL игнорируются.

В режиме JPEG бит снят.

— Бит 3 **JPEG**: Разрешение JPEG.

0: Несжатый формат видео

1: Передача JPEG данных.

Сигнал HSYNC это разрешение данных. Обрезка и встроенная синхронизация невозможны.

— Бит 2 **CROP**: Обрезание.

0: Всё изображение, общее число байтов кадра должно быть кратно 4.

1: Выделенное окно, Если размер окна достигает размеров картинки, то она и принимается.

— Бит 1 **CM**: Режим приёма.

0: Непрерывный - Передача через DMA до программного снятия бита CAPTURE.

1: Снимок - Только один кадр через DMA. По концу передачи бит CAPTURE снимается сам.

— Бит 0 **CAPTURE**: Включение приёма.

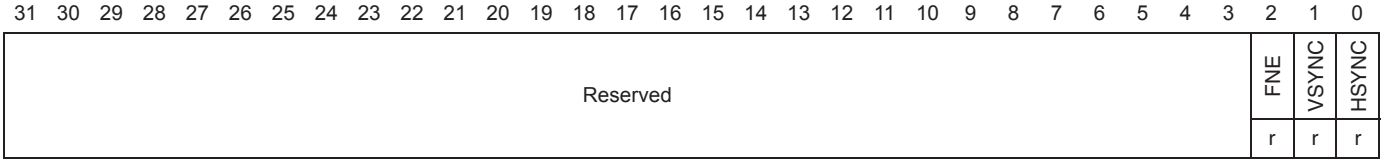
0: Нельзя.

1: Поехали, ждём первого сигнала начала кадра .

15.8.2. Регистр состояния (DCMI_SR)

Смещение адреса: 0x04

По сбросу: 0x0000 0000



— Биты 31:3 Резерв, не трогать.

— Бит 2 **FNE**: FIFO не пуст.

0: FIFO пуст.

1: FIFO не пуст.

— Бит 1 **VSYNC**: Состояние ножки VSYNC с установленной полярностью.

При встроенной синхронизации и стоящем бите CAPTURE:

0: активный кадр

1: Синхросигнал кадров

— Бит 0 **HSYNC**: Состояние ножки HSYNC с установленной полярностью.

При встроенной синхронизации и стоящем бите CAPTURE:

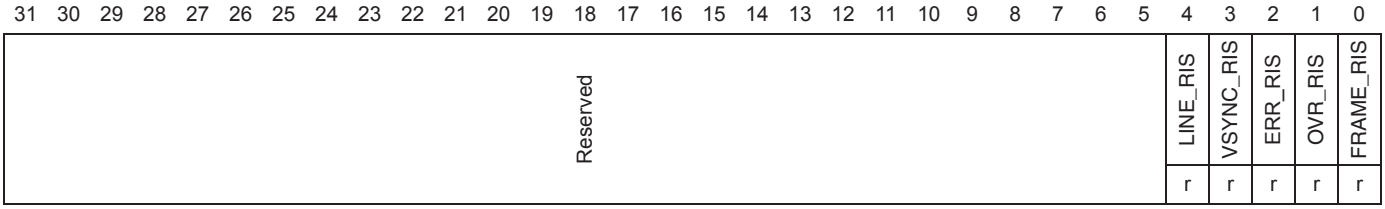
0: активная строка

1: Синхросигнал строк

15.8.3. Регистр состояния запросов прерываний (DCMI_RIS)

Смещение адреса: 0x08

По сбросу: 0x0000 0000



Состояние запросов прерываний до маскирования в регистре DCMI_IER.

— Биты 31:5 Резерв, не трогать.

— Бит 4 **LINE_RIS**: Прерывание строки.

Ставится при переходе HSYNC в активное состояние даже при неверной строке.

При встроенной синхронизации встаёт только при стоящем бите CAPTURE в DCMI_CR.
Снимается записью ‘1’ в бит LINE_ISC регистра DCMI_ICR.

— Бит 3 **VSYNC_RIS**: Прерывание кадра.

Ставится при переходе VSYNC в активное состояние.

При встроенной синхронизации встаёт только при стоящем бите CAPTURE в DCMI_CR.
Снимается записью ‘1’ в бит VSYNC_ISC регистра DCMI_ICR.

— Бит 2 **ERR_RIS**: Прерывание ошибки синхронизации.

0: Не было

1: Неверный порядок кодов синхронизации

Только для встроенной синхронизации.
Снимается записью ‘1’ в бит ERR_ISC регистра DCMI_ICR.

— Бит 1 **OVR_RIS**: Прерывание ошибки переполнения буфера.

0: Не было

1: Было, FiFO разрушен.

Снимается записью ‘1’ в бит OVR_ISC регистра DCMI_ICR.

— Бит 0 **FRAME_RIS**: Прерывание конца ввода кадра или окна.

0: Даже не было.

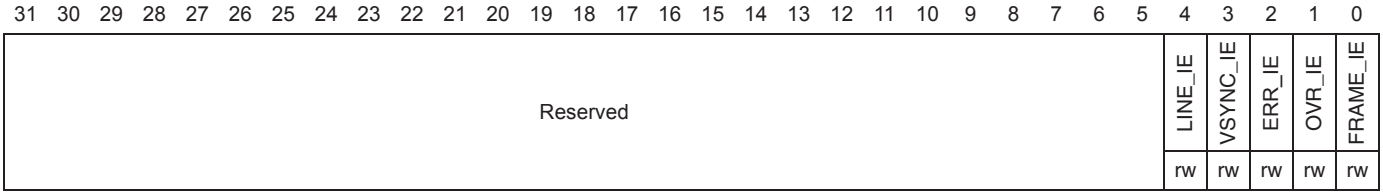
1: Сделано.

При вводе окна ставится по вводу последней строки даже пустого окна.
Снимается записью ‘1’ в бит FRAME_ISC регистра DCMI_ICR.

15.8.4. Регистр разрешения прерываний (DCMI_IER)

Смещение адреса: 0x0C

По сбросу: 0x0000 0000



Разрешает прерывание в регистре DCMI_IER установкой соответствующего бита в 1.

— Биты 31:5 Резерв, не трогать.

— Бит 4 **LINE_IE**: Прерывание приёма конца строки.

— Бит 3 **VSYNC_IE**: Прерывание приёма конца кадра.

— Бит 2 **ERR_IE**: Прерывание ошибки синхронизации.
Только для встроенной синхронизации.

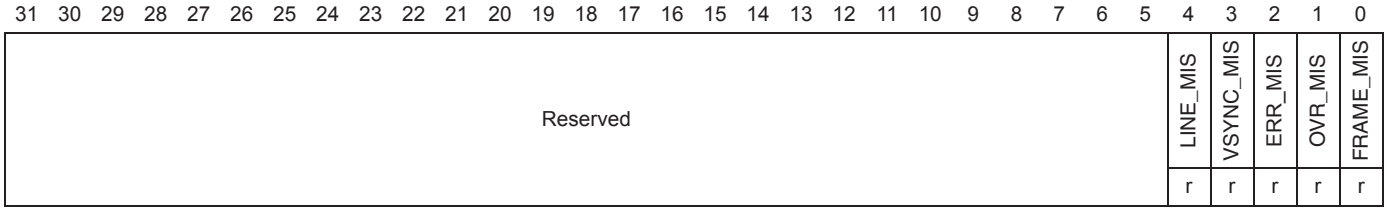
— Бит 1 **OVR_IE**: Прерывание ошибки переполнения буфера.

— Бит 0 **FRAME_IE**: Прерывание конца ввода кадра или окна.

15.8.5. Регистр состояния маскированных прерываний (DCMI_MIS)

Смещение адреса: 0x010

По сбросу: 0x0000 0000



Возвращает 1 при стоящих соответствующих битах и в регистре DCMI_IER и в DCMI_RIS.

— Биты 31:5 Резерв, не трогать.

— Бит 4 **LINE_MIS**: Прерывание приёма конца строки.

— Бит 3 **VSYNC_MIS**: Прерывание приёма конца кадра.

— Бит 2 **ERR_MIS**: Прерывание ошибки синхронизации.
Только для встроенной синхронизации.

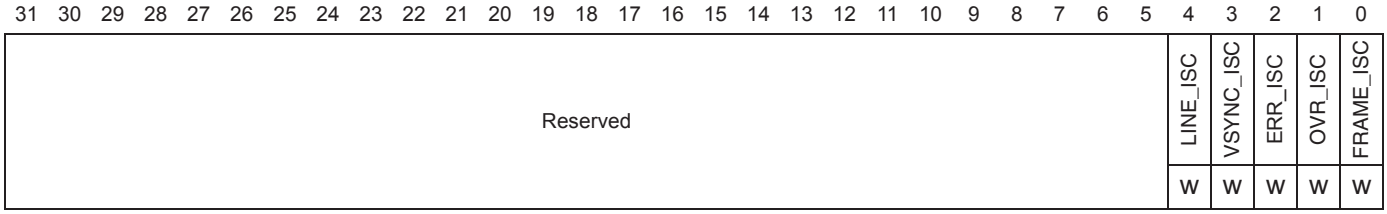
— Бит 1 **OVR_MIS**: Прерывание ошибки переполнения буфера.

— Бит 0 **FRAME_MIS**: Прерывание конца ввода кадра или окна.

15.8.6. Регистр очистки прерываний (DCMI_ICR)

Смещение адреса: 0x014

По сбросу: 0x0000 0000



Запись 1 снимает соответствующие биты и в регистрах DCMI_MIS и DCMI_RIS.

— Биты 31:5 Резерв, не трогать.

— Бит 4 **LINE_ISC**: Прерывание приёма конца строки.

— Бит 3 **VSYNC_ISC**: Прерывание приёма конца кадра.

— Бит 2 **ERR_ISC**: Прерывание ошибки синхронизации.
Только для встроенной синхронизации.

— Бит 1 **OVR_ISC**: Прерывание ошибки переполнения буфера.

— Бит 0 **FRAME_ISC**: Прерывание конца ввода кадра или окна.

15.8.7. Регистр кодов встроенной синхронизации (DCMI_ESCR)

Смещение адреса: 0x018

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FEC								LEC								LSC								FSC							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Коды состоят из 4 ,fqnjd в форме 0xFF0000XY.

— Биты 31:24 **FEC**: Конец кадра.
Если XY равен 0xFF, то все коды 0xFF0000XY интерпретируются как конец кадра.

— Биты 23:16 **LEC**: Конец строки.

— Биты 15:8 **LSC**: Начало строки.

— Биты 7:0 **FSC**: Начало кадра.
Если XY равен 0xFF, то начало кадра не обнаруживают, им считается первое появление LSC после FEC.

15.8.8. Регистр демаскирования встроенной синхронизации (DCMI_ESUR)

Смещение адреса: 0x01C

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FEU								LEU								LSU								FSU							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Маска битов для сравнения кодов встроенной синхронизации.
0: Соответствующий бит в байте из DCMI_ESCR не сравнивается с байтом в пришедших данных.
1: Соответствующий бит в байте из DCMI_ESCR сравнивается с байтом в пришедших данных.

— Биты 31:24 **FEU**: Конец кадра.
Если XY равен 0xFF, то все коды 0xFF0000XY интерпретируются как конец кадра.

— Биты 23:16 **LEU**: Конец строки.

— Биты 15:8 **LSU**: Начало строки.

— Биты 7:0 **FSU**: Начало кадра.
Если XY равен 0xFF, то начало кадра не обнаруживают, им считается первое появление LSC после FEC.

15.8.9. Регистр начала окна обрезки (DCMI_CWSTRT)

Смещение адреса: 0x020

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			VST[12:0													Reserv ed		HOFFCNT[13:0]													
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:29 Резерв, не трогать.

— Биты 28:16 **VST[12:0]**: Номер строки начала окна обрезания.

— Биты 15:14 Резерв, не трогать.

— Биты 15:8 **HOFFCNT[13:0]**: Номер пикселя начала окна обрезания.

15.8.10.Регистр начала окна обрезки (DCMI_CWSIZE)

Смещение адреса: 0x024

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		VLINE13:0]														Reserved		CAPCNT[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:30 Резерв, не трогать.

— Биты 29:16 **VLINE[13:0]**: Число строк окна обрезания - 1.

— Биты 15:14 Резерв, не трогать.

— Биты 15:8 **CAPCNT[13:0]**: Число пикселей окна обрезания - 1.

15.8.11.Регистр данных (DCMI_DR)

Смещение адреса: 0x028

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Byte3								Byte2								Byte1								Byte0							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

— Биты 31:24 Байт 3.

— Биты 23:16 Байт 2.

— Биты 15:8 Байт 1.

— Биты 7:0 Байт 0.

15.8.12.Карта регистров DCMI

Таблица 87.

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	DCMI_CR	Reserved																	ENABLE	Reserved		EDM		FCRC		VSPOL	HSPOL	PCKPOL	ESS	JPEG	CROP	CM	CAPTURE
	Reset value																		0			0	0	0	0	0	0	0	0	0	0	0	0
0x04	DCMI_SR	Reserved																													FNE	VSYNC	HSYNC
	Reset value																														0	0	0
0x08	DCMI_RIS	Reserved																											LINE_RIS	VSYNC_RIS	ERR_RIS	OVR_RIS	FRAME_RIS
	Reset value																												0	0	0	0	0
0x0C	DCMI_IER	Reserved																											LINE_IE	VSYNC_IE	ERR_IE	OVR_IE	FRAME_IE
	Reset value																												0	0	0	0	0
0x10	DCMI_MIS	Reserved																											LINE_MIS	VSYNC_MIS	ERR_MIS	OVR_MIS	FRAME_MIS
	Reset value																												0	0	0	0	0
0x14	DCMI_ICR	Reserved																											LINE_ISC	VSYNC_ISC	ERR_ISC	OVR_ISC	FRAME_ISC
	Reset value																												0	0	0	0	0
0x18	DCMI_ESCR	FEC								LEC									LSC										FSC				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	DCMI_ESUR	FEU								LEU									LSU										FSU				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	DCMI_CWSTRT	Reserved								VST[12:0									Reserved			HOFFCNT[13:0]											
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	DCMI_CWSIZE	Reserved			VLINE13:0]														Reserved			CAPCNT[13:0]											
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	DCMI_DR	Byte3								Byte2									Byte1										Byte0				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16. Контроллер LCD-TFT (LTDC)

Только для STM32F429xx/439xx.

16.1. Введение в LTDC

Это параллельный цифровой RGB интерфейс для разнообразных панелей LCD и TFT.

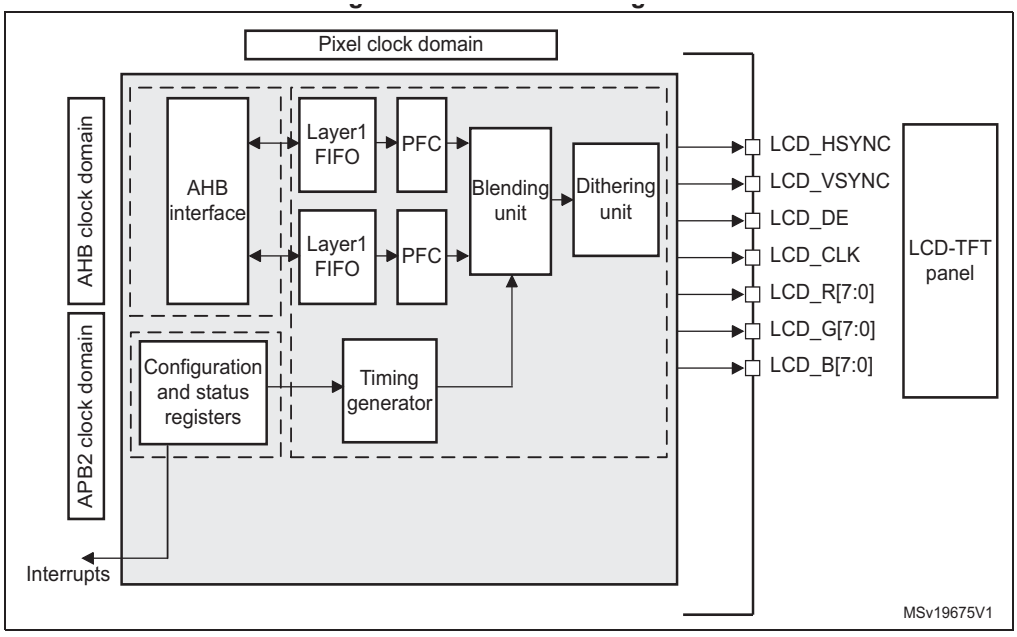
16.2. Основные свойства LTDC

- 24-бит RGB выход (RGB888)
- 2 слоя отображения со своими FIFO (64x32-бит)
- Свои Таблицы выбора цвета (CLUT) до 256 слов (256x24-бит) для слоёв
- Разрешение до XGA (1024x768)
- Программируемые времянки для разных панелей
- Программируемый цвет фона
- Программируемая полярность HSync, VSync Разрешения Данных
- До 8 входных форматов цвета на слой
  - ARGB8888
  - RGB888
  - RGB565
  - ARGB1555
  - ARGB4444
  - L8 (8-бит Luminance или CLUT)
  - AL44 (4-бит alpha + 4-бит luminance)
  - AL88 (8-бит alpha + 8-бит luminance)
- Выход псевдослучайного размывания для младших битов канала
  - Ширина размывания по 2-бита Red, Green, Blue
- Гибкое смешивание двух слоёв с помощью альфа (по пикселю или постоянное)
- Ключевые цвета (цвет прозрачного)
- Программируемые позиция и размер Окна
- Поддержка TFT дисплеев
- Ведущий шины AHB с группами по 16 слов
- До 4 программируемых прерываний

16.3. Функциональное описание LTDC

16.3.1. Блок-схема LTDC

Рис. 81. Блок-схема LTDC



FIFO слоя: По одному FIFO 64x32 бит на слой.

PFC: Преобразователь входного формата пикселей в слова.

Интерфейс AHB: Передатчик данных из памяти в FIFO.

Смешивание, Размывание и Генератор времянки: См. Секции 16.4.1 и 16.4.2.

16.3.2. Сброс и такты LTDC

Используется 3 домена тактов:

- Такты AHB (HCLK) для передачи данных из памяти в FIFO и регистр конфигурации буфера кадра
- Такты APB2 (PCLK2) для глобального регистра конфигурации и регистров прерываний
- Домен тактов пикселей (LCD_CLK) для сигналов интерфейса LCD-TFT, создания пикселей и конфигурации слоёв. Выход LCD_CLK должен соответствовать панели. конфигурируются LCD_CLK в регистре PLLSAI (см. секцию RCC).

Таблица 88. Регистры LTDC и домены тактов

Регистры LTDC	Домен тактов
LTDC_LxCR	HCLK
LTDC_LxCFBAR	
LTDC_LxCFBLR	
LTDC_LxCFBLNR	
LTDC_SRCR	PCLK2
LTDC_IER	
LTDC_ISR	
LTDC_ICR	
LTDC_SSCR	Pixel Clock (LCD_CLK)
LTDC_BPCR	
LTDC_AWCR	
LTDC_TWCR	
LTDC_GCR	
LTDC_BCCR	
LTDC_LIPCR	
LTDC_CPSR	
LTDC_CDSR	
LTDC_LxWHPCR	
LTDC_LxWVPCR	
LTDC_LxCKCR	
LTDC_LxPFCR	
LTDC_LxCACR	
LTDC_LxDCCR	
LTDC_LxBFCR	
LTDC_LxCLUTWR	

При доступе к регистрам LTDC шина APB2 приостанавливается если идут операции:

- Запись регистров на 6xPCKL2 периодов + 5xLCD_CLK периодов (5xHCLK периодов для регистров домена тактов AHB)
- Чтение регистров на 7xPCKL2 периодов + 5xLCD_CLK периодов (5xHCLK периодов для регистров домена тактов AHB).

Для регистров домена тактов PCLK2 шина APB2 приостанавливается при записи на 6xPCKL2 периодов и на 7xPCKL2 периодов при чтении.

Контроллер LCD сбрасывают установкой соответствующего бита в регистре RCC_APB2RSTR. Он сбрасывает три домена тактов.

16.3.3. Ножки и сигналы интерфейса LCD-TFT

Таблица 89. Ножки и сигналы интерфейса LCD-TFT

Сигналы LCD-TFT	I/O	Описание
LCD_CLK	O	Выход тактов
LCD_HSYNC	O	Горизонтальная синхронизация
LCD_VSYNC	O	Вертикальная синхронизация
LCD_DE	O	НЕ-Разрешение данных
LCD_R[7:0]	O	Данные: 8-бит Красный
LCD_G[7:0]	O	Данные: 8-бит Зелёный
LCD_B[7:0]	O	Данные: 8-бит Синий

Ножки контроллера LCD-TFT конфигурируются пользователем. Неиспользованные ножки можно использовать в других целях.

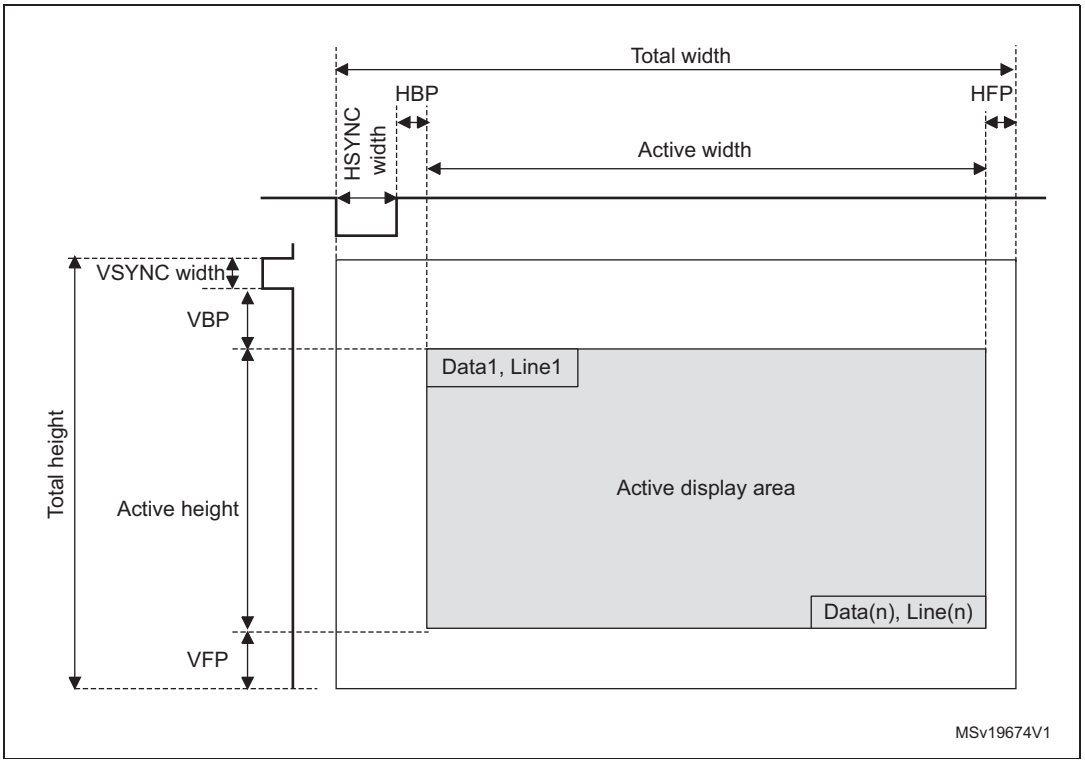
Не занятые ножки данных при меньшей ширине цвета (не 8 бит) надо подключать к старшему биту линий данных своего цвета RGB контроллера LCD-TFT.

16.4. Программируемые параметры LTDC

16.4.1. Глобальные параметры конфигурации LTDC

Синхронные времянки:

Рис. 82. Синхронные времянки LCD-TFT



NB: HBP и HFP это Горизонтальное заднее и переднее крыльцо соответственно. VBP и VFP и это Вертикальное заднее и переднее крыльцо соответственно.

Программируемые синхронные времянки LCD-TFT:

- Ширина HSYNC и VSYNC: В регистре `LTDC_SSCR` пишут ширину HSYNC-1 и ширину VSYNC-1.
- HBP и VBP: В регистре `LTDC_BPCR` пишут ширину **HSYNC+HBP-1** и ширину **VSYNC+VBP-1**.
- Активная Ширина и Высота: В регистре `LTDC_AWCR` пишут суммарное значение **HSYNC+ HBP +Активная_Ширина-1** и суммарное значение **VSYNC +VBP+Активная_Высота-1** (поддерживается до 1024x768).
- Общая ширина: В регистре `LTDC_TWCR` пишут суммарное значение **HSYNC+ HBP+ Активная_Ширина+HFP-1**.
- Общая высота: В регистре `LTDC_TWCR` пишут суммарное значение **VSYNC+VBP +Активная_Высота+VFP-1**.

У включённого LTDC времянка генерируется начиная с позиции X/Y=0/0 как первый пиксель горизонтальной синхронизации в области вертикальной синхронизации с последующими задним крыльцом, активной областью и передним крыльцом.

У выключенного LTDC блок генератора времянки сброшен в состоянии X=Общая_Ширина - 1, Y=Общая_Высота - 1 и держит последний пиксель перед фазой вертикальной синхронизации и слитый FIFO. Так выдаются только данные затемнения.

Пример времянки

Времянка TFT-LCD (из описания панели):

- Ширина Горизонтальной и Вертикальной синхронизации: 0x8 пикселей и 0x4 строк
- Горизонтальное в Вертикальное заднее крыльцо: 0x7 пикселей и 0x2 строк
- Активные Ширина и Высота: 0x280 пикселей, 0x1E0 строк (640x480)
- Горизонтальное переднее крыльцо: 0x6 пикселей
- Вертикальное переднее крыльцо: 0x2 строк

Регистры времянки LTDC:

- Регистр LTDC_SSCR: пишут 0x00070003. (HSW[11:0] — 0x7 и VSH[10:0] — 0x3)
- Регистр LTDC_BPCR: пишут 0x000E0005. (AHBP[11:0] — 0xE(0x8 + 0x6) и AVBP[10:0] — 0x5(0x4 + 0x1))
- Регистр LTDC_AWCR: пишут 0x028E01E5. (AAW[11:0] — 0x28E(0x8 +0x7 +0x27F) и AAH[10:0] — 0x1E5(0x4 +0x2 + 0x1DF)
- Регистр LTDC_TWCR: пишут 0x00000294. (TOTALW[11:0] — 0x294(0x8 +0x7 +0x280 + 0x5)
- Регистр LTDC_THCR: пишут 0x000001E7. (TOTALH[10:0] — 0x1E7(0x4 +0x2 + 0x1E0 + 1)

Программируемая полярность

Полярность Горизонтальной и Вертикальной синхронизации, Разрешения данных и Тактов пикселей задают в регистре LTDC_GCR.

Цвет фона

Постоянный цвет фона (RGB888) пишут в регистр LTDC_BCCR. Он используется при смешивании с нижним слоем.

Размывание

Техника псевдослучайного размывания с LFSR используется как добавление малых случайных чисел (порог) к каждому каналу цвета пикселей (R, G или B), как бы округляя старшие биты при выводе 24-бит данных на 18-бит дисплей. Размывание применяется к пикселям, различным в разных кадрах.

Эта техника эквивалентна сравнению младших битов с порогом и добавлении 1 к старшим битам если младшая часть больше или равна порогу. Обычно младшие биты теряются.

Ширина добавляемых псевдослучайных чисел равна по 2 бита на каждый канал.

LFSR включается при включении контроллера LCD-TFT, постоянно работает и выключается вместе с ним.

Включать и выключать размывание можно на лету в регистре LTDC_GCR.

Перезагрузка теневых регистров

Некоторые регистры конфигурации являются теневыми и имеют активную копию. В регистре LTDC_SRCR можно определить порядок записи из теневых регистров в активные, сразу или во время обратного хода кадровой развёртки. При немедленной перезаписи её надо активировать только после записи всех новых регистров.

Теневые регистры не должно модифицировать до их перезаписи. Чтение из теневого регистра возвращает значение активной копии, так что записанное значение можно прочесть только после перезаписи.

Прерывание перезаписи регистров разрешается в регистре LTDC_IER. Теневыми регистрами являются все регистры слоёв 1 и 2, кроме регистра LTDC_LxCLUTWR.

Выдача прерываний

См. Секцию 16.5.

16.4.2. Программируемые параметры слоя

Есть два слоя, управляемых раздельно. Порядок слоёв фиксирован, начиная с нижнего. Если включены оба слоя, то Слой 2 в выводимом окне отображается сверху.

Окна

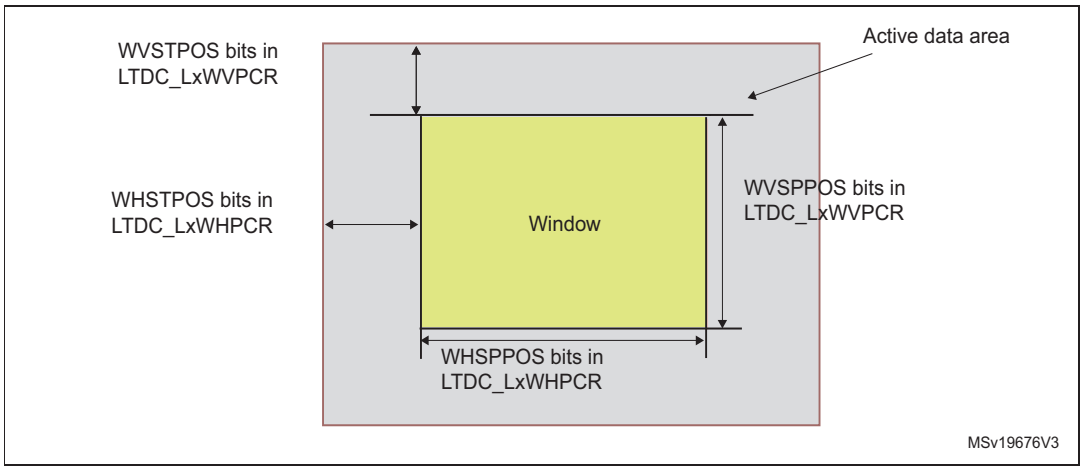
Слои могут быть разного размера и размещаться внутри Активной области отображения.

Позиция и размер окна задаются с помощью координат верхнего левого и нижнего правого углов и внутреннего генератора времянки, включающего синхронность, размер заднего крыльца и активную область данных. См. регистры LTDC_LxWHPCR и LTDC_WVPCR.

Программируемые позиция и размер слоя определяют первый/последний видимый пиксель строки и первую/последнюю видимую строку в окне. Это позволяет выводить весь кадр или его часть.

- Первый и последний видимый пиксели слоя задают в полях WHSTPOS[11:0] и WHSPPOS[11:0] регистра LTDC_LxWHPCR.
- Первую и последнюю видимые строки слоя задают в полях WVSTPOS[10:0] и WVSPPOS[10:0] регистра LTDC_LxWVPCR.

Рис. 83. Программируемые параметры слоя



Входной формат пикселя

В кадровом буфере слоя биты хранятся во входном формате. Они определяются в регистре LTDC_LxPFCR для каждого слоя.

Пиксель из буфера кадра преобразуются во внутренний формат 8888 (ARGB):

- Компоненты короче 8 расширяются до 8 репликацией. Выбранный диапазон битов сливается несколько раз до получения больше 8 битов. В полученном векторе оставляют старшие 8.

Пример: 5 битов красного канала в RGB565 становятся (позиции битов): 43210432 (три 3 младших бита заполняются старшими битами из 5).

Таблица 90. Размещение данных пикселей и формат цвета

ARGB8888			
@+3 A_X[7:0]	@+2 R_X[7:0]	@+1 G_X[7:0]	@ B_X[7:0]
@+7 A_{X+1}[7:0]	@+6 R_{X+1}[7:0]	@+5 G_{X+1}[7:0]	@+4 B_{X+1}[7:0]
RGB888			
@+3 B_{X+1}[7:0]	@+2 R_X[7:0]	@+1 G_X[7:0]	@ B_X[7:0]
@+7 G_{X+2}[7:0]	@+6 B_{X+2}[7:0]	@+5 R_{X+1}[7:0]	@+4 G_{X+1}[7:0]
RGB565			
@+3 R_{X+1}[4:0] G_{X+1}[5:3]	@+2 G_{X+1}[2:0] B_{X+1}[4:0]	@+1 R_X[4:0] G_X[5:3]	@ G_X[2:0] B_X[4:0]
ARGB8888			
@+7 R_{X+3}[4:0] G_{X+3}[5:3]	@+6 G_{X+3}[2:0] B_{X+3}[4:0]	@+5 R_{X+2}[4:0] G_{X+2}[5:3]	@+4 G_{X+2}[2:0] B_{X+2}[4:0]

ARGB1555			
@+3 $A_{x+1}[0]R_{x+1}[4:0]$ $G_{x+1}[4:3]$	@+2 $G_{x+1}[2:0]$ $B_{x+1}[4:0]$	@+1 $A_x[0]$ $R_x[4:0]$ $G_x[4:3]$	@ $G_x[2:0]$ $B_x[4:0]$
@+7 $A_{x+3}[0]R_{x+3}[4:0]$ $G_{x+3}[4:3]$	@+6 $G_{x+3}[2:0]$ $B_{x+3}[4:0]$	@+5 $A_{x+2}[0]R_{x+2}[4:0]G_{x+2}[4:3]$	@+4 $G_{x+2}[2:0]$ $B_{x+2}[4:0]$
ARGB4444			
@+3 $A_{x+1}[3:0]R_{x+1}[3:0]$	@+2 $G_{x+1}[3:0]$ $B_{x+1}[3:0]$	@+1 $A_x[3:0]$ $R_x[3:0]$	@ $G_x[3:0]$ $B_x[3:0]$
@+7 $A_{x+3}[3:0]R_{x+3}[3:0]$	@+6 $G_{x+3}[3:0]$ $B_{x+3}[3:0]$	@+5 $A_{x+2}[3:0]R_{x+2}[3:0]$	@+4 $G_{x+2}[3:0]$ $B_{x+2}[3:0]$
L8			
@+3 $L_{x+3}[7:0]$	@+2 $L_{x+2}[7:0]$	@+1 $L_{x+1}[7:0]$	@ $L_x[7:0]$
@+7 $L_{x+7}[7:0]$	@+6 $L_{x+6}[7:0]$	@+5 $L_{x+5}[7:0]$	@+4 $L_{x+4}[7:0]$
AL44			
@+3 $A_{x+3}[3:0]$ $L_{x+3}[3:0]$	@+2 $A_{x+2}[3:0]$ $L_{x+2}[3:0]$	@+1 $A_{x+1}[3:0]$ $L_{x+1}[3:0]$	@ $A_x[3:0]$ $L_x[3:0]$
@+7 $A_{x+7}[3:0]$ $L_{x+7}[3:0]$	@+6 $A_{x+6}[3:0]$ $L_{x+6}[3:0]$	@+5 $A_{x+5}[3:0]$ $L_{x+5}[3:0]$	@+4 $A_{x+4}[3:0]$ $L_{x+4}[3:0]$
AL88			
@+3 $A_{x+1}[7:0]$	@+2 $L_{x+1}[7:0]$	@+1 $A_x[7:0]$	@ $L_x[7:0]$
@+7 $A_{x+3}[7:0]$	@+6 $L_{x+3}[7:0]$	@+5 $A_{x+2}[7:0]$	@+4 $L_{x+2}[7:0]$

Таблица выбора цвета (CLUT)

CLUT полезна только в случае индексированных цветов в форматах L8, AL44 и AL88. Её можно подключить на лету регистром LTDC_LxCR.

Сначала CLUT заполняют значениями R, G и B, которые заменят исходные значения пикселя R, G, B (индексируемый цвет). Каждый цвет (значение RGB) имеет свой адрес в CLUT.

Значения R, G и B и их соответствующий адрес пишут через регистр LTDC_LxCLUTWR.

- При форматах L8 и AL88 в CLUT пишут 256 цветов. Адрес цвета ставят в битах CLUTADD регистра LTDC_LxCLUTWR.
- При формате AL44 в CLUT пишут 16 цветов. Адрес цвета это репликация 4-бит канала L до 8:
  - L0 (индексируемый цвет 0), по адресу 0x00
  - L1, по адресу 0x11
  - L2, по адресу 0x22
  - .....
  - L15, по адресу 0xFF

Адрес буфера цветов кадра

Адрес буфера цветов кадра каждого слоя пишут в регистр LTDC_LxCFBAR. У включённого слоя данные извлекаются из буфера цветов кадра.

Длина буфера цветов кадра

Для каждого слоя общая длина строки в буфере цветов кадра и число строк задаются в регистрах LTDC_LxCFBLR и LTDC_LxCFBLNR. Они используются для остановки выборки данных в FIFO слоя по концу кадра.

- Если они меньше нужного, то выдаётся прерывание исчерпания FIFO.
- Если они больше нужного, то бесполезные данные из FIFO выбрасываются и не выводятся.

Заполнение буфера цветов кадра

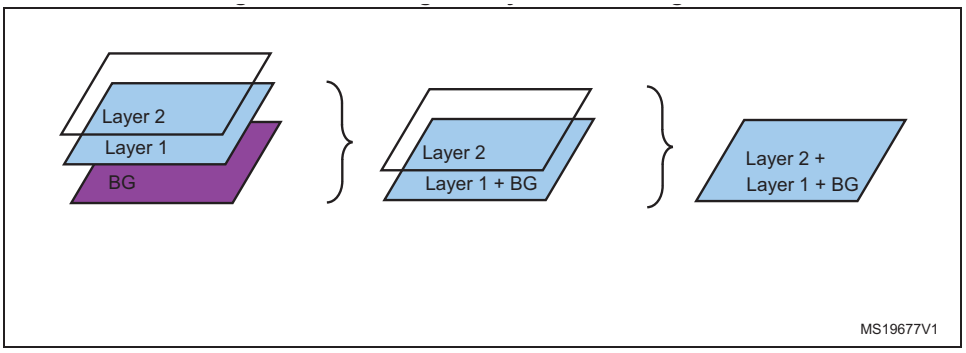
У каждого слоя есть конфигурируемое наполнение буфера цветов кадра. Это дистанция между началами соседних строк в байтах. Его пишут в регистр LTDC_LxCFBLR.

Смешивание слоёв

Смешивание всегда включено, параметры задаются в регистре LTDC_LxBFCR.

Порядок смешивания фиксирован, снизу вверх. При двух включённых слоях сначала слой 1 смешивается с фоновым, затем и слой 2.

Рис. 84. Смешивание двух слоёв с фоном.



Цвет слоя по умолчанию

Он используется для вывода вне заданного окна и при выключенном слое. Формат ARGB. Задаётся в регистре LTDC_LxDCCR. Смешивание выполняется даже с выключенным слоем, так что сбросьте его для выключенного слоя в регистре LTDC_LxBFCR.

Ключ прозрачного цвета

Если эта опция включена, то результирующий цвет пикселя сравнивается с ключом, и при совпадении все каналы пикселя (ARGB) обнуляются. Использование ключа можно менять на лету в регистре LTDC_LxCKCR.

16.5. Прерывания LTDC

Четыре маскируемых прерывания LTDC объединены по OR в два вектора.

Разрешаются источники прерываний раздельно в регистре LTDC_IER.

События прерываний:

- Прерывание строки: выдаётся при достижении строки, заданной в регистре LTDC_LIPCR
- Перезапись теневых регистров: выдаётся после перезаписи во время кадрового гашения
- Исчерпание FIFO: выдаётся при запросе пикселя из пустого FIFO слоя
- Ошибка передачи шины AHB.

Рис. 85. События прерываний

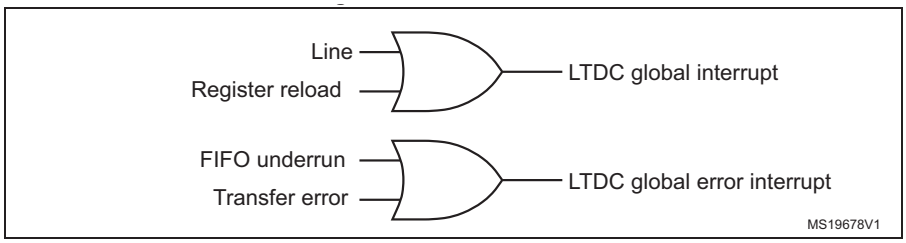


Таблица 91. Запросы прерываний

Событие	Флаг	Бит разрешения
Строка	LIF	LIE
Перезапись регистров	RRIF	RRIEN
Исчерпание FIFO	FUDERRIF	FUDERRIE
Ошибка передачи	TERRIF	TERRIE

16.6. Процедура программирования LTDC

- Включаем такты LTDC в регистре RCC
- Задаём такты пикселей согласно описания панели
- Ставим синхронную времянку: VSYNC, HSYNC, Вертикальное и Горизонтальное крыльцо, активную область данных для панели. См. Секцию 16.4.1
- Задаём полярность сигналов синхронизации и тактов в регистре LTDC_GCR
- Если надо, пишем цвет фона в регистр LTDC_BCCR
- Определяем нужные прерывания в регистрах LTDC_IER и LTDC_LIPCR

- Ставим параметры Слоёв 1/2:
  - Горизонтальное и вертикальное размещение окна слоя в регистрах LTDC_LxWHPCR и LTDC_WVPCR. Оно должно быть в активной области.
  - Входной формат пикселей в регистре LTDC_LxPFCR
  - Адрес начала буфера цветов кадра в регистре LTDC_LxCFBAR
  - Длину строки и заполнение буфера цветов кадра в регистре LTDC_LxCFBLR
  - Число строк буфера цветов кадра в регистре LTDC_LxCFBLNR
  - Если надо, грузим CLUT с значениями RGB и её адрес в регистр LTDC_LxCLUTWR
  - Если надо, ставим цвет по умолчанию и смешивание в регистры LTDC_LxDCCR и LTDC_LxBFCR
- Включаем слои 1/2 и, если надо, CLUT в регистре LTDC_LxCR
- Если надо, размывание и ключ цвета включают в регистрах LTDC_GCR и LTDC_LxCKCR, можно и на лету.
- Через регистр LTDC_SRCR переписываем теневые регистры в активные.
- Включаем контроллер LCD-TFT в регистре LTDC_GCR.
- Все параметры слоёв, кроме CLUT, можно менять на лету. Новая конфигурация загрузится немедленно или при вертикальном гашении, в зависимости от регистра LTDC_SRCR.

Все регистры слоёв теневые. После записи их нельзя менять до перезаписи в активные регистры.

16.7. Регистры LTDC

16.7.1. Регистр размера синхронизации (LTDC_SSCR)

Смещение адреса: 0x08

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				HSW[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				VSH[10:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:28 Резерв, не трогать.
— Биты 27:16 **HSW[11:0]**: Размер горизонтальной синхронизации (в тактах пикселей минус 1)
— Биты 15:11 Резерв, не трогать.
— Биты 10:0 **VSH[10:0]**: Размер вертикальной синхронизации (в строках минус 1)

16.7.2. Регистр размера заднего и переднего крыльца (LTDC_BPCR)

Смещение адреса: 0x0C

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				AHBP[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				AVBP[10:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Содержит суммарные значения (ширина HSYNC + HBP- 1) и (высота VSYNC + VBP - 1). См. Секцию 16.4.

— Биты 31:28 Резерв, не трогать.
— Биты 27:16 **AHBP[11:0]**: Размер заднего крыльца (в тактах пикселей)
— Биты 15:11 Резерв, не трогать.
— Биты 10:0 **AVBP[10:0]**: Размер переднего (в строках)

16.7.3. Регистр размера активной области (LTDC_BPCR)

Смещение адреса: 0x10

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				AHBP[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				AVBP[10:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Содержит суммарные значения (ширина HSYNC + HBP + Активная_ширина - 1) и (высота VSYNC + BVBP + Активная_высота - 1). См. Секцию 16.4.

— Биты 31:28 Резерв, не трогать.

— Биты 27:16 **AHBP[11:0]**: Активная_ширина (в тактах пикселей)

— Биты 15:11 Резерв, не трогать.

— Биты 10:0 **AVBP[10:0]**: Активная_высота (в строках)

16.7.4. Регистр общей ширины (LTDC_TWCR)

Смещение адреса: 0x14

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				TOTALW[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

16	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				TOTALH[10:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Содержит суммарные значения (ширина HSYNC + HBP + Активная_ширина + HFP - 1) и (высота VSYNC + BVBP + Активная_высота + VFP - 1). См. Секцию 16.4.

— Биты 31:28 Резерв, не трогать.

— Биты 27:16 **TOTALW[11:0]**: Общая_ширина (в тактах пикселей)

— Биты 15:11 Резерв, не трогать.

— Биты 10:0 **TOTALH[10:0]**: Общая_высота (в строках)

16.7.5. Глобальный регистр управления (LTDC_GCR)

Смещение адреса: 0x18

По сбросу: 0x0000 2220

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HSPOL	VSPOL	DEPOL	PCPOL	Reserved											DEN
rw	rw	rw	rw												rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	DRW[2:0]			Reserved	DGW[2:0]			Reserved	DBW[2:0]			Reserved			LTDCEN
	r	r	r		r	r	r		r	r	r				rw

— Бит 31 **HSPOL**: Полярность горизонтальной синхронизации
Читают и пишут программно.
0: Активна низким
1: Активна низким

— Бит 30 **VSPOL**: Полярность вертикальной синхронизации
Читают и пишут программно.
0: Активна низким
1: Активна низким

— Бит 29 **DEPOL**: Полярность разрешения данных
Читают и пишут программно.

0: Активно низким
1: Активно низким

— Бит 28 **PCPOL**: Полярность тактов пикселей
Читают и пишут программно.
0: Активны низким
1: Активны низким

— Биты 27:17 Резерв, не трогать.

— Бит 16 **DEN**: Разрешение размывания
Читают и пишут программно.
0: Нельзя
1: Можно

— Бит 15 Резерв, не трогать.

— Биты 14:12 **DRW[2:0]**: Ширина размывания Красного

— Бит 11 Резерв, не трогать.

— Биты 10:8 **DGW[2:0]**: Ширина размывания Зелёного

— Бит 7 Резерв, не трогать.

— Биты 6:4 **DBW[2:0]**: Ширина размывания Синего

— Биты 3:1 Резерв, не трогать.

— Бит 0 **LTDCEN**: Включение контроллера LCD-TFT
Читают и пишут программно.
0: Выкл.
1: Вкл.

16.7.6. Регистр режима перезаписи теневых регистров (LTDC_SRCR)

Смещение адреса: 0x24

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														VBR	IMR
														rw	rw

— Биты 31:2 Резерв, не трогать.

— Бит 1 **VBR**: Перезапись при вертикальном гашении
Читают программно, снимается аппаратно после перезаписи.
0: Ничего
1: Перезапись при вертикальном гашении (в начале первой строки после активной области)

— Бит 0 **VBR**: Немедленная перезапись
Читают программно, снимается аппаратно после перезаписи.
0: Ничего
1: Немедленная перезапись

16.7.7. Регистр фонового цвета (LTDC_BCCR)

Смещение адреса: 0x2C

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								BCRED[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BCGREEN[7:0]								BCBLUE[7:0]							
rw								rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:24 Резерв, не трогать.

— Биты 23:16 **BCRED[7:0]**: Красный

— Биты 23:16 **BCGREEN[7:0]**: Зелёный
— Биты 23:16 **BCBLUE[7:0]**: Синий

16.7.8. Регистр разрешения прерываний (LTDC_IER)

Смещение адреса: 0x34

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												RRIE	TERRIE	FUIE	LIE
												rw	rw	rw	rw

Запись 1 разрешает прерывание.

— Биты 31:4 Резерв, не трогать.
— Бит 3 **RRIE**: Перезапись регистров
— Бит 2 **TERRIE**: Ошибка передачи
— Бит 1 **FUIE**: Исчерпание FIFO
— Бит 0 **LIE**: Конец строки

16.7.9. Регистр флагов прерываний (LTDC_ISR)

Смещение адреса: 0x38

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												RRIF	TERRIF	FUIF	LIF
												r	r	r	r

При наличии запроса выдаёт 1.

— Биты 31:4 Резерв, не трогать.
— Бит 3 **RRIF**: Перезапись регистров
— Бит 2 **TERRIF**: Ошибка передачи
— Бит 1 **FUIF**: Исчерпание FIFO
— Бит 0 **LIF**: Конец строки

16.7.10.Регистр очистки флагов прерываний (LTDC_ICR)

Смещение адреса: 0x3C

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												CRRIF	CTERRIF	CFUIF	CLIF
												w	w	w	w

При наличии запроса выдаёт 1.

— Биты 31:4 Резерв, не трогать.
— Бит 3 **CRRIF**: Перезапись регистров
— Бит 2 **CTERRIF**: Ошибка передачи
— Бит 1 **CFUIF**: Исчерпание FIFO
— Бит 0 **CLIF**: Конец строки

16.7.11.Регистр номера строки прерывания (LTDC_LIPCR)

Смещение адреса: 0x40

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					LIPOS[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Номер строки зависит от параметров времянки.

— Биты 31:11 Резерв, не трогать.

— Биты 10:0 **LIPOS[10:0]**: Номер строки

16.7.12.Регистр текущей позиции (LTDC_CPSR)

Смещение адреса: 0x44

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CXPOS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CYPOS{15;0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

— Биты 31:16 **CXPOS[15:0]**: Текущая позиция X

— Биты 15:0 **CYPOS[15:0]**: Текущая позиция Y

16.7.13.Регистр текущего состояния отображения (LTDC_CDSR)

Смещение адреса: 0x48

По сбросу: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												HSYNCS	VSYNCS	HDES	VDES
												r	r	r	r

Для всех битов, независимо от полярности:

0: Неактивно

1: Активно

— Биты 31:4 Резерв, не трогать.

— Бит 3 **HSYNCS**: Горизонтальная синхронизация

— Бит 2 **VSYNCS**: Вертикальная синхронизация

— Бит 1 **HDES**: Горизонтальное разрешение данных

— Бит 0 **VDES**: Вертикальное разрешение данных

16.7.14.Регистр управления слоя x (LTDC_LxCR) (x=1..2)

Смещение адреса: 0x84 + 0x80 x (Layerx -1), Layerx = 1 или 2

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											CLUTEN	Reserved		COLKEN	LEN
											rw			rw	rw

Ставится и снимается программно.

0: Нельзя

1: Можно

— Биты 31:5 Резерв, не трогать.

— Бит 4 **CLUTEN**: Разрешение CLUT

— Биты 3:2 Резерв, не трогать.

— Бит 1 **COLKEN**: Разрешение ключа цвета

— Бит 0 **LEN**: Разрешение слоя

16.7.15.Регистр горизонтальной позиции окна слоя x (LTDC_LxWHPCR) (x=1..2)

Смещение адреса: 0x88 + 0x80 x (Layerx -1), Layerx = 1 или 2

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				WHSPPOS[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				WHSTPOS[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Первый видимый бит строки это запись в битах AHBP[10:0] + 1 в регистре LTDC_BPCR.

Последний видимый бит строки это запись в битах AAW[10:0] в регистре LTDC_AWCR.

— Биты 31:28 Резерв, не трогать.

— Биты 27:16 **WHSPPOS[11:0]**: Последний видимый бит окна слоя

Должен быть:

WHSPPOS[11:0] ≥ AHBP[10:0] + 1 (в регистре LTDC_BPCR)

— Биты 15:12 Резерв, не трогать.

— Биты 11:0 **WHSTPOS[11:0]**: Первый видимый бит окна слоя

Должен быть:

WHSTPOS[11:0] ≤ AAW[10:0] (в регистре LTDC_AWCR).

16.7.16.Регистр вертикальной позиции окна слоя x (LTDC_LxWVPCR) (x=1..2)

Смещение адреса: 0x8C + 0x80 x (Layerx -1), Layerx = 1 или 2

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					WVSPPOS[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					WVSTPOS[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Первая видимая строка это запись в битах AVBP[10:0] + 1 в регистре LTDC_BPCR.

Последняя видимая строка это запись в битах AAH[10:0] в регистре LTDC_AWCR.

— Биты 31:28 Резерв, не трогать.

— Биты 27:16 **WVSPPOS[11:0]**: Последняя видимая строка окна слоя

Должен быть:

WVSPPOS[11:0] ≥ AVBP[10:0] + 1 (в регистре LTDC_BPCR)

— Биты 15:12 Резерв, не трогать.

— Биты 11:0 **WVSTPOS[11:0]**: Первая видимая строка окна слоя
Должен быть:
WVSTPOS[11:0] ≤ AAH[10:0] (в регистре LTDC_AWCR).

16.7.17.Регистр ключа цвета слоя x (LTDC_LxCKCR) (x=1..2)

Смещение адреса: 0x90 + 0x80 x (Layerx -1), Layerx = 1 или 2
По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								CKRED[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CKGREEN[7:0]								CKBLUE[7:0]							
rw								rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:24 Резерв, не трогать.
— Биты 23:16 **CKRED[7:0]**: Ключ Красного
— Биты 15:8 **CKGREEN[7:0]**: Ключ Зелёного
— Биты 7:0 **CKBLUE[7:0]**: Ключ Синего

16.7.18.Регистр формата пикселя слоя x (LTDC_LxPFCR) (x=1..2)

Смещение адреса: 0x94 + 0x80 x (Layerx -1), Layerx = 1 или 2
По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													PF[2:0]		
													rw	rw	rw

— Биты 31:3 Резерв, не трогать.
— Биты 2:0 **PF[2:0]**: Формат пикселя
000: ARGB8888
001: RGB888
010: RGB565
011: ARGB1555
100: ARGB4444
101: L8 (8-Bit Luminance)
110: AL44 (4-Bit Alpha, 4-Bit Luminance)
111: AL88 (8-Bit Alpha, 8-Bit Luminance)

16.7.19.Регистр постоянного альфа слоя x (LTDC_LxCACR) (x=1..2)

Смещение адреса: 0x98 + 0x80 x (Layerx -1), Layerx = 1 или 2
По сбросу: 0x0000 00FF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CONSTA[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:8 Резерв, не трогать.
— Биты 7:0 **CONSTA[7:0]**: Постоянное значение альфа для смешивания
Аппаратно делится на 255.

16.7.20.Регистр цвета по умолчанию слоя x (LTDC_LxDCCR) (x=1..2)

Смещение адреса: 0x9C + 0x80 x (Layerx -1), Layerx = 1 или 2

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DCALPHA[7:0]								DCRED[7:0]							
rw								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DCGREEN[7:0]								DCBLUE[7:0]							
rw								rw	rw	rw	rw	rw	rw	rw	rw

Задаёт цвет по умолчанию в формате ARGB, используемый при выключенном слое или вне окна слоя. Значение 0x00000000 это прозрачный цвет.

— Биты 31:24 **DCALPHA[7:0]**: Альфа
— Биты 23:16 **DCRED[7:0]**: Красный
— Биты 15:8 **DCGREEN[7:0]**: Зелёный
— Биты 7:0 **DCBLUE[7:0]**: Синий

16.7.21.Регистр множителей смешивания слоя x (LTDC_LxBFCR) (x=1..2)

Смещение адреса: 0xA0 + 0x80 x (Layerx -1), Layerx = 1 или 2

По сбросу: 0x0000 0607

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					BF1[2:0]			Reserved					BF2[2:0]		
					rw	rw	rw						rw	rw	rw

Задаёт множители смешивания F1 и F2.

Общая формула смешивания: BC = BF1 x C + BF2 x Cs

- BC = Смешанный цвет
- BF1 = Множитель 1
- C = Цвет текущего слоя
- BF2 = Множитель 2
- Cs = Смешанный цвет нижележащего слоя.

— Биты 31:11 Резерв, не трогать.
— Биты 10:8 **BF1[2:0]**: Множитель 1
000: Резерв
001: Резерв
010: Резерв
011: Резерв
100: Постоянная Альфа (при вычислениях аппаратно делится на 255)
101: Резерв
110: Альфа пикселя x Постоянная Альфа
111: Резерв
— Биты 7:3 Резерв, не трогать.
— Биты 2:0 **BF2[2:0]**: Множитель 2
000: Резерв
001: Резерв
010: Резерв
011: Резерв
100: 1 - Постоянная Альфа
101: Резерв
110: 1 - (Альфа пикселя x Постоянная Альфа)
111: Резерв

16.7.22.Регистр адреса буфера цвета кадра слоя x (LTDC_LxBFCR) (x=1..2)

Смещение адреса: 0xAC + 0x80 x (Layerx -1), Layerx = 1 или 2

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CFBADD[31:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CFBADD[31:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:0 **CFBADDR[31:0]**: Адрес

16.7.23.Регистр длины буфера цвета кадра слоя x (LTDC_LxCFBLR) (x=1..2)

Смещение адреса: 0xB0 + 0x80 x (Layerx -1), Layerx = 1 или 2

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			CFBP[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CFBLL[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:29 Резерв, не трогать.

— Биты 28:16 **CFBP[12:0]**: Заполнение (расстояние между началами соседних строк в байтах)

— Биты 15:13 Резерв, не трогать.

— Биты 12:0 **CFBLL[12:0]**: Длина строк в байтах (*наибольшая ширина* x *байтов/пиксель* - 3)

16.7.24.Регистр числа срок буфера цвета кадра слоя x (LTDC_LxCFBLNR) (x=1..2)

Смещение адреса: 0xB4 + 0x80 x (Layerx -1), Layerx = 1 или 2

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					CFBLNBR[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:11 Резерв, не трогать.

— Биты 10:0 **CFBLNBR[12:0]**: Число строк

Число строк и длина строки задают число извлекаемых данных кадра каждого слоя. Если задано меньше нужного, то выдаётся прерывание исчерпания FIFO.

16.7.25.Регистр записи CLUT слоя x (LTDC_LxCLUTWR) (x=1..2)

Смещение адреса: 0xC4 + 0x80 x (Layerx -1), Layerx = 1 или 2

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CLUTADD[7:0]								RED[7:0]							
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GREEN[7:0]								BLUE[7:0]							
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

— Биты 31:24 **CLUTADD[7:0]**: Позиция цвета в для каждого значения RGB.
— Биты 23:16 **RED[7:0]**: Красный.
— Биты 15:8 **GREEN[7:0]**: Зелёный.
— Биты 7:0 **BLUE[7:0]**: Синий.

В этот регистр можно писать только во время гашения или при выключенном слое (регистр LTDC_LxCR). CLUT имеет смысл только с форматами L8, AL44 и AL88.

16.7.26.Карта регистров LTDC

Таблица 92. Карта регистров LTDC

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0008	LTDC_SSCR	Reserved						HSW[9:0]										Reserved					VSH[10:0]										
	Reset value							0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x000C	LTDC_BPCR	Reserved				AHBP[11:0]												Reserved					AVBP[10:0]										
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x0010	LTDC_AWCR	Reserved				AAV[11:0]												Reserved					AAH[10:0]										
	Reset value					0	9	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x0014	LTDC_TWCR	Reserved				TOTALW[11:0]												Reserved					TOTALH[10:0]										
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x0018	LTDC_GCR	HSPOL	VSPOL	DEPOL	PCPOL	Reserve											DEN	Reserved	DRW[2:0]			Reserved	DGW[2:0]			Reserved	DBW[2:0]			Reserved				LTDCEN
	Reset value	0	0	0	0												0		0	1	0		0	1	0		0	1	0				0
0x0024	LTDC_SRCR	Reserved																														VBR	IMR
	Reset value																															0	0
0x002C	LTDC_BCCR	Reserved						BC[23:0]																									
	Reset value							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0034	LTDC_IER	Reserved																												RRIE	TERRIE	FUIE	LIE
	Reset value																													0	0	0	0
0x0038	LTDC_ISR	Reserved																												RRIF	TERRIF	FUIF	LIF
	Reset value																													0	0	0	0
0x003C	LTDC_ICR	Reserved																												CRRIF	CTERRIF	CFUIF	CLIF
	Reset value																													0	0	0	0
0x0040	LTDC_LIPCR	Reserved																					LIPOS[10:0]										
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x0044	LTDC_CPSR	CXPOS[15:0]																CYPOS[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0048	LTDC_CDSR	Reserved																												HSYNCS	VSYNCS	HDES	VDES
	Reset value																													1	1	1	1
0x0084	LTDC_L1CR	Reserved																										CLUTEN	Reserved			COLKEN	LEN
	Reset value																											0				0	0
0x0088	LTDC_L1WHPCR	Reserved				WHSPPOS[11:0]												Reserved					WHSTPOS[11:0]										
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x008C	LTDC_L1WVPCR	Reserved					WVSPPOS[10:0]											Reserved					WVSTPOS[10:0]										
	Reset value						0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x0090	LTDC_L1CKCR	Reserved								CKRED[7:0]								CKGREEN[7:0]								CKBLUE[7:0]							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0094	LTDC_L1PFCR	Reserved																														PF[2:0]	
	Reset value																															0	0	0
0x0098	LTDC_L1CACR	Reserved																								CONSTA[7:0]							
	Reset value																									1	1	1	1	1	1	1	1
0x009C	LTDC_L1DCCR	DCALPHA[7:0]								DCRED[7:0]								DCGREEN[7:0]								DCBLUE[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00A0	LTDC_L1BFCR	Reserved																					BF1[2:0]			Reserved						BF2[2:0]	
	Reset value																						1	1	0							1	1	1
0x00AC	LTDC_L1CFBAR	CFBADD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00B0	LTDC_L1CFBLR	Reserved			CFBP[12:0]													Reserved			CFBLL[12:0]												
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00B4	LTDC_L1CFBLNR	Reserved																					CFBLNBR[10:0]										
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x00C4	LTDC_L1CLUTWR	CLUTADD[7:0]								RED[7:0]								GREEN[7:0]								BLUE[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0104	LTDC_L2CR	Reserved																											CLUTEN	Reserved		COLKEN	LEN
	Reset value																												0			0	0
0x0108	LTDC_L2WHPCR	Reserved			WHSPPOS[11:0]													Reserved			WHSTPOS[11:0]												
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010C	LTDC_L2WVPCR	Reserved					WVSPPOS[10:0]											Reserved					WVSTPOS[10:0]										
	Reset value						0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x0110	LTDC_L2CKCR	Reserved								CKRED[7:0]								CKGREEN[7:0]								CKBLUE[7:0]							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0114	LTDC_L2PFCR	Reserved																														PF[2:0]	
	Reset value																															0	0	0
0x0118	LTDC_L2CACR	Reserved																								CONSTA[7:0]							
	Reset value																									1	1	1	1	1	1	1	1
0x011C	LTDC_L2DCCR	DCALPHA[7:0]								DCRED[7:0]								DCGREEN[7:0]								DCBLUE[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0120	LTDC_L2BFCR	Reserved																					BF1[2:0]			Reserved						BF2[2:0]	
	Reset value																						1	1	0							1	1	1
0x012C	LTDC_L2CFBAR	CFBADD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0130	LTDC_L2CFBLR	Reserved			CFBP[12:0]													Reserved			CFBLL[12:0]												
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0134	LTDC_L2CFBLNR	Reserved																					CFBLNBR[10:0]										
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x0144	LTDC_L2CLUTWR	CLUTADD[7:0]								RED[7:0]								GREEN[7:0]								BLUE[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

17. Улучшенные таймеры (TIM1 и TIM8)

17.1. Введение в TIM1 и TIM8

Это 16-бит само-перегружаемые таймеры с программируемым предделителем.

Они могут использоваться для измерения длительности входных импульсов (захват входа), генерации сигналов (сравнение выхода, ШИМ, инверсный ШИМ) и пр.

Длина импульсов и сигналов может модулироваться от нескольких микросекунд до миллисекунд с помощью предделителей таймера и тактов RCC.

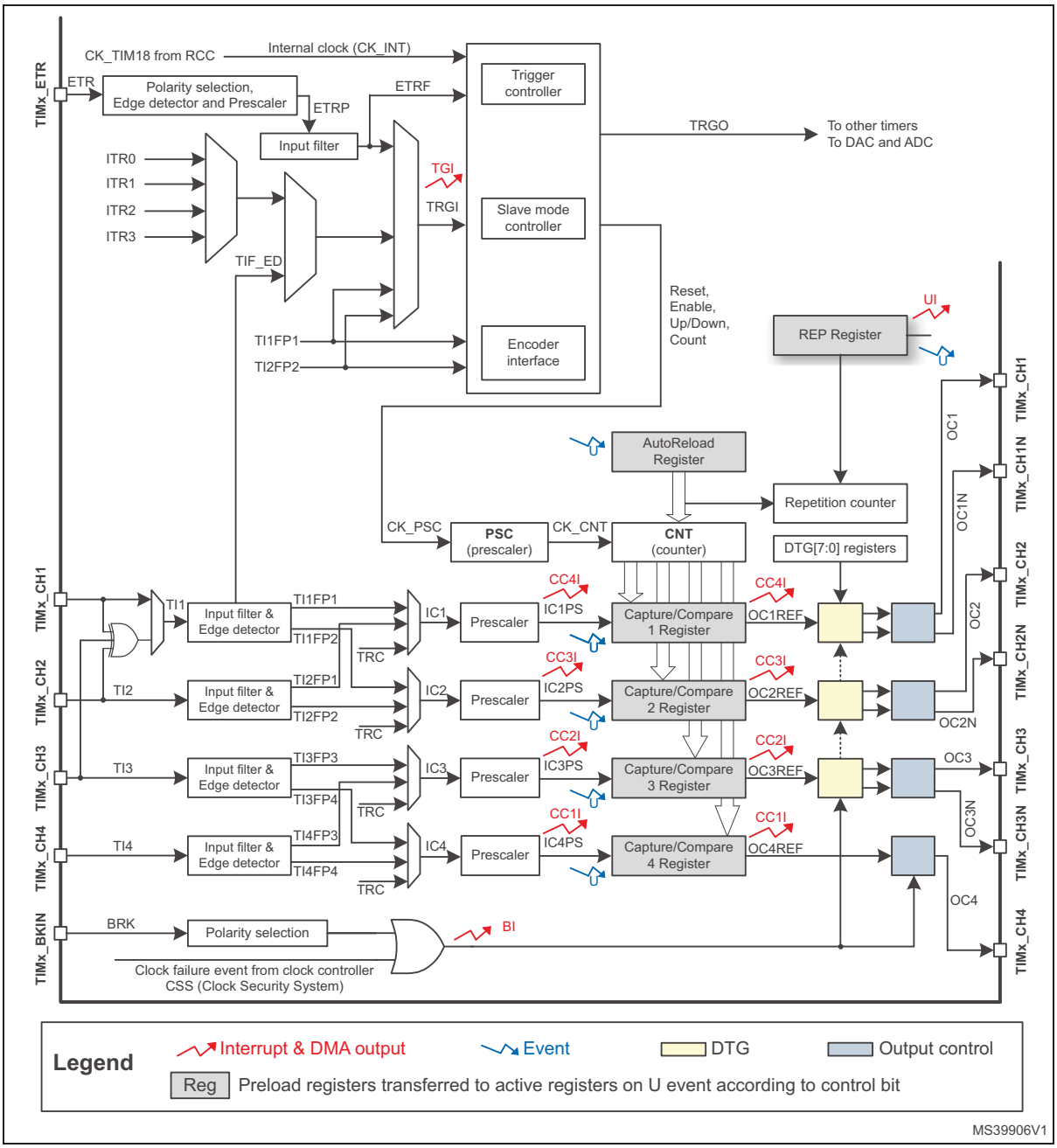
Улучшенные (TIM1 and TIM8) и обычные (TIMx) таймеры полностью независимы и не имеют общих ресурсов. Могут синхронизироваться между собой. См. *Секцию 17.3.20*.

17.2. Основные свойства TIM1 и TIM8

Это:

- 16-бит прямой, обратный, реверсивный счёт с само-перезагрузкой.
- 16-бит программируемый предделитель (можно “налету”) тактов на число от 1 до 65536.
- До 4 независимых каналов для:
  - Захвата входа
  - Сравнение выхода
  - ШИМ генерация (Фронт и Центр)
  - Одиночный импульс
- Инверсные выходы с программируемой задержкой
- Схема синхронизации внешнего управления и объединения таймеров.
- Счётчик повторения для обновления регистров таймера только после нескольких циклов счёта.
- Вход остановки для перевода выходных сигналов таймера в заданное состояние.
- Генерация Прерываний/DMA по следующим событиям:
  - Обновление: переполнение/исчерпание, инициализация счётчика (программно или запуском)
  - Запуск (старт, стоп, инициализация или счёт по сигналу)
  - Захват входа
  - Сравнение выхода
  - Вход остановки.
- Поддерживает инкрементный (квадратурный) кодер и датчики Холла для позиционирования.
- Вход запуска от внешних тактов или шагового управления током.

Рис. 86. Блок-схема улучшенных таймеров.



17.3. Функциональное описание таймеров

17.3.1. Узел счёта

Это 16-бит счётчик с регистром само-перезагрузки. Имеет прямой, обратный и реверсивный режимы. Тактирование может поступать через предделитель.

Счётчик, регистр перезагрузки и предделитель доступны программно для чтения и записи даже во время счёта.

Включает:

- Регистр счётчика (TIMx_CNT)
- Регистр предделителя (TIMx_PSC)
- Регистр перезагрузки (TIMx_ARR)
- Регистр повторения счёта (TIMx_RCR)

Регистр перезагрузки предзагружаемый (есть видимый и скрытый регистры). Содержимое видимого регистра пишется в скрытый по каждому событию обновления (UEV), в зависимости от бита разрешения (ARPE) в регистре TIMx_CR1. Оно посылается программно или при переполнении/исчерпании счётчика при снятом бите UDIS в регистре TIMx_CR1.

Счётчик тактируется выходом предделителя CK_CNT. Разрешается битом CEN в регистре TIMx_CR1.

Счёт начинается через 1 такт после установки бита CEN в регистре TIMx_CR1.

Описание предделителя

Это 16-бит счётчик делителя входной частоты (от 1 до 65536) с видимым регистром TIMx_PSC. его можно менять налету. Новое значение счётчика пишется по следующему событию обновления.

Рис. 87. Временная диаграмма с изменением предделителя с 1 на 2.

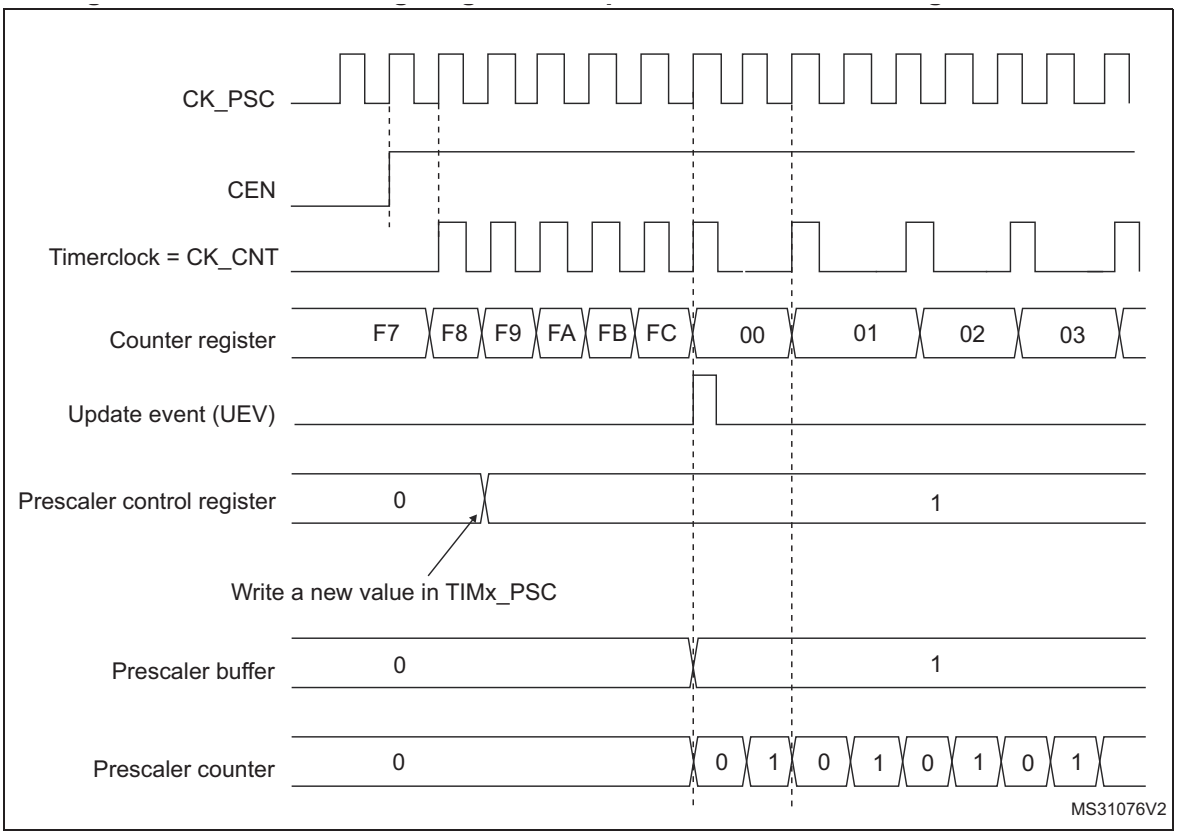
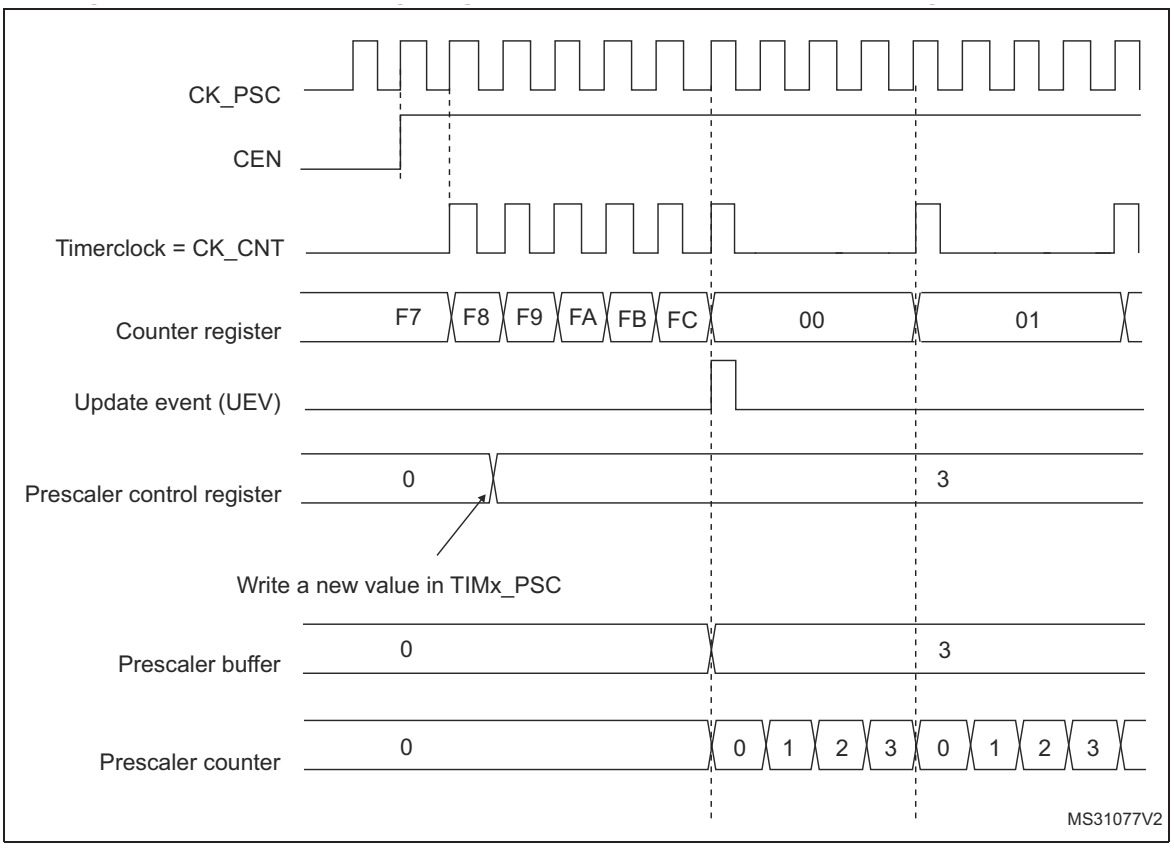


Рис. 88. Временная диаграмма с изменением предделителя с 1 на 4.



17.3.2. Режимы счёта

Прямой счёт

Счёт идёт от 0 до значения перезагрузки (регистр TIMx_ARR), сбрасывается в 0 и выдаёт событие переполнения счётчика.

Если используется счётчик повторений, то событие UEV выдаётся после обнуления заданных повторений плюс 1 (TIMx_RCR+1). Иначе событие обновления выдаётся при каждом переполнении счётчика.

Установка бита UG в регистре TIMx_EGR (программно или контроллером режима ведомого) также выдаёт событие обновления.

Событие UEV выключается установкой бита UDIS в регистре TIMx_CR1. Этим предупреждается запись скрытого регистра во время изменения регистра предзагрузки. Также событие UEV не появляется при сброшенном бите UDIS 0. Однако, счётчик продолжает считать с 0, равно как и предделитель (не изменив своего значения). Кроме того, если стоит бит URS в регистре TIMx_CR1, то установка бита UG выдаёт событие UEV без установки флага UIF (без прерывания и запроса DMA). Этим предупреждается одновременная выдача прерываний обновления и захвата при очистке счётчика по событию захвата.

По событию обновления обновляются все регистры и ставится флаг обновления (бит UIF в регистре TIMx_SR). Это зависит от бита URS:

- Счётчик повторений перегружается из регистра TIMx_RCR,
- В скрытый регистр перезагрузки пишется содержимое TIMx_ARR,
- Буфер предделителя перегружается из регистра TIMx_PSC.

Примеры ниже используют TIMx_ARR=0x36.

Рис. 89. Временная диаграмма с делителем 1.

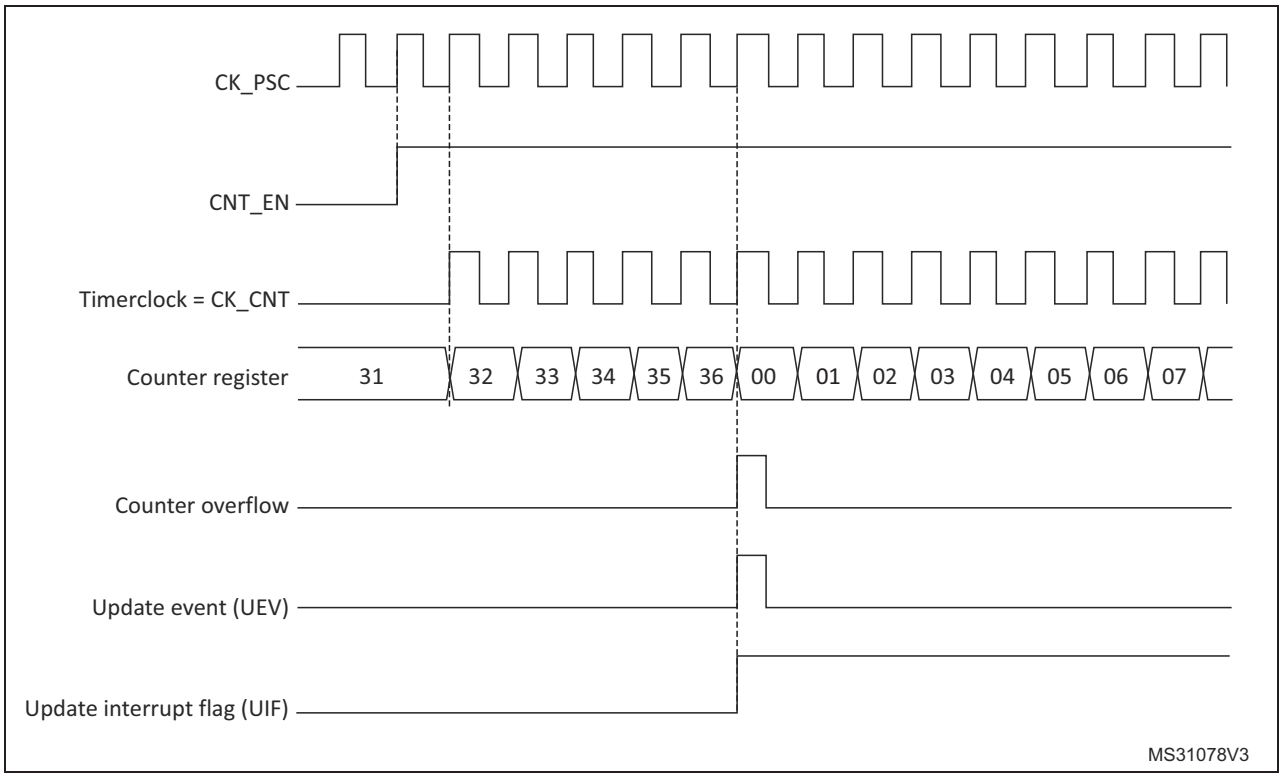


Рис. 90. Временная диаграмма с делителем 2.

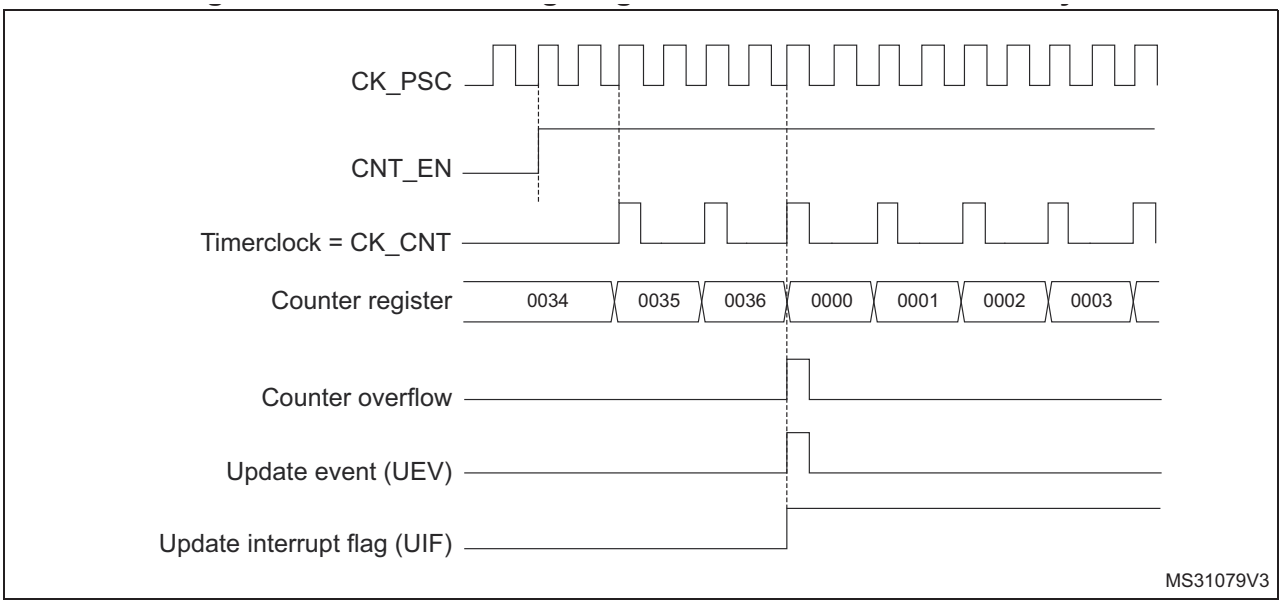


Рис. 91. Временная диаграмма с делителем 4.

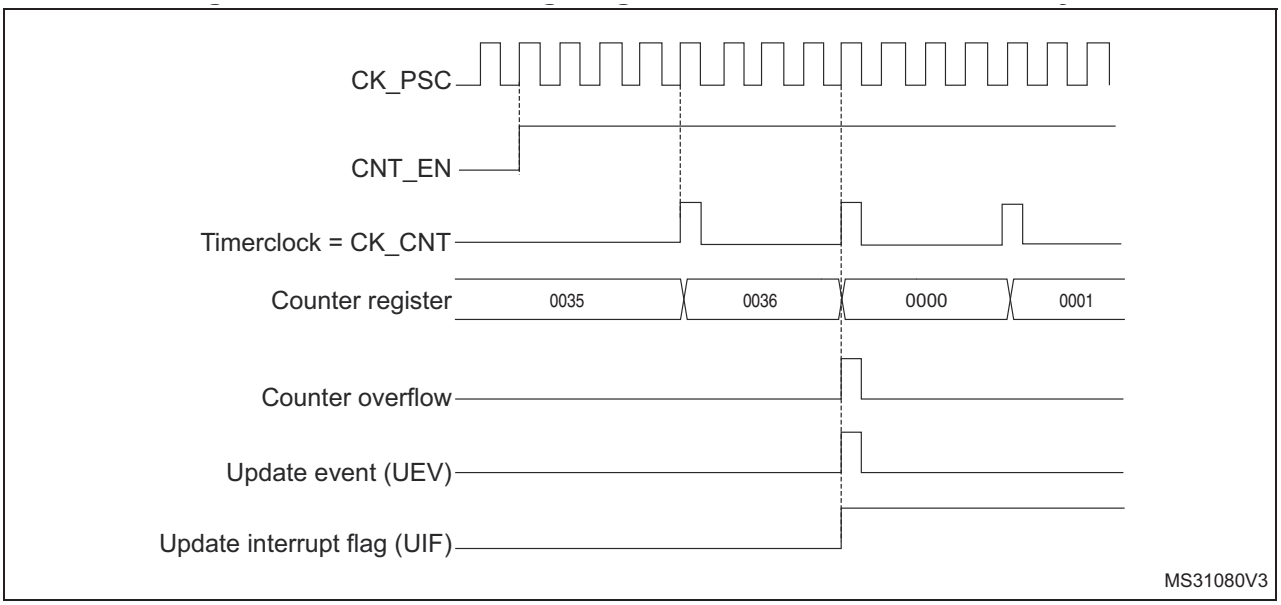


Рис. 92. Временная диаграмма с делителем N.

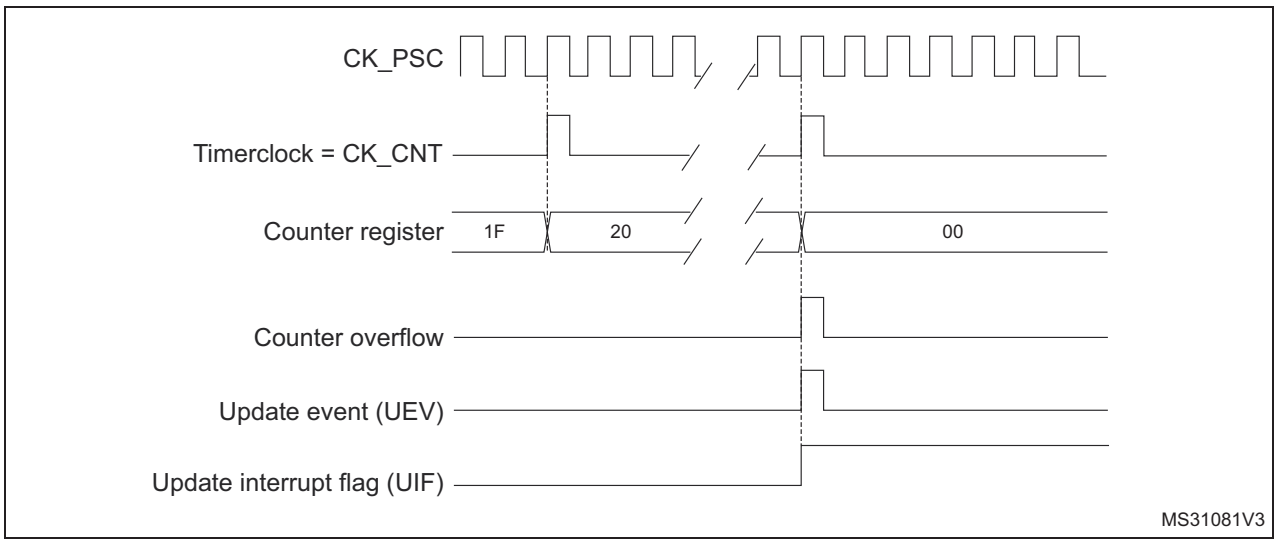


Рис. 93. Временная диаграмма события при ARPE=0 (TIMx_ARR не предзагружен).

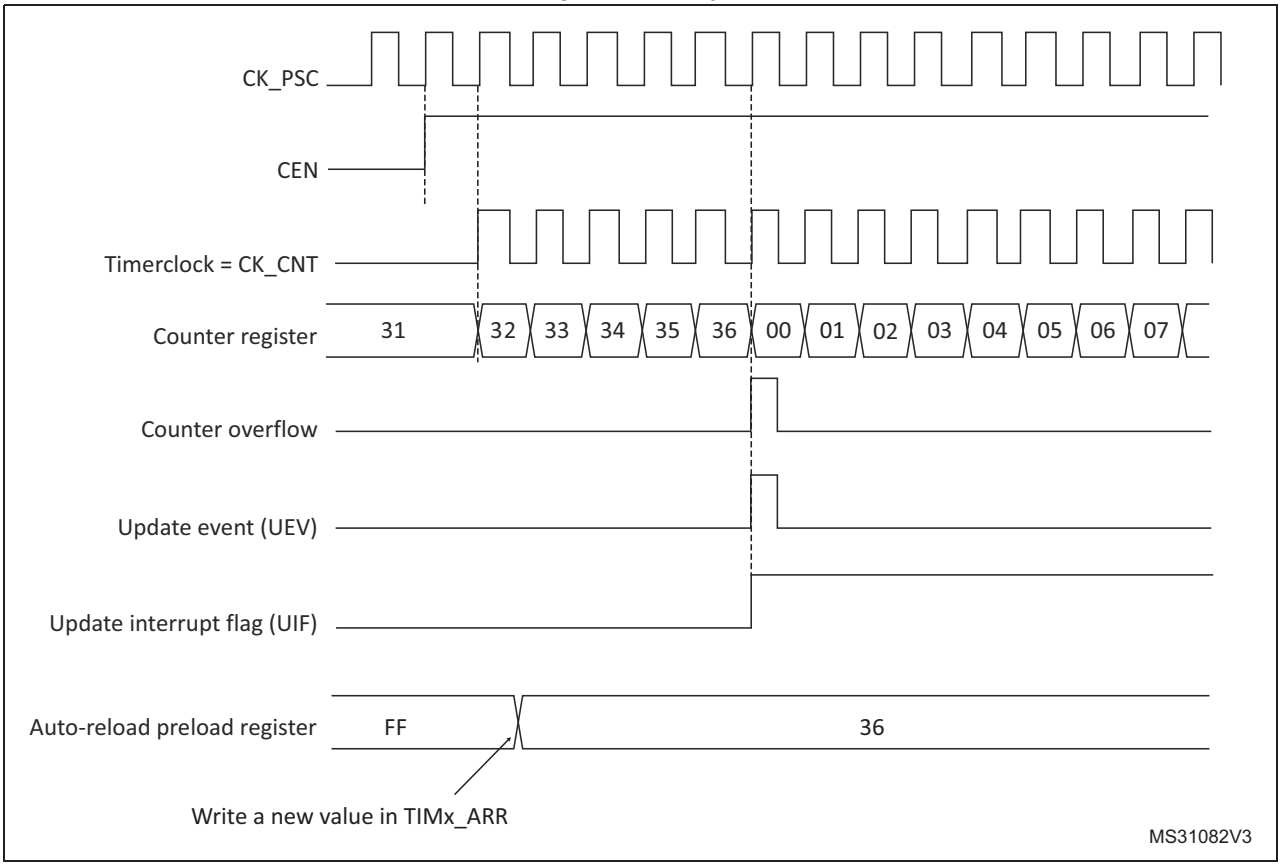
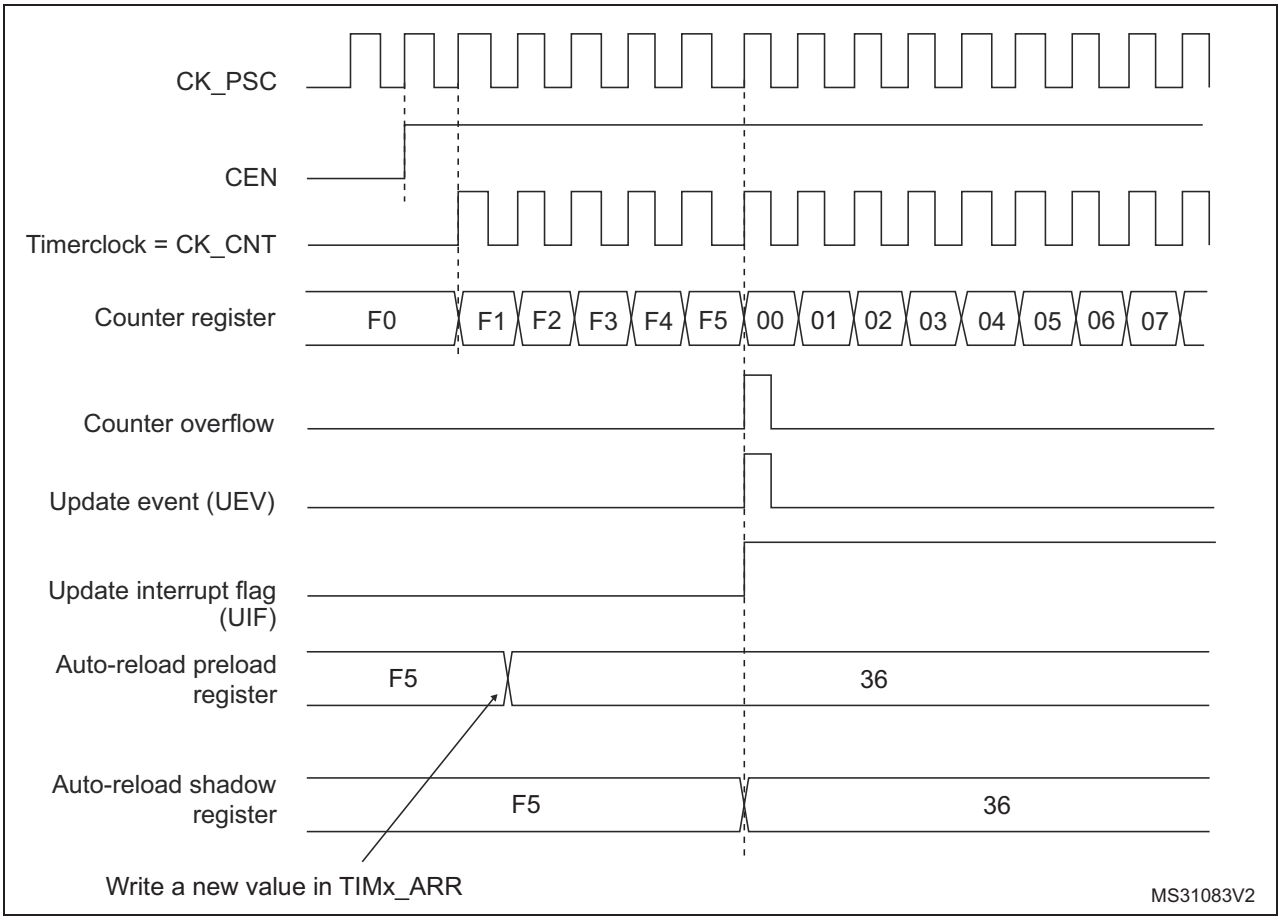


Рис. 94. Временная диаграмма события при ARPE=1 (TIMx_ARR предзагружен).



Обратный счёт

Счёт идёт от значения перезагрузки (регистр TIMx_ARR) до 0, перегружает счётчик и выдаёт событие исчерпания счётчика.

Если используется счётчик повторений, то событие UEV выдаётся после исчерпания заданных повторений плюс 1 (TIMx_RCR+1). Иначе событие обновления выдаётся при каждом исчерпании счётчика.

Установка бита UG в регистре TIMx_EGR (программно или контроллером режима ведомого) также выдаёт событие обновления.

Событие UEV выключается установкой бита UDIS в регистре TIMx_CR1. Этим предупреждается запись скрытого регистра во время изменения регистра предзагрузки. Также событие UEV не появляется при сброшенном бите UDIS 0. Однако, счётчик продолжает считать с значения перезагрузки, тогда как и предделитель с 0 (не изменив своего значения). Кроме того, если стоит бит URS в регистре TIMx_CR1, то установка бита UG выдаёт событие UEV без установки флага UIF (без прерывания и запроса DMA). Этим предупреждается одновременная выдача прерываний обновления и захвата при очистке счётчика по событию захвата.

По событию обновления обновляются все регистры и ставится флаг обновления (бит UIF в регистре TIMx_SR). Это зависит от бита URS:

- Счётчик повторений перегружается из регистра TIMx_RCR,
- Буфер предделителя перегружается из регистра TIMx_PSC.
- В скрытый регистр перезагрузки пишется содержимое TIMx_ARR. Скрытый регистр перезагрузки обновляется раньше счётчика и работа продолжается с нового значения.

Примеры ниже используют TIMx_ARR=0x36.

Рис. 95. Временная диаграмма с делителем 1.

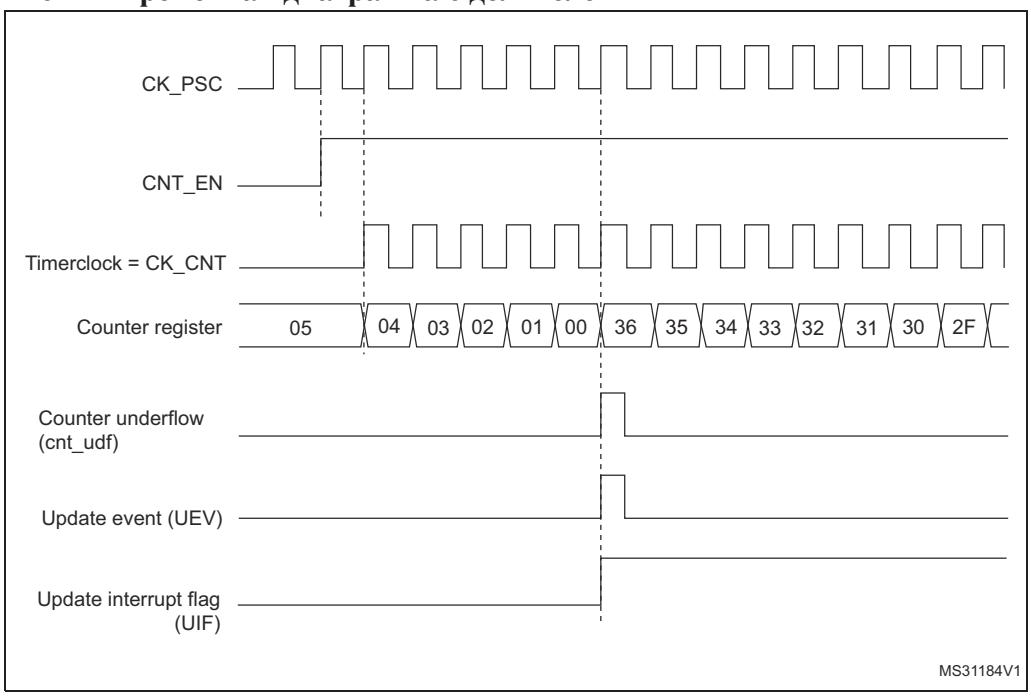


Рис. 96. Временная диаграмма с делителем 2.

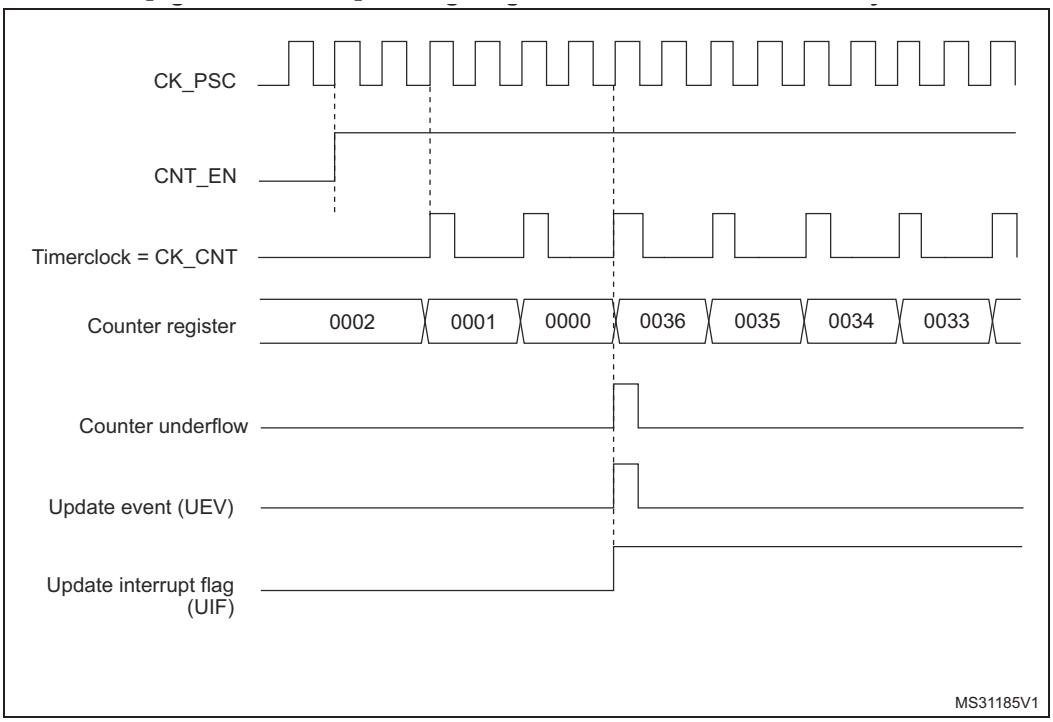


Рис. 97. Временная диаграмма с делителем 4.

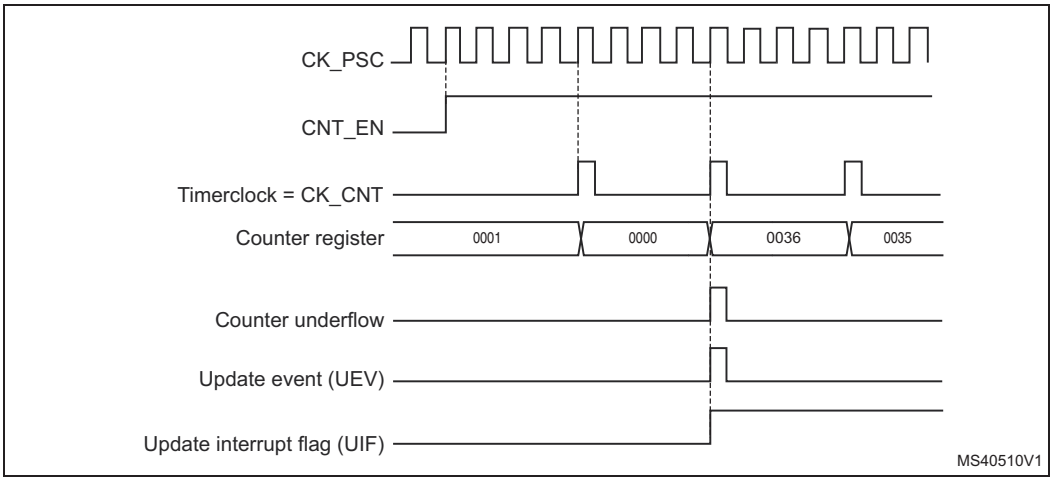


Рис. 98. Временная диаграмма с делителем N.

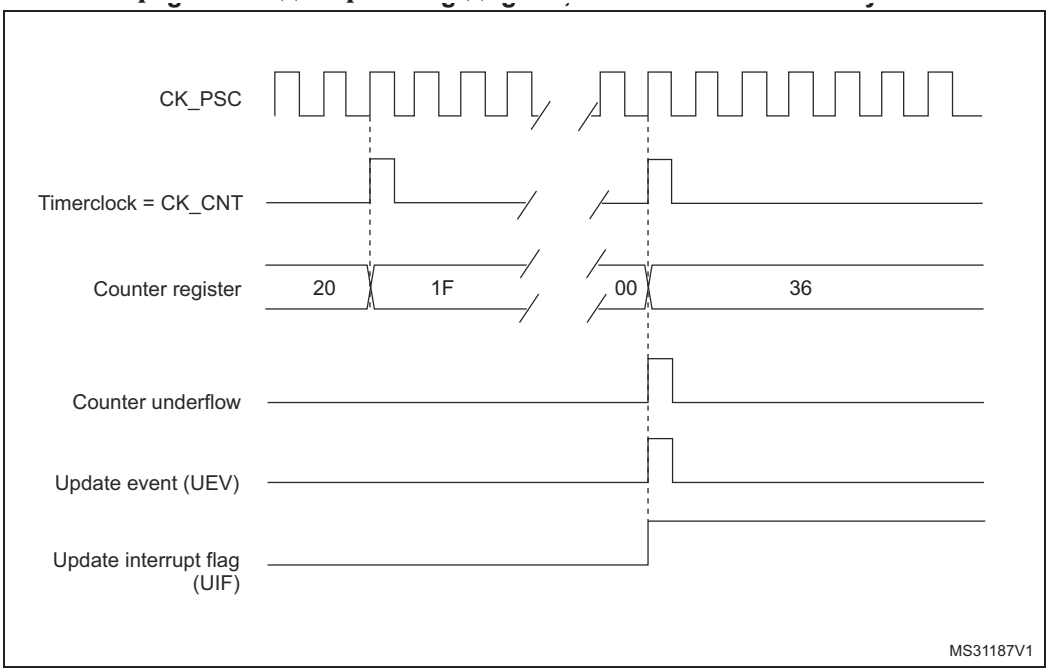
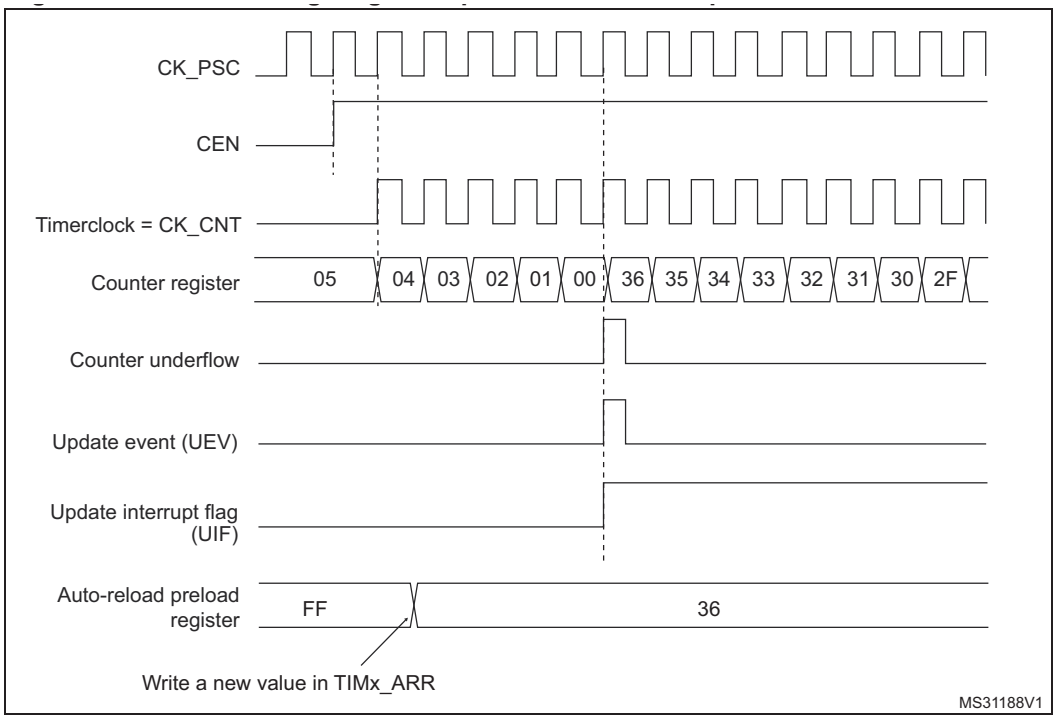


Рис. 99. Временная диаграмма события без счётчика повторений.



Реверсивный счёт

Счёт идёт от 0 до значения перезагрузки минус 1 (регистр TIMx_ARR-1), выдаёт событие переполнения счётчика, затем считает обратно до 1, выдаёт событие исчерпания счётчика и начиняет счёт с 0.

Реверсивный режим включается если биты CMS в регистре TIMx_CR1 не равны '00'. Флаг прерывания Сравнения выхода выводных каналов ставится когда: завершается обратный счёт (Реверсивный режим 1, CMS = "01"), завершается прямой счёт (Реверсивный режим 2, CMS = "10") завершается прямой и обратный счёт (Реверсивный режим 3, CMS = "11").

В этом режиме бит DIR в регистре TIMx_CR1 писать нельзя. Он изменяется аппаратно и указывает текущее направление счёта.

Событие обновления может выдаваться на каждое переполнение и на каждое исчерпание счётчика установкой бита UG в регистре TIMx_EGR (программно или контроллером режима ведомого). В этом случае счётчик и предделитель начинают счёт с 0.

Событие UEV выключается установкой бита UDIS в регистре TIMx_CR1. Этим предупреждается запись скрытого регистра во время изменения регистра предзагрузки. Также событие UEV не появляется при сброшенном бите UDIS 0. Однако, счётчик продолжает считать с вверх и вниз, основываясь на текущем значении предзагрузки.

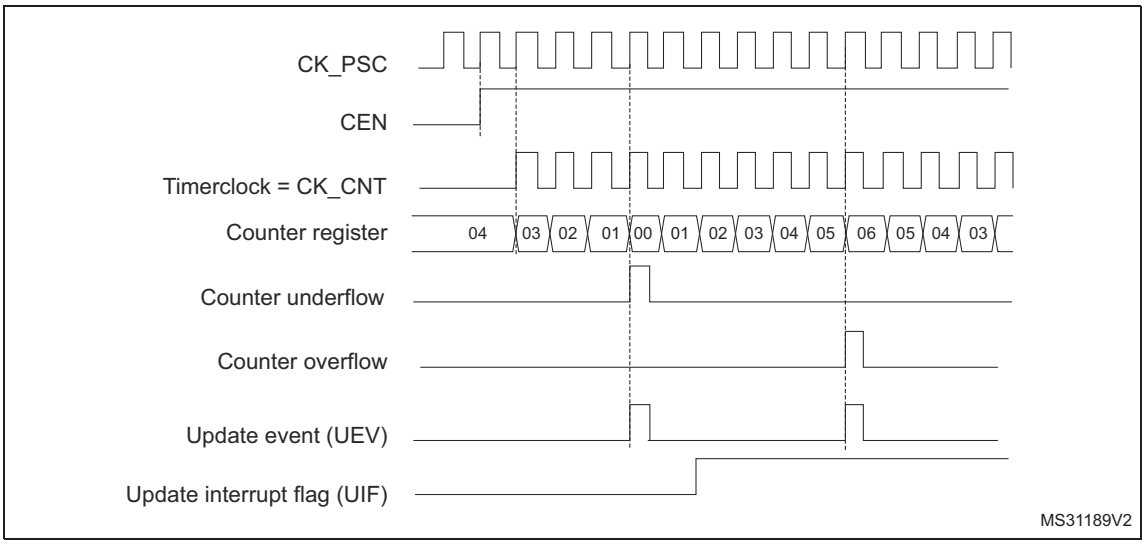
Кроме того, если стоит бит URS в регистре TIMx_CR1, то установка бита UG выдаёт событие UEV без установки флага UIF (без прерывания и запроса DMA). Этим предупреждается одновременная выдача прерываний обновления и захвата при очистке счётчика по событию захвата.

По событию обновления обновляются все регистры и ставится флаг обновления (бит UIF в регистре TIMx_SR). Это зависит от бита URS:

- Счётчик повторений перегружается из регистра TIMx_RCR,
- Буфер предделителя перегружается из регистра TIMx_PSC.
- В скрытый регистр перезагрузки пишется содержимое TIMx_ARR. В случае обновления по переполнению счётчика скрытый регистр перезагрузки обновляется раньше счётчика и работа продолжается с нового значения.

Примеры ниже используют разные частоты тактов.

Рис. 100. Временная диаграмма с делителем 1, TIMx_ARR = 0x6.



1. Реверсивный режим 1.

Рис. 101. Временная диаграмма с делителем 2.

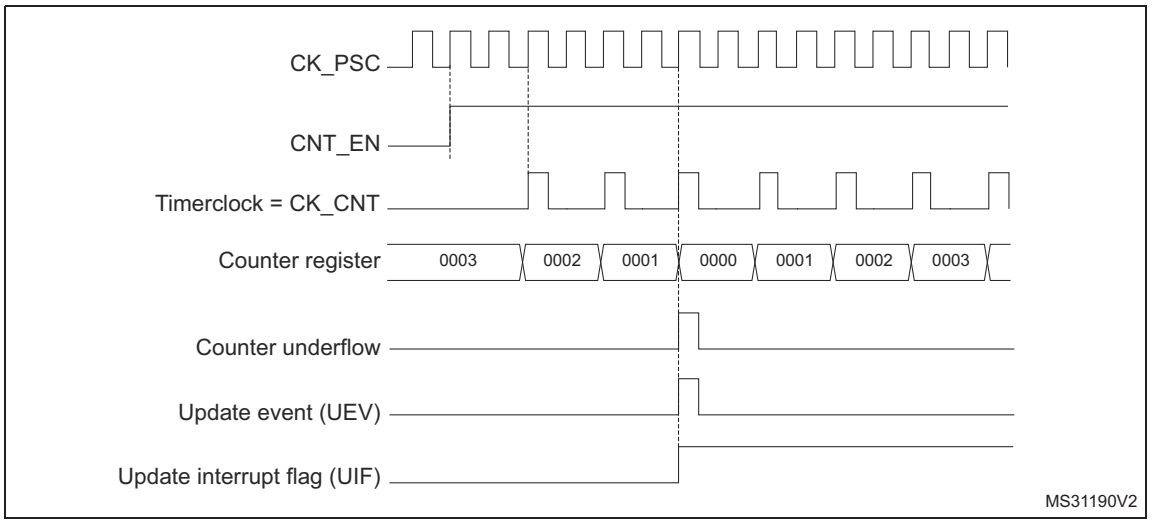
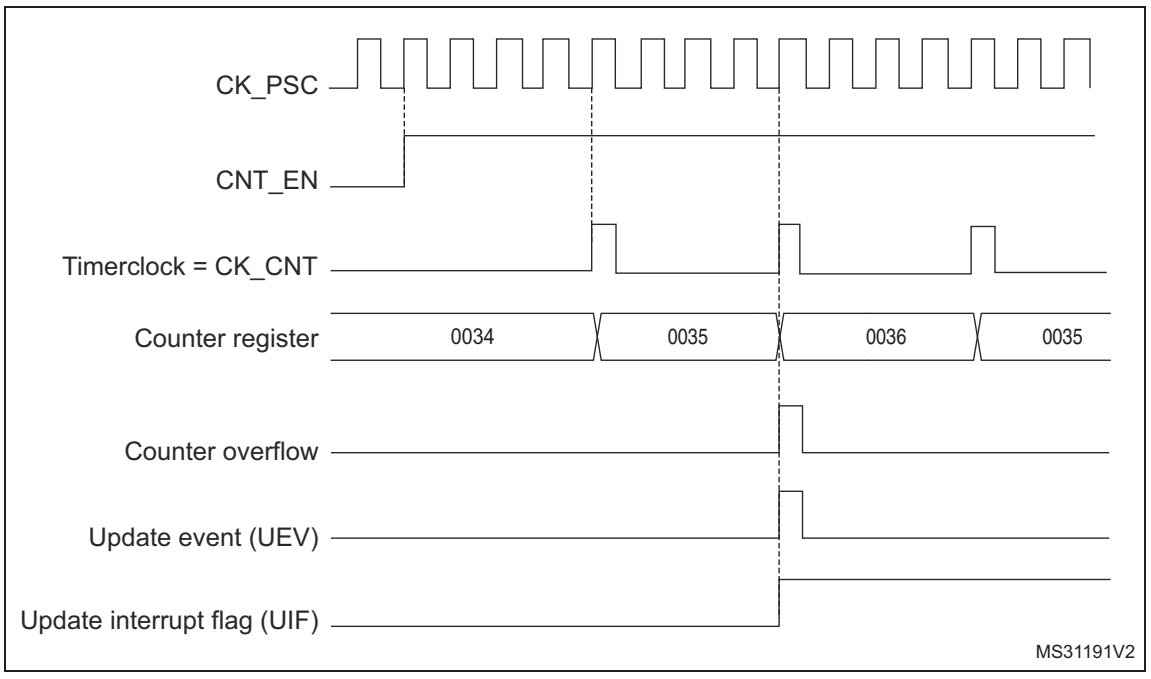


Рис. 102. Временная диаграмма с делителем 4, TIMx_ARR=0x36.



1. Реверсивный режим 2 или 3 с флагом UIF по переполнению.

Рис. 103. Временная диаграмма с делителем N.

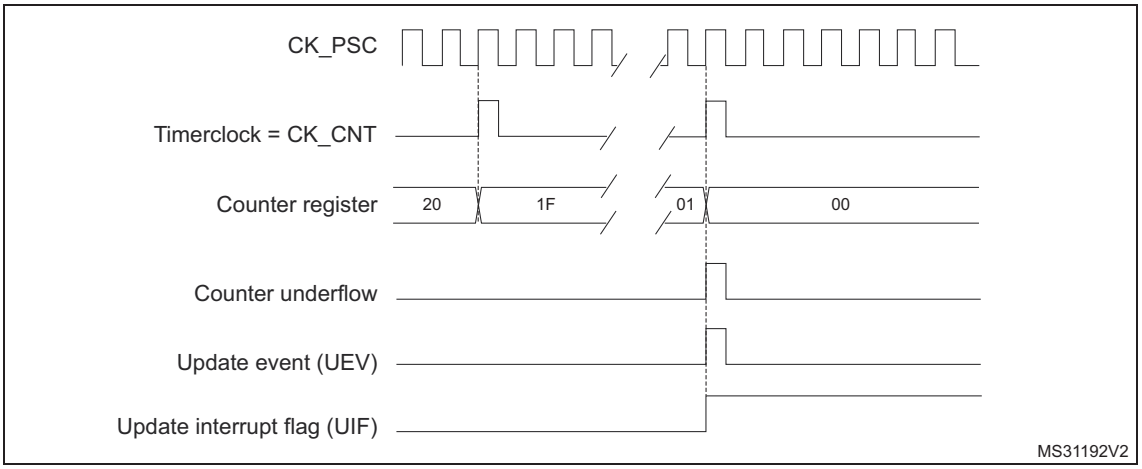


Рис. 104. Временная диаграмма, обновление с ARPE=1 (исчерпание).

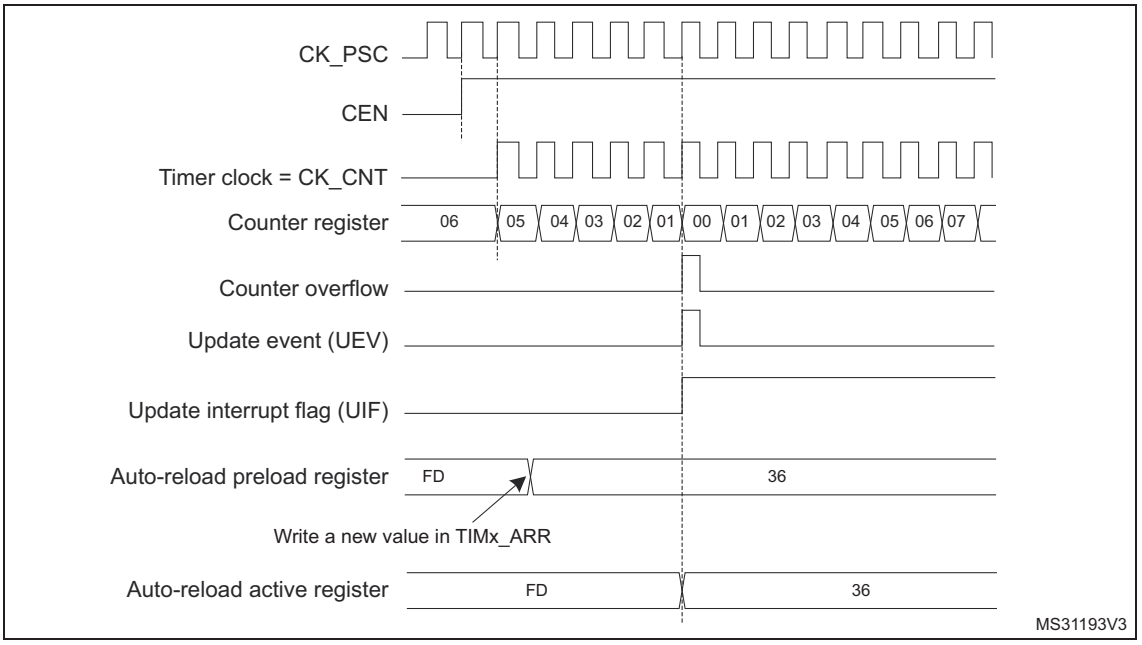
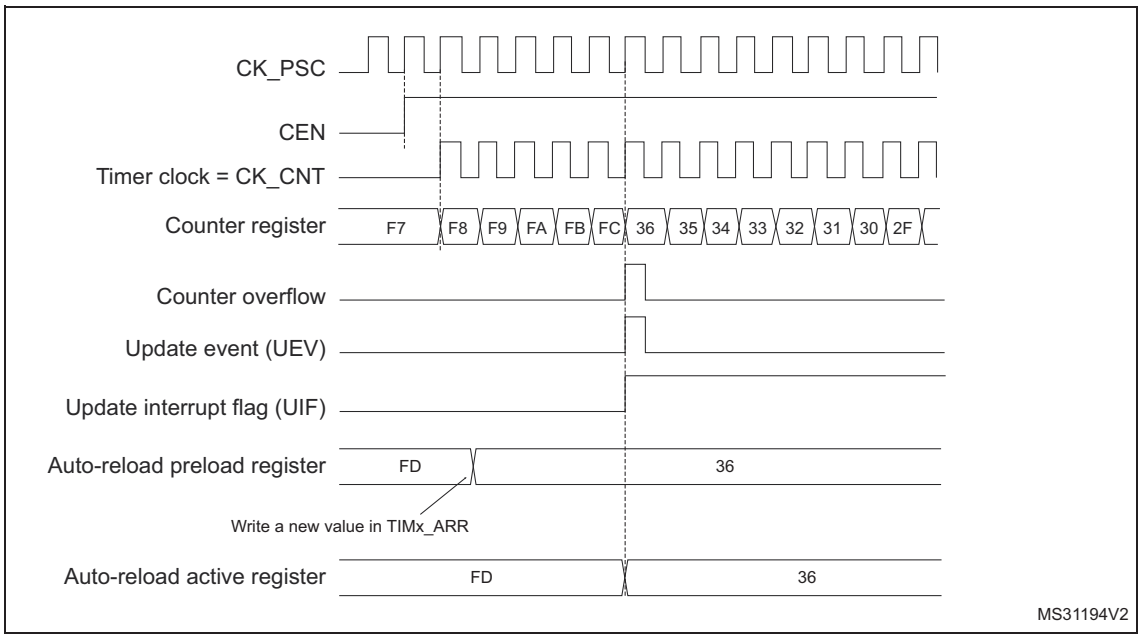


Рис. 105. Временная диаграмма, обновление с ARPE=1 (переполнение).



17.3.3. Счётчик повторения

В действительности событие обновления (UEV) выдаётся только при нулевом счётчике повторения. Это полезно при генерации ШИМ сигналов.

То есть, данные из регистров TIMx_ARR, TIMx_PSC, и TIMx_CCRx пишутся в скрытые регистры каждые N+1 переполнений или исчерпаний счётчика, где N значение счётчика повторений TIMx_RCR.

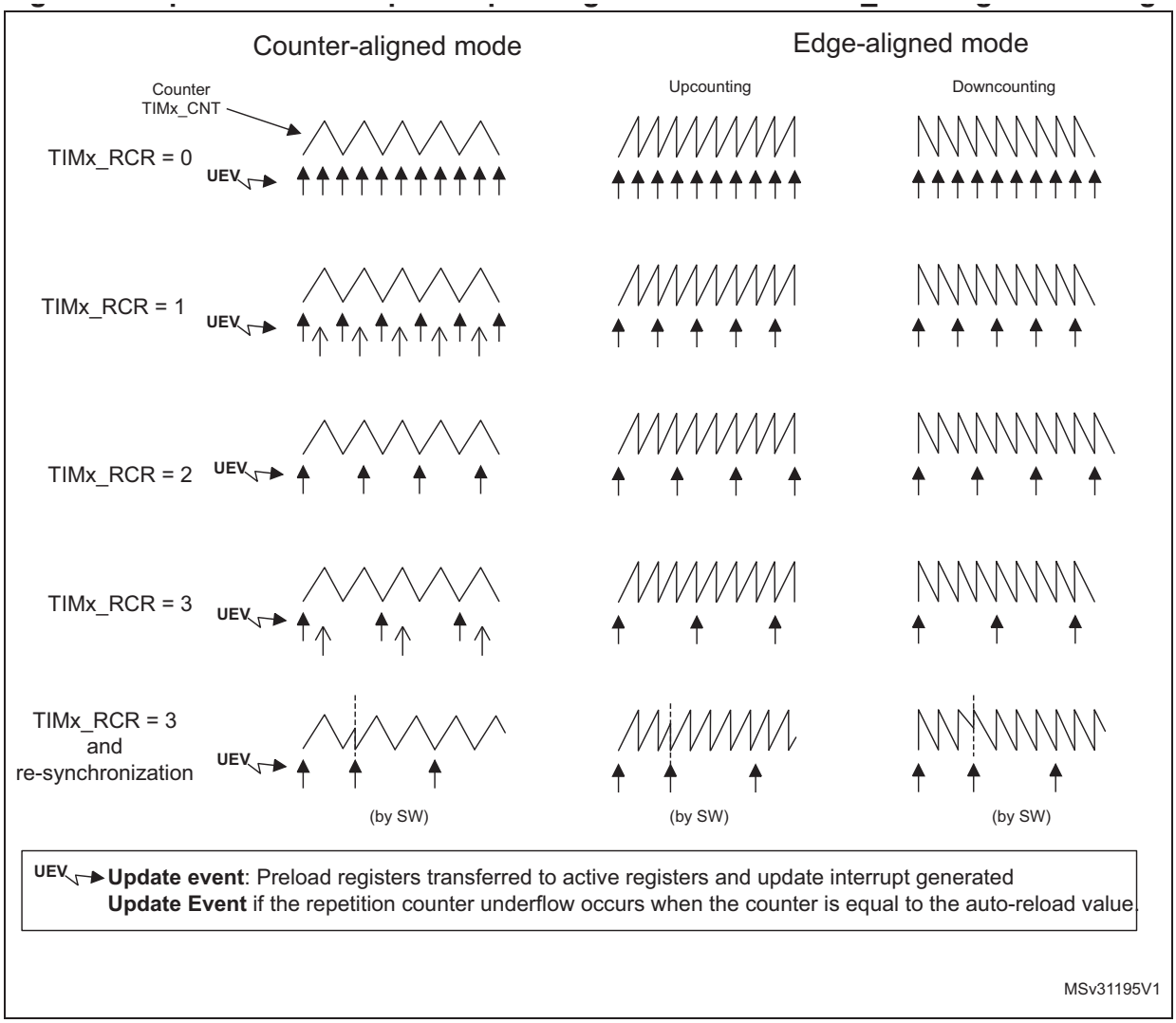
Счётчик повторений декрементируется:

- На каждое переполнение счётчика при прямом счёте,
- На каждое исчерпание счётчика при обратном счёте,
- На каждое переполнение и исчерпание счётчика при реверсивном счёте. Хотя это ограничивает число повторений 128 циклами ШИМ, зато так можно обновлять заполнение цикла дважды за период ШИМ. При обновлении регистров сравнения единожды за период ШИМ в реверсивном режиме максимальное разрешение равно $2xT_{ck}$ из-за симметрии сигнала.

Счётчик повторений сам перегружается из регистра TIMx_RCR. Если событие UEV выдаётся программно (битом UG в регистре TIMx_EGR) или аппаратно контроллером ведомого, то счётчик повторений немедленно загружается из регистра TIMx_RCR, независимо от его содержимого.

В реверсивном режиме при нечётных значениях RCR, событие обновления выдаётся либо при переполнении счётчика, либо при исчерпании, в зависимости от того, когда был записан регистр RCR, до старта счётчика, или после него. Если RCR был записан до старта, то UEV появляется при переполнении, иначе при исчерпании. Например при RCR = 3, событие UEV выдаётся на каждое 4-е переполнение или исчерпание, в зависимости от того, когда был записан RCR.

Рис. 106. Обновления в зависимости от установок регистра TIMx_RCR.



17.3.4. Выбор тактов

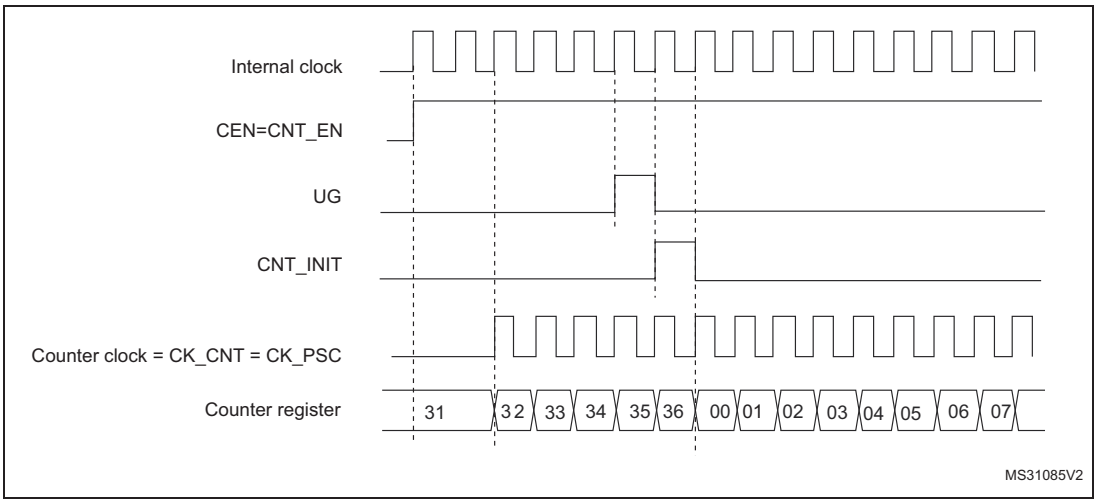
Счётчик может тактироваться из следующих источников:

- Внутренний (CK_INT)
- Внешний режим 1: внешняя ножка
- Внешний режим 2: вход внешнего сигнала ETR
- Входы внутреннего сигнала (ITRx): использование одного таймера предделителем для другого.

Внутренний источник (CK_INT)

Если в ведомом режиме контроллер выключен (SMS=000), то всем управляют только биты CEN, DIR (в регистре TIMx_CR1) и UG (в регистре TIMx_EGR). Они изменяются только программно (кроме UG, снимающегося аппаратно). Сразу после установки бита CEN предделитель начинает получать внутренние такты CK_INT.

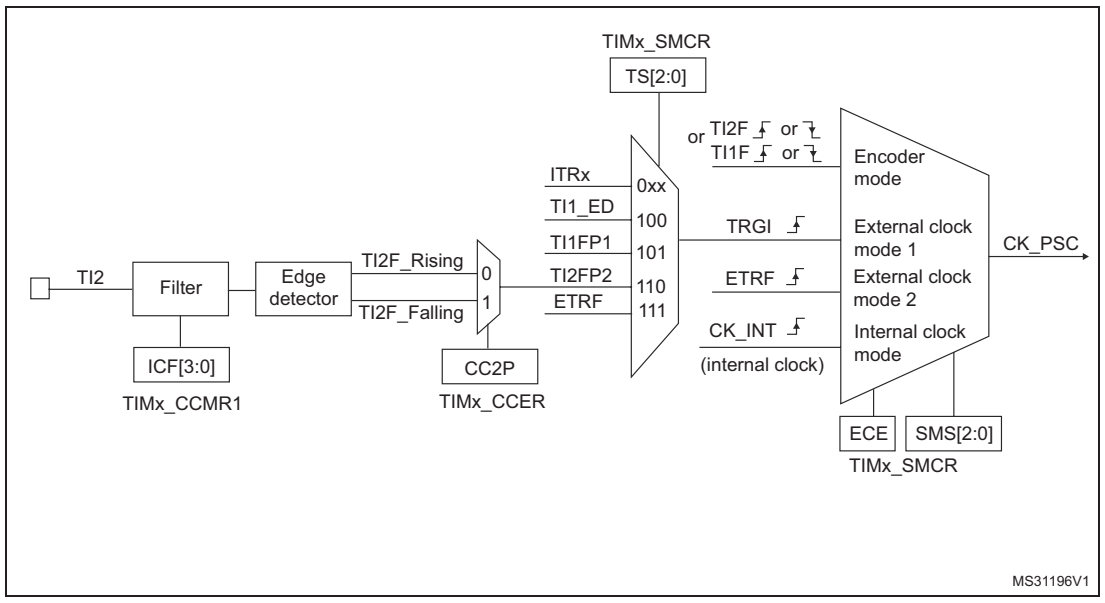
Рис. 107. Схема управления в нормальном режиме прямого счёта без предделителя.



Внешний источник режим 1.

Включается при установке SMS=111 в регистре TIMx_SMCR. Счётчик работает от переднего или заднего фронта выбранного входа.

Рис. 108. Внешний источник TI2.



Например, процедура включения режима прямого счёта по переднему фронту входа TI2:

1. Включить определение переднего фронта TI2 канала 2 записью CC2S = '01' в регистре TIMx_CCMR1.

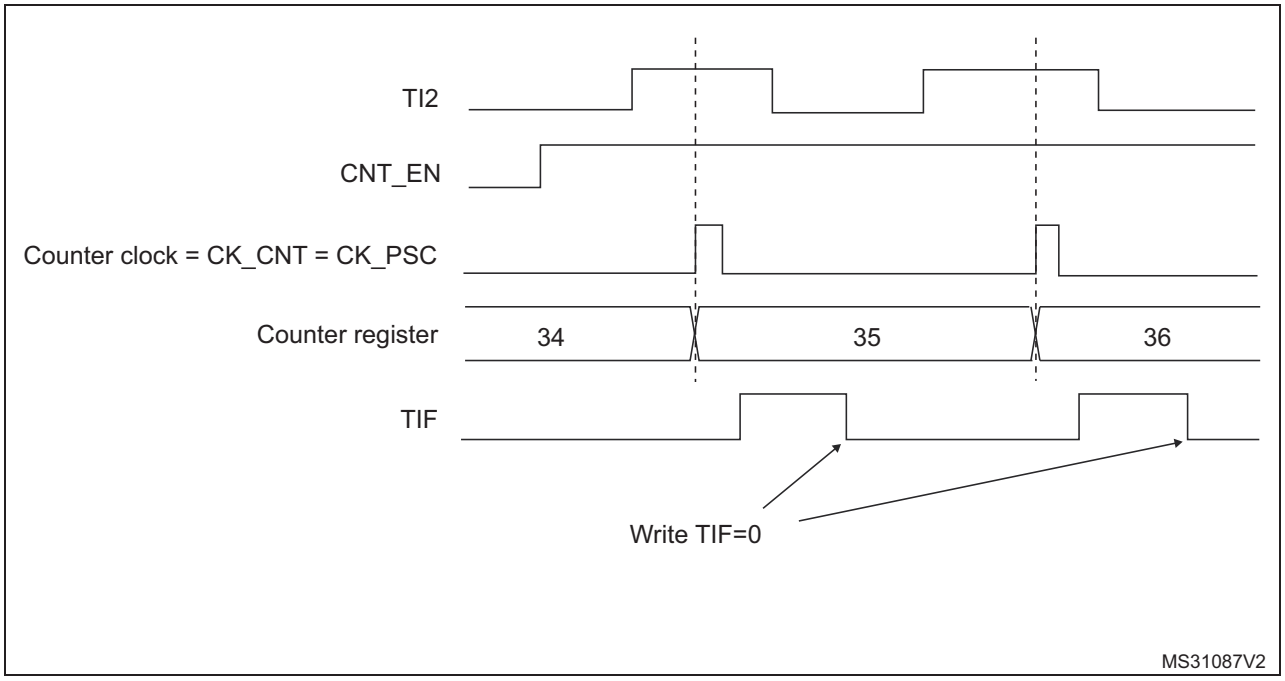
2. Установить длительность входного фильтра битами IC2F[3:0] в регистре TIMx_CCMR1 (если фильтр не нужен, то остаётся IC2F=0000).
3. Выбрать полярность переднего фронта записью CC2P=0 в регистре TIMx_CCER.
4. Включить внешний режим 1 тактирования записью SMS=111 в регистре TIMx_SMCR.
5. Выбрать TI2 как источник сигнала записью TS=110 в регистре TIMx_SMCR.
6. Включить счётчик записью CEN=1 в регистре TIMx_CR1.

Предделитель захвата для запуска не используется и конфигурировать его нужды нет.

По переднему фронту на TI2 счётчик один раз тикает и ставит флаг TIF.

Задержка между передним фронтом на TI2 и реальным тиком появляется из-за схемы ресинхронизации на входе TI2.

Рис. 109. Схема управления внешнего режима 1.

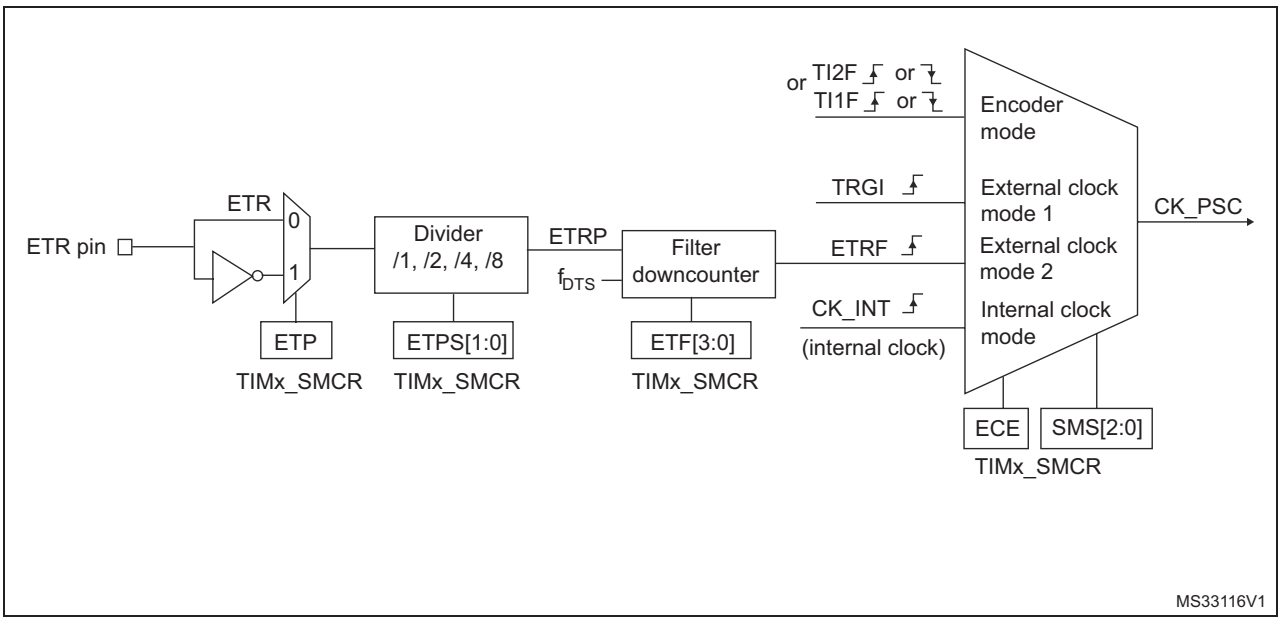


Внешний источник режим 2.

Включается записью ECE=1 в регистре TIMx_SMCR.

Счётчик работает по переднему или заднему фронту входа ETR.

Рис. 110. Блок входа внешнего запуска.



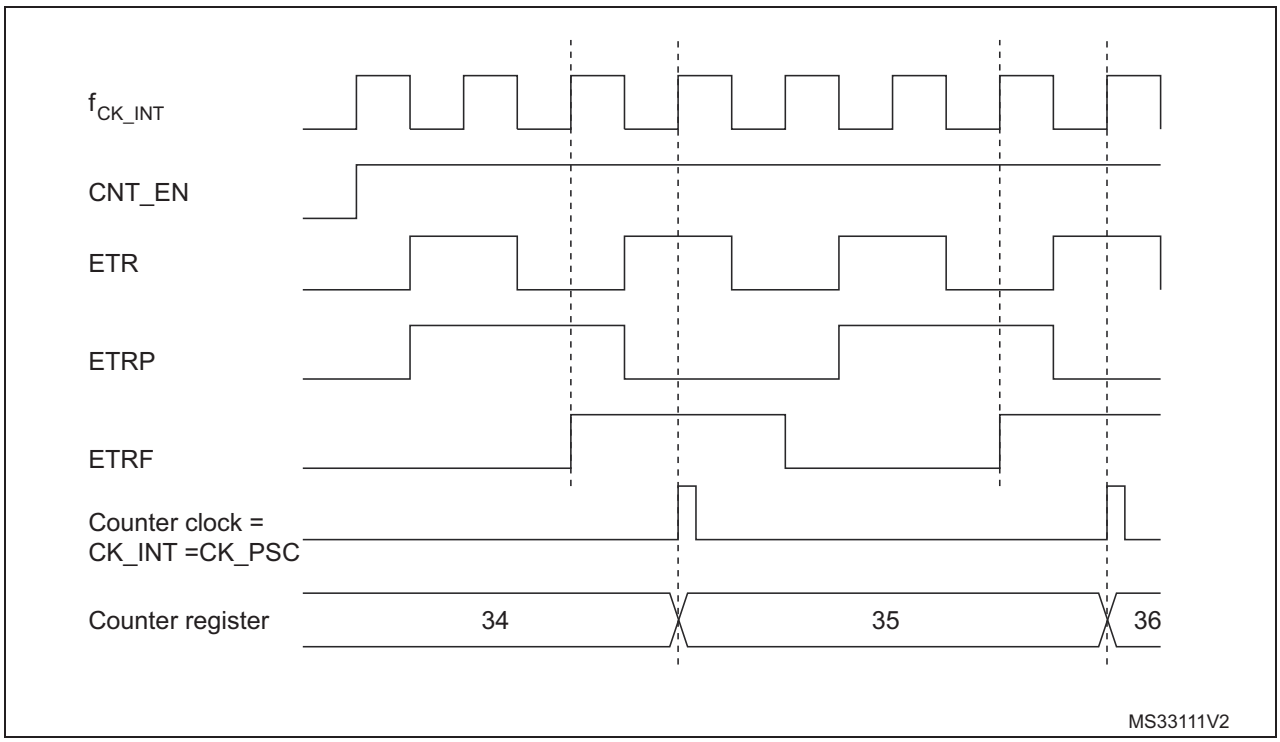
Например, включение режима прямого счёта по каждому 2 переднему фронту на входе ETR:

1. Здесь фильтр не нужен, так что ETF[3:0]=0000 в регистр TIMx_SMCR.
2. В предделитель пишем ETPS[1:0]=01 в регистре TIMx_SMCR
3. Выбираем передний фронт на ножке ETR записью ETP=0 в регистре TIMx_SMCR.
4. Разрешаем внешний режим 2 записью ECE=1 в регистре TIMx_SMCR.
5. Включаем счётчик записью CEN=1 в регистре TIMx_CR1.

Счётчик тикает единожды на каждые 2 передних фронта ETR.

Задержка между передним фронтом на ETR и реальным тиком появляется из-за схемы ресинхронизации на входе сигнала ETRP.

Рис. 111. Схема управления с внешним источником режим 2.

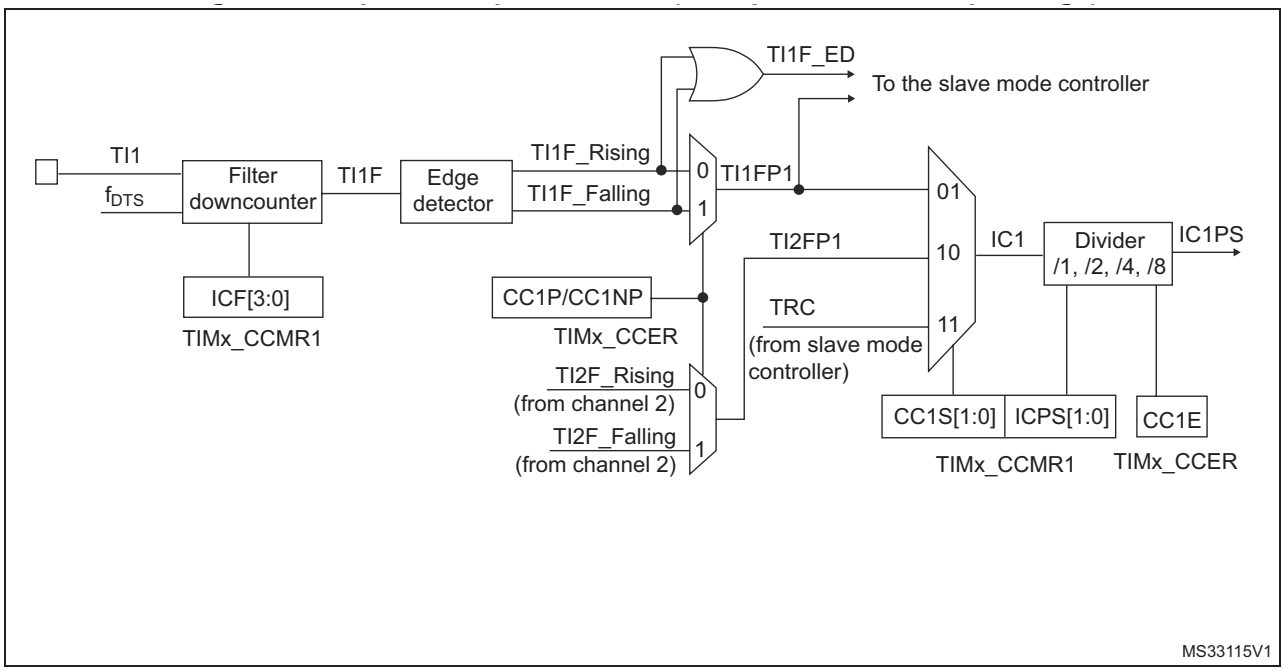


17.3.5. Каналы захвата/сравнения

Каждый канал захвата/сравнения построен вокруг парного регистра (включая невидимый), входного каскада захвата (с цифровым фильтром, мультиплексором и предделителем) и выходного каскада (с компаратором и управлением выходом).

Входной каскад считывает нужный вход TIx для выдачи фильтрованного сигнала TIxF. Далее, детектор фронта с выбором полярности выдаёт сигнал (TIxFPx), что может быть использован как вход запуска контроллером режима ведомого или как команда захвата. Предделитель стоит до регистра захвата (ICxPS).

Рис. 112. Входной каскад канала 1.



Выходной каскад выдаёт промежуточный сигнал, используемый как опорный: OCxRef (активный высокий). Полярность действует в конце цепочки.

Рис. 113. Основная схема канала 1.

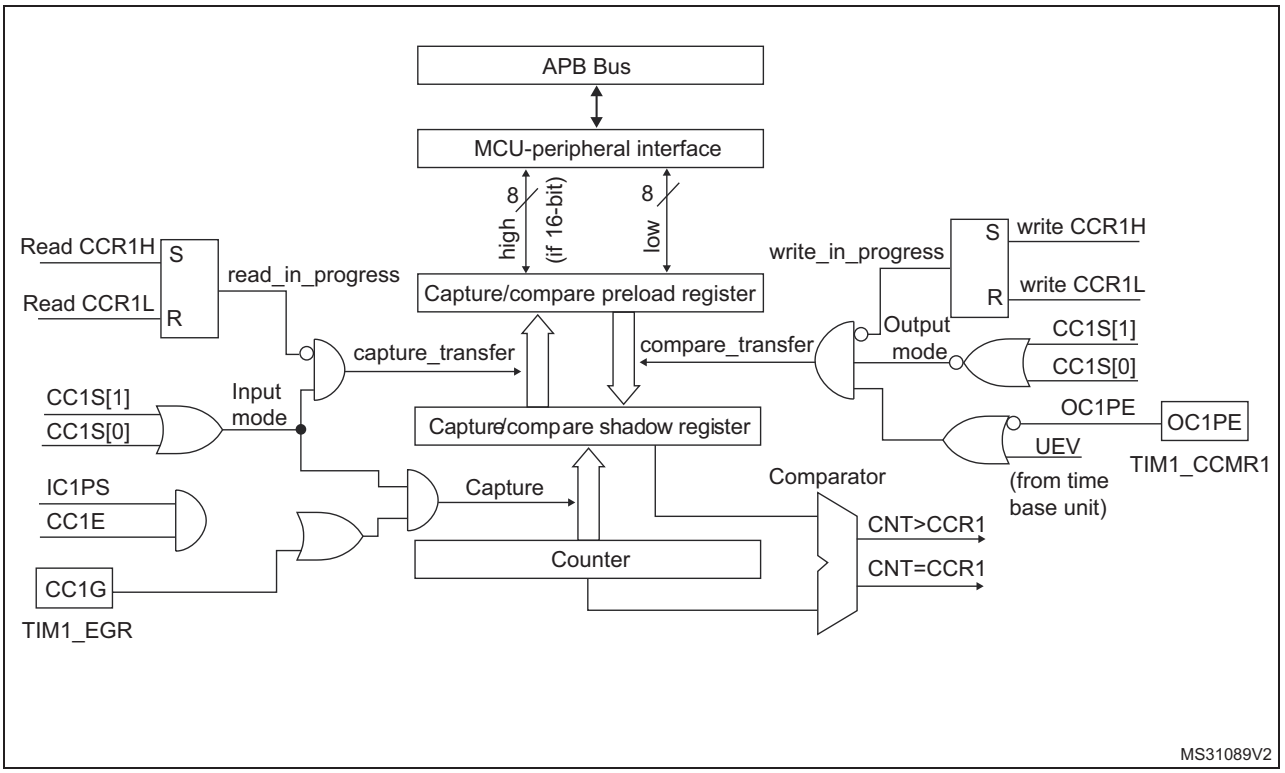


Рис. 114. Выходной каскад каналов 1 - 3.

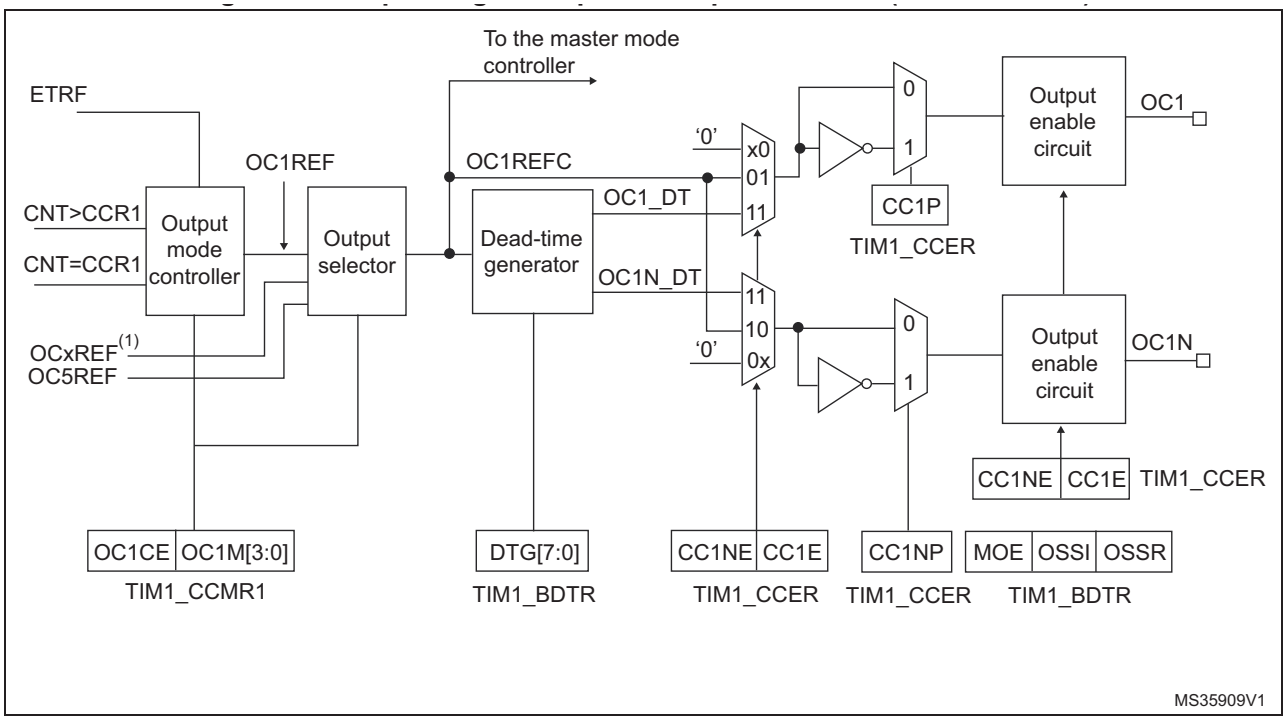
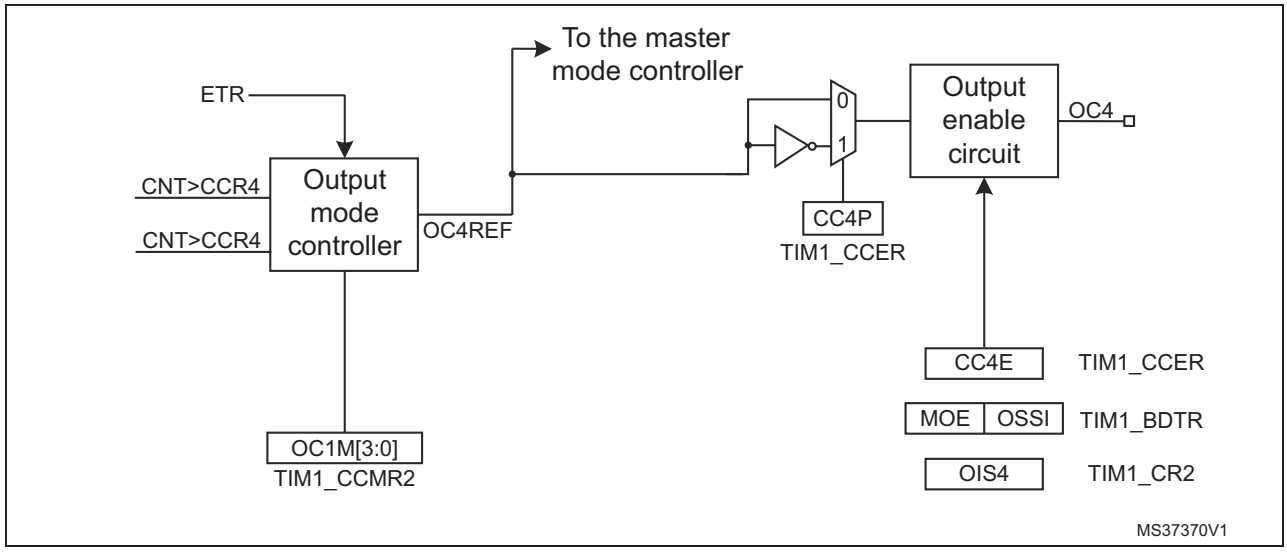


Рис. 115. Выходной каскад канала 4.



Блок захвата/сравнения состоит из двух регистров: предзагрузки и невидимого. Снаружи доступен только регистр предзагрузки и по чтению и по записи.

Захват выполняется внутренним регистром и результат пишется в регистр предзагрузки.

При сравнении регистр предзагрузки пишется в невидимый, где и сравнивается со счётчиком.

17.3.6. Режим захвата входа

Регистры `TIMx_CCRx` защёлкивают содержимое счётчика после передачи, обнаруженной соответствующим сигналом `ICx`. При этом ставится флаг `CCXIF` в регистре `TIMx_SR` и ставятся запрос DMA и прерывания. Если захват происходит при стоящем флаге `CCxIF`, то ставится флаг перезахвата `CCxOF` в регистре `TIMx_SR`. Флаг `CCxIF` снимается записью в него нуля или чтением захваченных данных из регистра `TIMx_CCRx`. `CCxOF` снимается записью '`0`'.

Далее захватываем содержимое `TIMx_CCR1` по переднему фронту `TI1`. Делаем так:

- Выбираем активный вход: `TIMx_CCR1` подключаем в `TI1` и пишем "`01`" в биты `CC1S` регистра `TIMx_CCMR1`. Биты `CC1S` отличаются от `00`, т. е. канал на вводе и регистр `TIMx_CCR1` доступен только по чтению.

- Ставим длительность входного фильтра битами ICxF в регистре TIMx_CCMRx с учётом входного сигнала на TIx. Допустим, что при переключении входной сигнал нестабилен в течении пяти внутренних тактов. Значит длительность фильтра должна быть больше этих пяти тактов. Мы можем определить передачу на TI1 за 8 последовательных выборок нового уровня на частоте f_{DTS}. И мы пишем 0011 в биты IC1F регистра TIMx_CCMR1.
- Выбираем фронт сигнала на TI1 записью 0 в бит CC1P регистра TIMx_CCER (передний).
- Ставим предделитель. Здесь он нам не нужен (пишем 00 в биты IC1PS регистра TIMx_CCMR1).
- Разрешаем запись счётчика в регистр захвата установкой бита CC1E в TIMx_CCER.
- Если нужно разрешаем выдачу запросов прерывания битом CC1IE в TIMx_DIER и DMA битом CC1DE в регистре TIMx_DIER.

Если захват входа произошёл, то:

- Регистр TIMx_CCR1 получает значение счётчика активной передачи.
- Ставится флаг прерывания CC1IF. Если он ещё стоял, то ставится и флаг CC1OF.
- В зависимости от бита CC1IE генерируется прерывание.
- В зависимости от бита CC1DE генерируется запрос DMA.

Чтобы обработать перезахват рекомендуется читать регистр данных до опроса флага перезахвата.

NB: Запросы прерывания и DMA IC можно выдавать программно установкой битов CCxG в регистре TIMx_EGR.

17.3.7. Режим ввода ШИМ

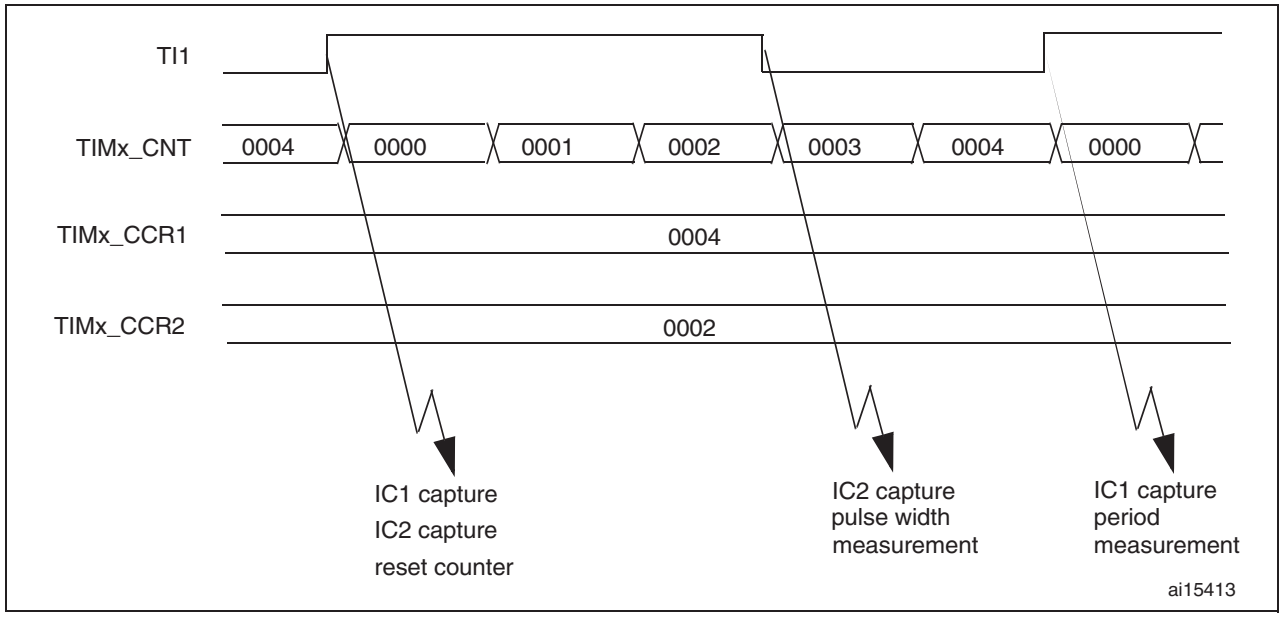
Это часть режима захвата входа. Процедура такая же, кроме:

- Два сигнала ICx направляются на на один вход TIx.
- Полярность этих входов ICx противоположная.
- Сигналом запуска берётся один из двух сигналов TIxFP, а контроллер ведомого режима выключается.

Например, можно измерить период (в регистре TIMx_CCR1) и длительность заполнения (в регистре TIMx_CCR2) ШИМ сигнала на TI1 следующей процедурой (зависит от частоты CK_INT и предделителя):

- Выбрать активный вход для TIMx_CCR1: Записать 01 в биты CC1S регистра TIMx_CCMR1 (TI1).
- Выбрать полярность для TI1FP1 (для захвата в TIMx_CCR1 и очистки счётчика): записать 0 в CC1P (передний фронт).
- Выбрать активный вход для TIMx_CCR2: записать 10 в биты CC2S регистра TIMx_CCMR1 (TI1).
- Выбрать полярность для TI1FP2 (для захвата в TIMx_CCR2): записать 1 в CC2P (задний фронт).
- Выбрать вход запуска: записать 101 в биты TS регистра TIMx_SMCR (TI1FP1).
- Поставить контроллер режима ведомого в сброс: записать 100 в биты SMS регистра TIMx_SMCR.
- Включить захват: поставить биты CC1E и CC2E в регистре TIMx_CCER.

Рис. 116. Времянка режима ввода ШИМ.



1. Режим ШИМ может использоваться только с сигналами TIMx_CH1/TIMx_CH2 так как к контроллеру режима ведомого подключены только TI1FP1 и TI2FP2.

17.3.8. Принудительный вывод

В режиме вывода (биты CCxS = 00 в TIMx_CCMRx), каждый выходной сигнал сравнения (OCxREF и затем OCx/OCxN) можно программно поставить в активное или неактивное состояние, независимо от результата сравнения выходного регистра сравнения и счётчика.

Активными выходные сигналы (OCxREF/OCx) ставятся записью 101 в биты OCxM нужного регистра TIMx_CCMRx. Активный сигнал OCxREF всегда высокий, а OCx противоположен биту полярности CCxP.

Сигнал OCxREF снимается записью 100 в биты OCxM регистра TIMx_CCMRx.

Но сравнение внутреннего регистра TIMx_CCRx со счётчиком всё равно выполняется, биты ставятся, генерируются запросы прерывания и DMA. См. ниже.

17.3.9. Режим сравнения выхода

Используется для управления выходным сигналом и определения истечения периода времени.

При совпадении регистра захвата/сравнения со счётчиком эта схема:

- Ставит выходную ножку в заданное состояние (биты OCxM в регистре TIMx_CCMRx) и полярность (бит CCxP в регистре TIMx_CCER). Ножка может хранить состояние(OCxM=000), стать активной (OCxM=001), неактивной (OCxM=010) или переключиться (OCxM=011).
- Ставит флаг прерывания (бит CCxIF в регистре TIMx_SR).
- При установленной маске прерывания (бит CCxIE в регистре TIMx_DIER) выдаёт запрос.
- При установленном разрешении DMA (бит CCxDE в регистре TIMx_DIER) выдаёт нужный (бит CCDS в регистре TIMx_CR2) запрос.

В регистр TIMx_CCRx можно писать как через регистр предзагрузки, так и без него (см. бит OCxPE в регистре TIMx_CCMRx).

В режиме сравнения выхода бит события UEV на выходы OCxREF и OCx не влияет. Разрешение составляет один счёт счётчика. Этот режим можно использовать для выдачи одного импульса.

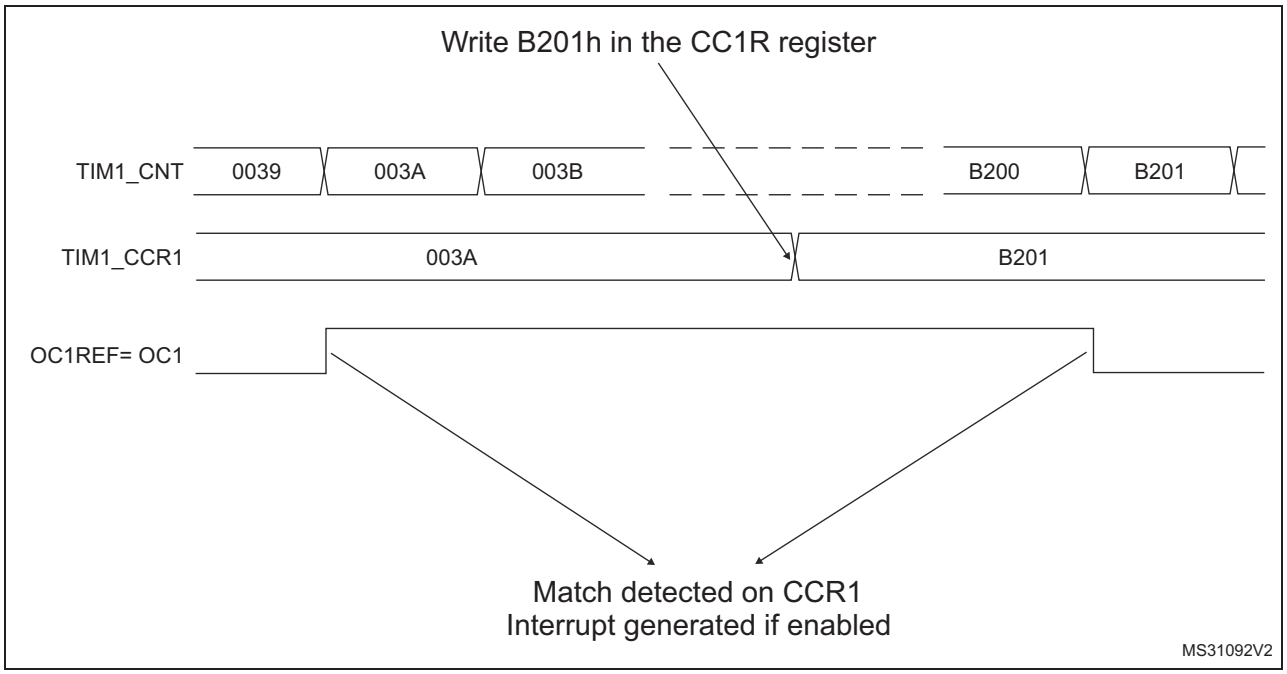
Процедура:

1. Выбрать такты счётчика (внутренние, внешние, предделитель).
2. Записать желаемое в регистры TIMx_ARR и TIMx_CCRx.
3. Если нужно прерывание, то установить бит CCxIE.

4. Выбрать режим выхода. Например:
   - Записать OCxM = 011 для переключения ножки OCx при совпадении CNT и CCRx
   - Записать OCxPE = 0 для отключения регистра предзагрузки
   - Записать CCxP = 0 ради высокого активного уровня
   - Записать CCxE = 1 для разрешения вывода
5. Включить счётчик установкой бита CEN в регистре TIMx_CR1.

При запрещённом регистре предзагрузки (OCxPE='0') регистр TIMx_CCRx можно обновлять в любое время, иначе он будет обновлён только по событию UEV.

Рис. 117. Режим сравнения выхода, переключение ножки OC1.



17.3.10.Режим ШИМ

Частота сигнала ШИМ определяется регистром TIMx_ARR, а коэффициент заполнения регистром TIMx_CCRx.

Режим ШИМ может включаться независимо по одному на каждый выход OCx записью '110' (ШИМ1) или '111' (ШИМ2) в биты OCxM регистра TIMx_CCMRx. Соответствующий регистр предзагрузки должно включать битом OCxPE в регистре TIMx_CCMRx и, соответсвенно, регистр авто-загрузки (в режимах прямого и реверсивного счёта) нужно включить битом ARPE в регистре TIMx_CR1.

Поскольку регистр предзагрузки пишется в теневой регистр только по событию обновления, то все регистры нужно заранее инициализировать установкой бита UG в регистре TIMx_EGR.

Полярность OCx (активный высокий или низкий) выбирается битом CCxP в регистре TIMx_CCER. Выходы OCx включаются битами CCxE, CCxNE, MOE, OSSI и OSSR в регистрах TIMx_CCER и TIMx_BDTR.

В режиме ШИМ (1 или 2), TIMx_CNT и TIMx_CCRx сравниваются на выполнение условия TIMx_CCRx ≤ TIMx_CNT или TIMx_CNT ≤ TIMx_CCRx (в зависимости от направления счёта).

Бит CMS в регистре TIMx_CR1 определяет генерацию ШИМ по фронту или по центру.

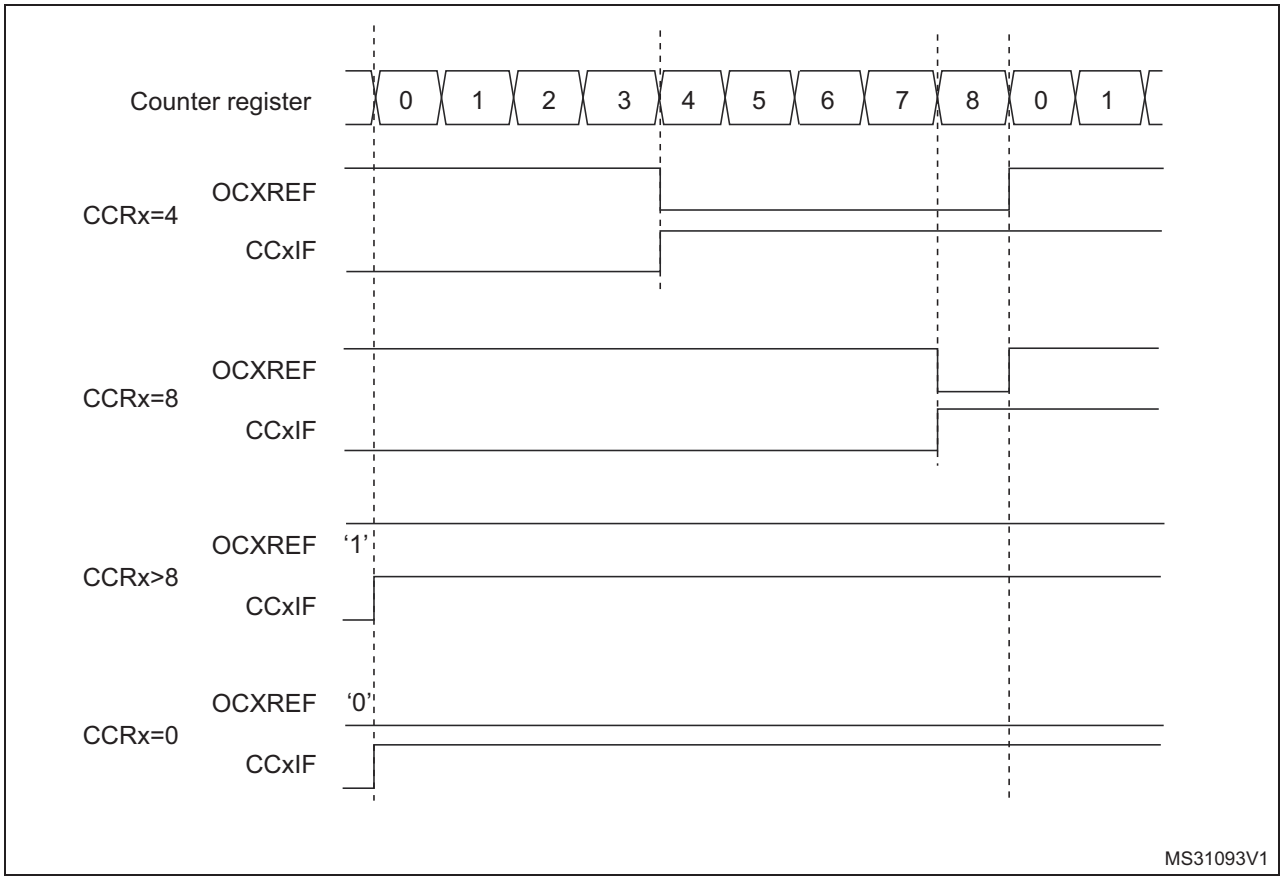
ШИМ по фронту

• Прямой счёт

Он определяется сброшенным битом DIR в регистре TIMx_CR1.

В примере ниже стоит режим ШИМ1. Опорный сигнал OCxREF остаётся высоким при TIMx_CNT < TIMx_CCRx. Если регистр TIMx_CCRx больше значения авто-загрузки (TIMx_ARR) то OCxREF равен '1'. При равенстве значений сравнения, OCxREF равен '0'. В примере ниже TIMx_ARR=8.

Рис. 118. ШИМ по фронту (TIMx_ARR=8).



- Обратный счёт

  Он определяется стоящим битом DIR в регистре TIMx_CR1.

  В режиме ШИМ1 опорный сигнал OCxREF остаётся низким при TIMx_CNT > TIMx_CCRx. Если регистр TIMx_CCRx больше значения авто-загрузки (TIMx_ARR) то OCxREF равен '1'. При равенстве значений сравнения, OCxREF равен '1'. 0% ШИМ в этом режиме недоступен.

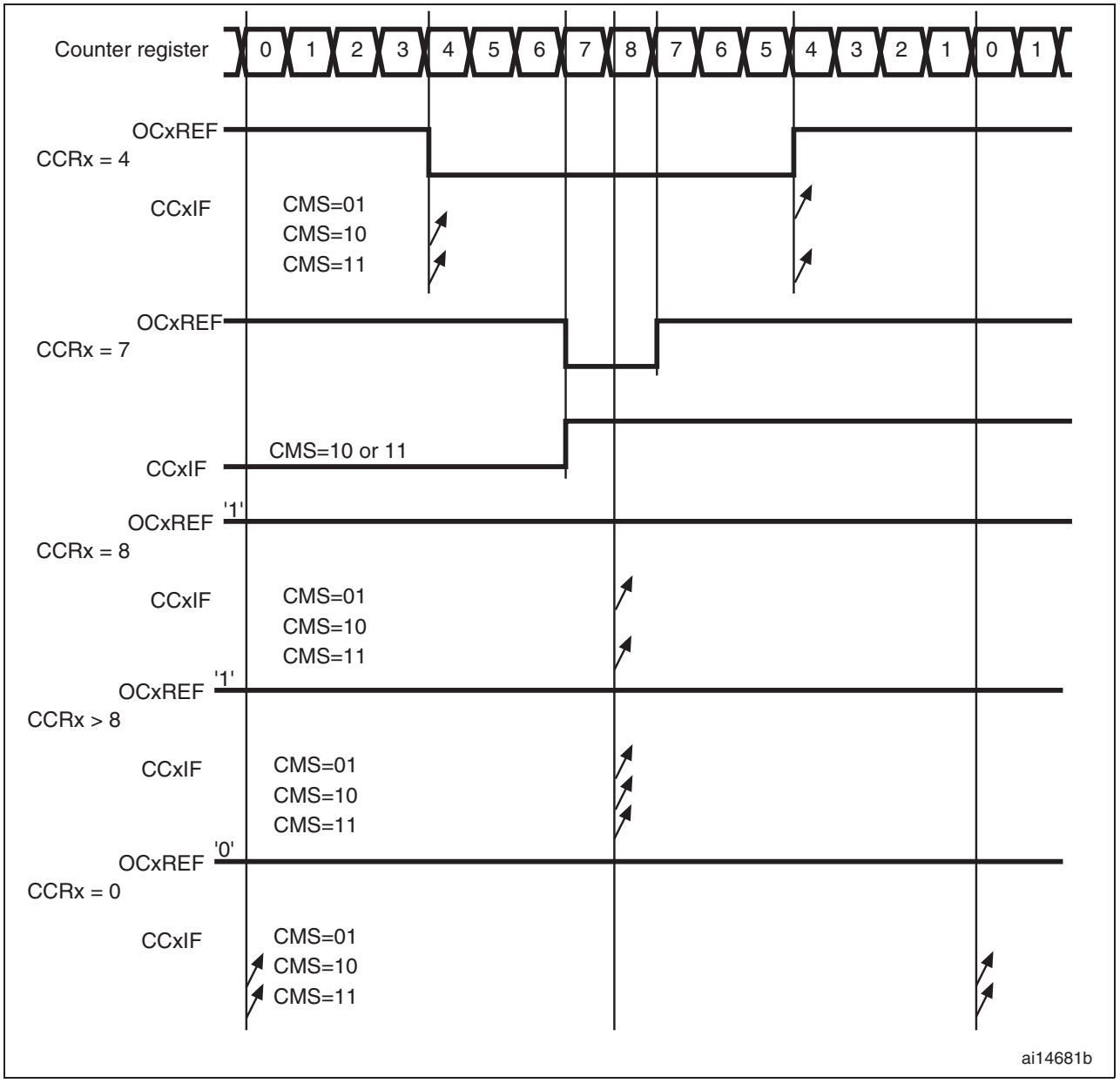
ШИМ по центру

Этот режим включается когда биты CMS в регистре TIMx_CR1 не равны '00' (остальные конфигурации на сигналы OCxREF/OCx влияют также). Флаг сравнения ставится в любых режимах счёта (прямом, обратном и реверсивном) в зависимости от битов CMS в регистре TIMx_CR1. Бит направления (DIR) в регистр TIMx_CR1 меняется аппаратно и не дай бог менять его программно.

В примере ниже:

- TIMx_ARR=8,
- Стоит режим ШИМ1,
- Флаг ставится при обратном счёте в режиме ШИМ1 по центру (CMS=01 в TIMx_CR1).

Рис. 119. ШИМ по центру (TIMx_ARR=8).



Советы:

- При старте режима по центру используется текущее направление счёта (зависит от бита DIR в регистре TIMx_CR1). Более того, в это время биты DIR и CMS программно менять нельзя.
- Запись в счётчик работающего режима ой как не рекомендуется. В частности:
  - При TIMx_CNT>TIMx_ARR направление счёта не изменяется и продолжает считать в том же направлении.
  - При записи 0 или изменении TIMx_ARR направление изменяется, но событие UEV не выдаётся.
- Самый безопасный способ это программно запустить обновление (битом UG в регистре TIMx_EGR) прямо перед стартом, а не писать в работающий счётчик.

17.3.11.Инверсные выходы и задержка

Таймеры TIM1 и TIM8 могут выдавать инверсные выходы одного сигнала с некоторой задержкой друг от друга, что могут потребовать характеристики управляемых устройств.

Полярность выходов (основного OCx и инверсного OCxN) можно выбирать независимо. Этому служат биты CCxP и CCxNP в регистре TIMx_CCER.

Сигналы OCx и OCxN активируются комбинацией битов: CCxE и CCxNE в регистре TIMx_CCER и битами MOE, OISx, OISxN, OSSI и OSSR в регистрах TIMx_BDTR и TIMx_CR2. В частности задержка включается при переходе в состояние IDLE (MOE падает в 0).

Задержка разрешается установкой обоих битов CCxE и CCxNE, и битом MOE при наличии схемы остановки. Биты DTG[7:0] регистра TIMx_BDTR определяют задержку для всех каналов. От сигнала OCxREF выдаются 2 выхода OCx и OCxN. Если OCx и OCxN активны высокими:

- Сигнал OCx повторяет опорный сигнал, но с задержкой переднего фронта.
- Сигнал OCxN это инверсный опорный сигнал, но с задержкой по противоположному фронту.

Если задержка больше ширины импульса активного выхода (OCx или OCxN), то соответствующий импульс не генерируется.

Далее показана связь генератора задержки выходных сигналов и опорным сигналом OCxREF. (В этих примерах CCxP=0, CCxNP=0, MOE=1, CCxE=1 and CCxNE=1).

Рис. 120. Инверсный выход с задержкой.

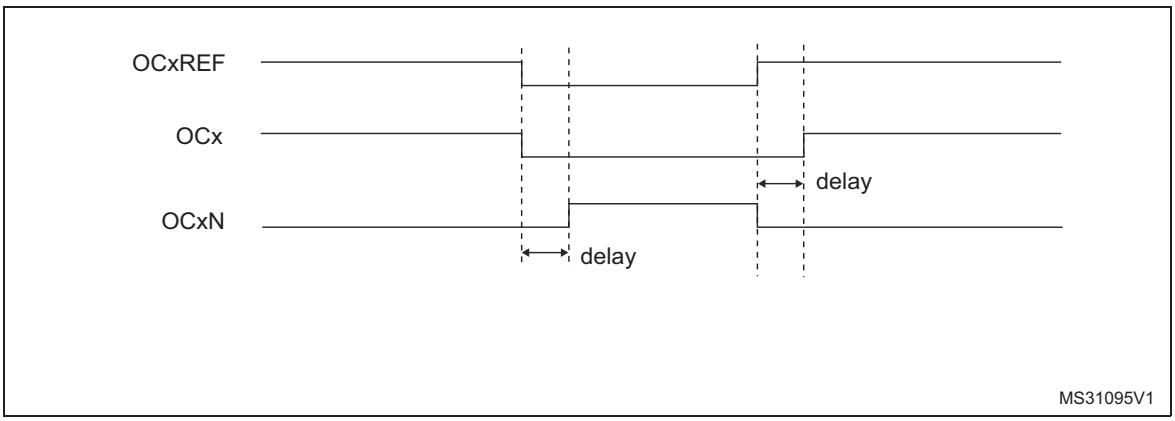


Рис. 121. Задержка больше инверсного импульса.

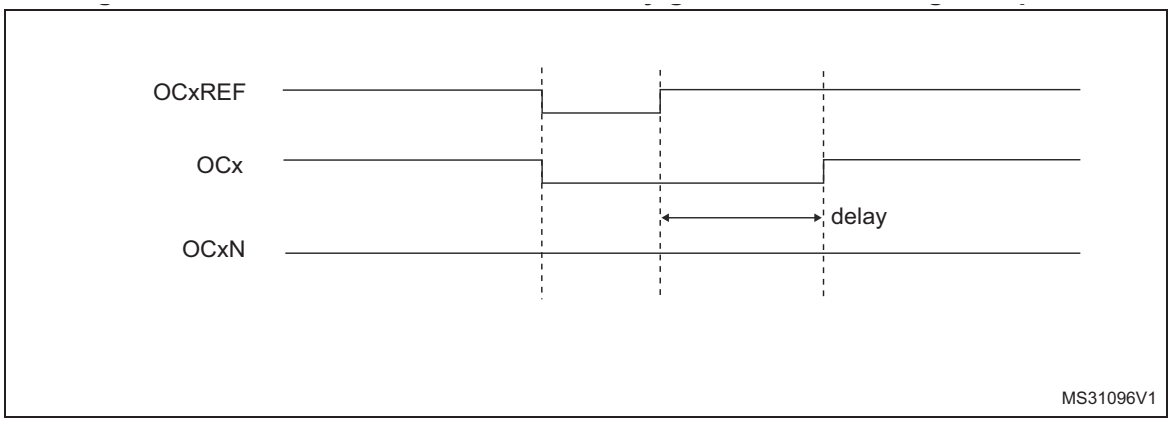
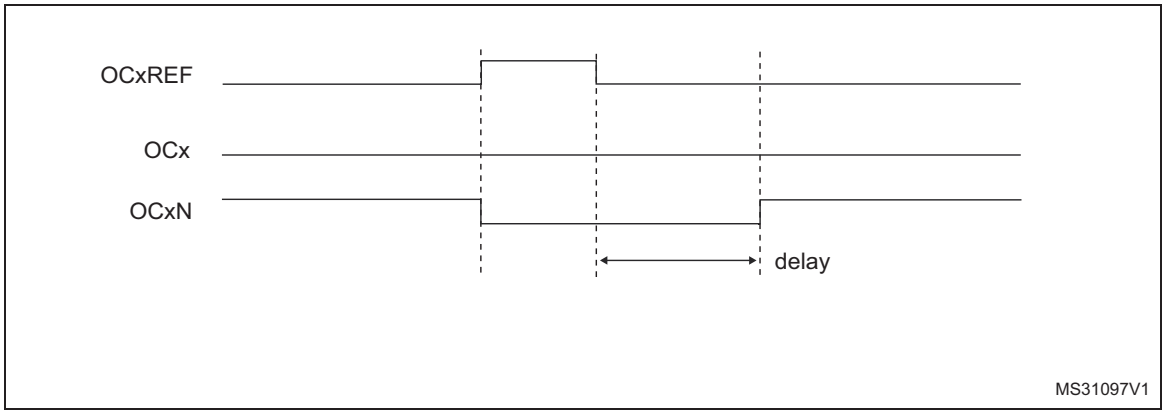


Рис. 122. Задержка больше прямого импульса.



Задержка определяется битами DTG в регистре TIMx_BDTR одинаковой для всех каналов.

Перенаправление OCxREF на OCx или OCxN

В режиме вывода (принудительном, сравнении выхода или ШИМ) OCxREF можно направлять на выходы OCx или OCxN битами CCxE и CCxNE регистра TIMx_CCER.

Так можно направить какой-либо сигнал (вроде ШИМ и пр.) на один из выходов, оставив инверсный неактивным. Другая возможность это сделать оба выхода неактивными или активными с инверсией и задержкой.

Когда разрешён только OCxN (CCxE=0, CCxNE=1), то он не инвертируется и становится активным при высоком OCxREF. Например, если CCxNP=0, то OCxN= OCxREF. С другой стороны, если разрешены оба сигнала OCx и OCxN (CCxE=CCxNE=1), то OCx становится активным при высоком OCxREF, а инверсный OCxN становится активным при низком OCxREF.

17.3.12.Функция останова

При останове разрешения вывода и неактивные уровни изменяются в соответствии с дополнительными битами управления (MOE, OSSI и OSSR в регистре TIMx_BDTR, OISx и OISxN в регистре TIMx_CR2). В любом случае, выходы OCx и OCxN не могут стать активными одновременно.

Сигнал останова может поступать с внешней ножки или по сбою тактирования от Системы безопасности тактов (CSS).

При выходе из сброса схема останова выключена и бит MOE сброшен. Включается схема останова установкой бита BKE в регистре TIMx_BDTR. Полярность сигнала сброса выбирается битом BKP там же. BKE и BKP можно изменять одновременно. Записанные BKE и BKP работают через 1 такт APB. Следовательно, их чтение истинно через 1 такт APB после записи.

Поскольку задний фронт MOE может быть асинхронным, то между действительным сигналом и синхронным битом в регистре TIMx_BDTR вставлена схема ресинхронизации. Так появляется задержка между синхронным и асинхронным сигналами. В частности, если MOE изменился с 0 на 1, то перед чтением должна быть выполнена пустая команда, поскольку запись асинхронна, а чтение синхронно.

При появлении сигнала останова:

- Бит MOE асинхронно очищается, переключая выходы в неактивное, пустое или сброшенное состояние (выбирается битом OSSI). Это работает даже при выключенном генераторе MCU.
- Все выходные каналы получают уровни, определённые битами OISx в регистре TIMx_CR2 сразу после сброса MOE. Если OSSI=0, то таймер освобождает включённые выходы, иначе они остаются высокими.
- Если используются инверсные выходы:
  - Сначала выходы переходят в неактивное состояние сброса (зависит от полярности). Это действие асинхронно и работает даже при выключенном тактировании.
  - Если тактирование ещё есть, то запускается генератор задержки перевода выходов на уровни, определённые битами OISx и OISxN. Даже в этом случае OCx и OCxN не могут стать активными одновременно. Из-за схемы ресинхронизации MOE, длительность задержки чуточку больше (около 2 ck_tim тактов).
  - Если OSSI=0, то таймер освобождает включённые выходы, иначе они остаются высокими если один из битов CCxE или CCxNE высокий.
- Ставится флаг сброса (BIFв регистре TIMx_SR). Если бит BIE в регистре TIMx_DIER стоит, то генерируется прерывание. Запрос DMA выдаётся если стоит бит BDEв регистр TIMx_DIER.
- Если бит AOE в TIMx_BDTR стоит, то бит MOE автоматически ставится снова по следующему событию UEV. Иначе MOE остаётся низким до следующей записи ‘1’.

NB: Входы останова действуют по уровню. Так что, MOE не ставится при активном входе (ни аппаратно, ни программно). Ну и флаг BIF не сбрасывается.

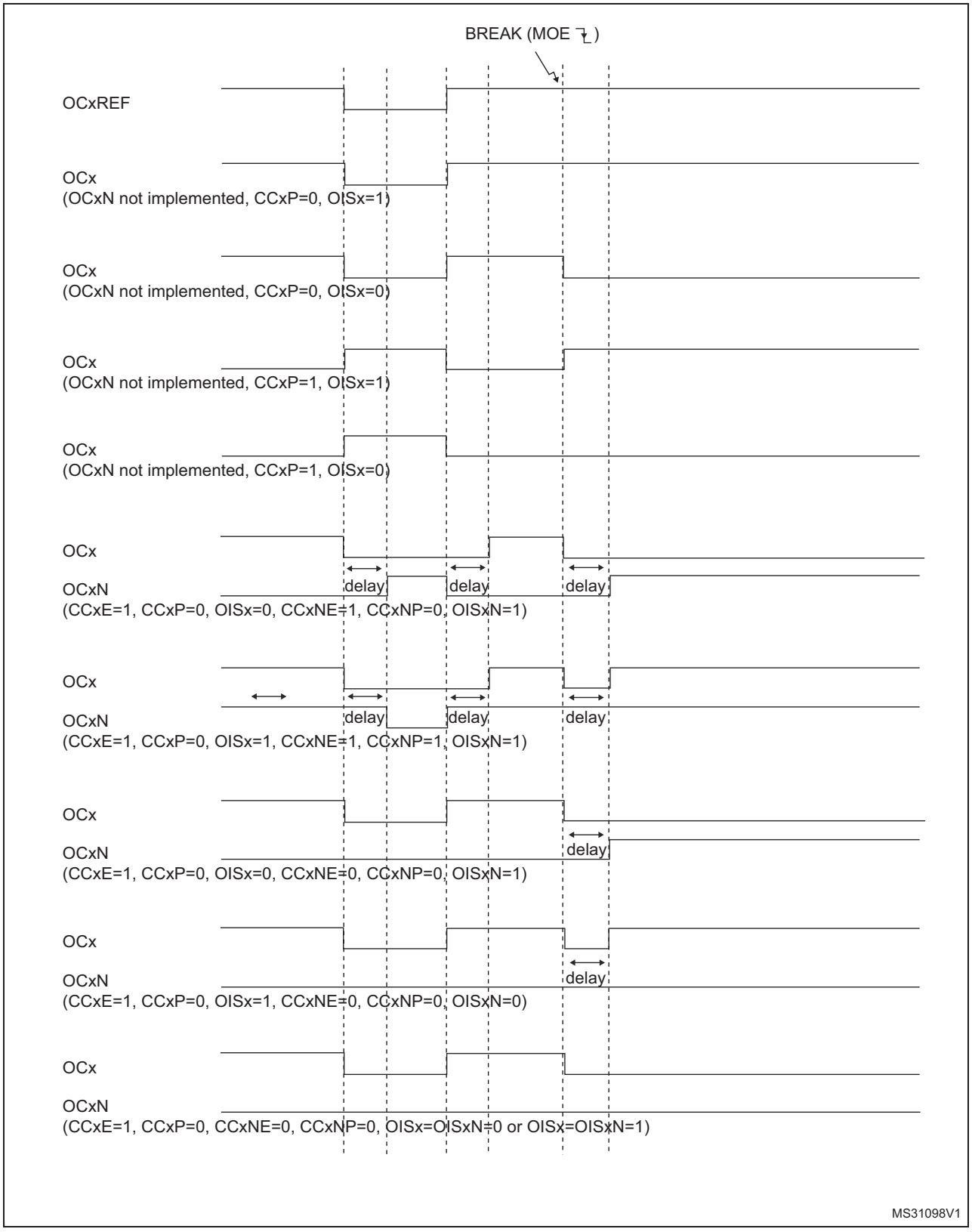
Останов можно генерировать по входу BRK с заданием полярности и битом разрешения BKE в регистре TIMx_BDTR.

Два способа генерации останова:

- По входу BRK с заданием полярности и битом разрешения BKE в регистре TIMx_BDTR.
- Программно битом BG в регистре TIMx_EGR.

Кроме того, в схеме останова реализована защита записи, позволяющая заморозить несколько параметров (длительность задержки, полярность и состояние выключенных OCx/OCxN, конфигурацию OCxM, разрешение и полярность останова). Есть три уровня защиты битами LOCK в регистре TIMx_BDTR. Биты LOCK можно писать единожды после сброса MCU.

Рис. 123. Выходные сигналы при останове.



17.3.13.Очистка OCxREF по внешнему событию

Сигнал OCxREF для заданных каналов можно сделать низким, подав высокий уровень на вход ETRF (бит разрешения OCxCE в TIMx_CCMRx стоит в ‘1’). Он остаётся низким вплоть до следующего события UEV.

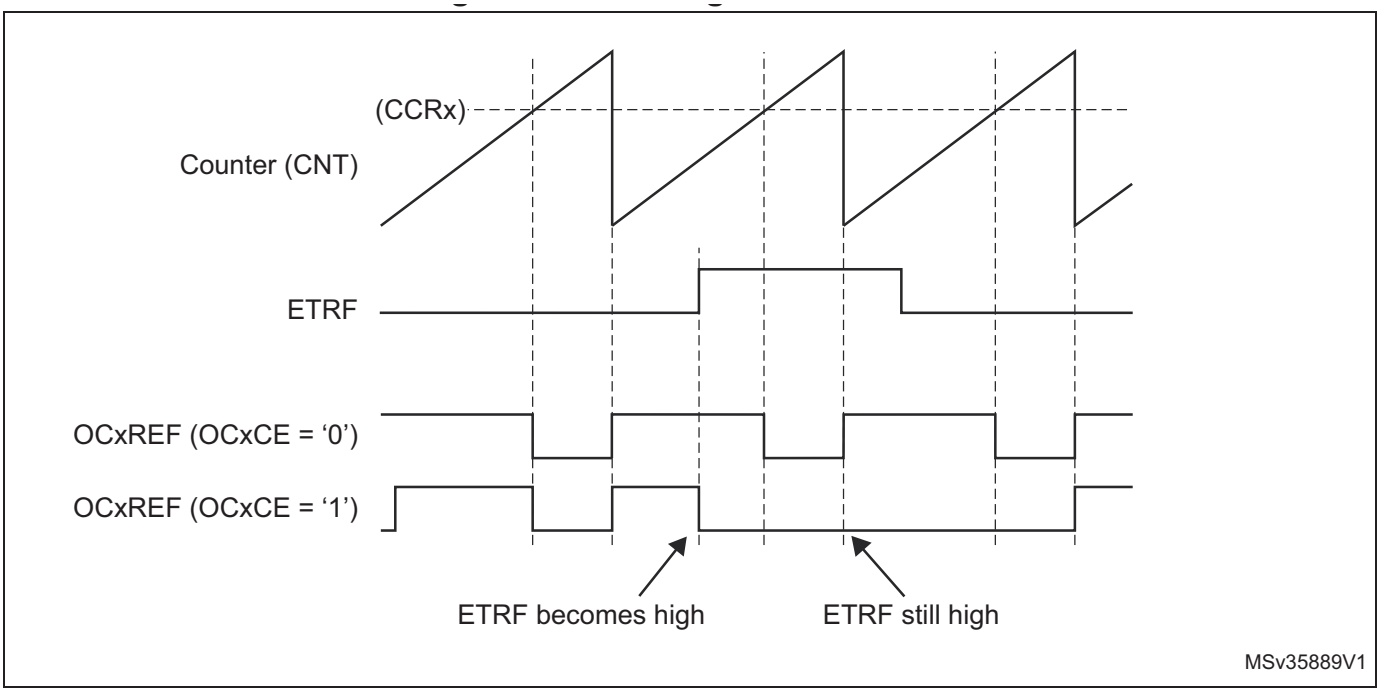
Эта функция работает в режимах сравнения выхода и ШИМ, но не в принудительном режиме.

Например, сигнал ETR можно подключить к к выходу компаратора для управления током. В этом случае ETR нужно конфигурировать так:

1. Выключить предделитель внешнего запуска: записать ’00’ в биты ETPS[1:0] в TIMx_SMCR.
2. Выключить режим 2 внешнего тактирования: снять бит ECE в TIMx_SMCR.
3. Поставить полярность и фильтр внешнего запуска (ETP и ETF) как нужно.

В примере ниже приведён сигнал OCxREF при переходе входа ETRF в высокий уровень для обоих значений OCxCE. Здесь TIMx в режиме ШИМ.

Рис. 124. Очистка OCxREF.



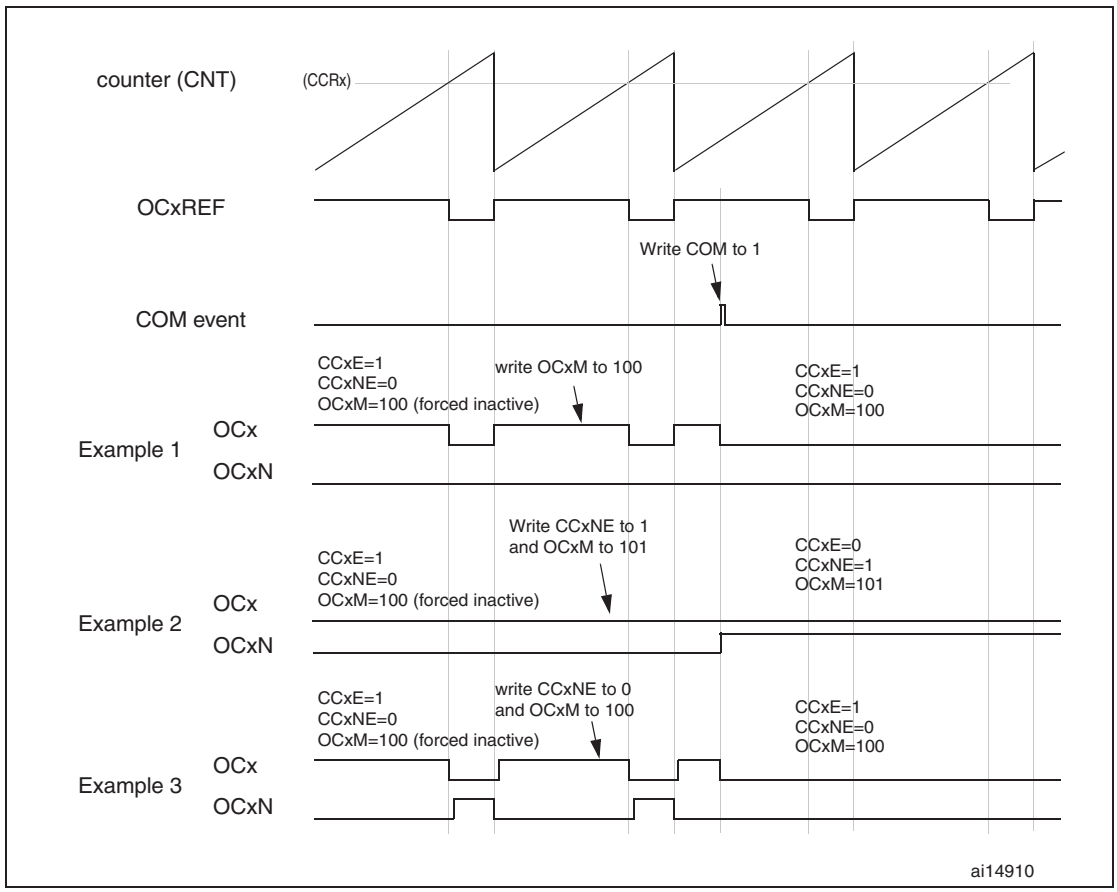
17.3.14.Шести-шаговая ШИМ

При работе с инверсными выходами доступна предзагрузка битов OCxM, CCxE и CCxNE. Они пишутся в теневые биты по событию коммутации COM. Так можно заранее создать конфигурацию и одновременно включить её для всех каналов. COM можно генерировать программно установкой бита COM в регистре TIMx_EGR или аппаратно (по переднему фронту TRGI).

При появлении события COM ставится флаг (COMIF в регистре TIMx_SR), который может вызвать прерывание (если стоит бит COMIE в регистре TIMx_DIER) или запрос DMA (если стоит бит COMDE в регистре TIMx_DIER).

Ниже приведены выходы OCx и OCxN при событии COM в 3 конфигурациях.

Рис. 125. Пример COM, (OSSR=1).



17.3.15.Режим одного импульса

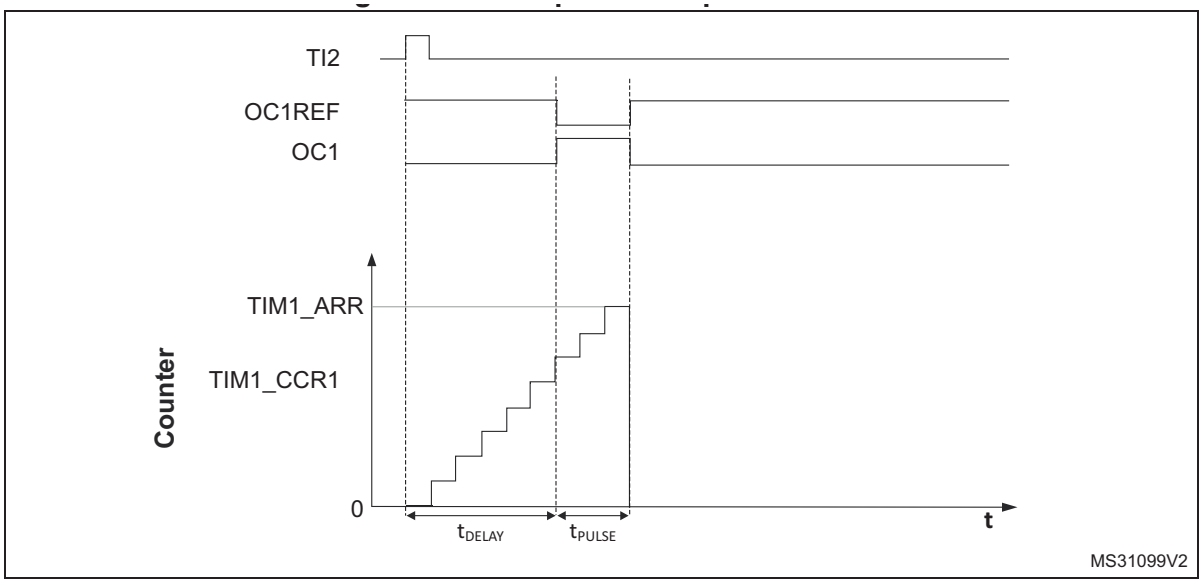
OPM это частный случай предыдущих режимов. Он позволяет счётчику запускаться по сигналу и генерировать импульс программируемой длины с программируемой задержкой.

Запускать счётчик можно из контроллера ведомого режима. Создавать сигнал можно в режиме сравнения выхода и ШИМ. Режим включается установкой бита `OPM` в регистре `TIMx_CR1`. Так счётчик автоматически останавливается по следующему событию `UEV`.

Импульс генерируется правильно только если значение сравнения отличается от начального значения счётчика. Перед стартом (таймер ждёт запуска) конфигурация должна быть:

- При прямом счёте: `CNT` < `CCRx` ≤ `ARR` (в частности, 0 < `CCRx`)
- При обратном счёте: `CNT` > `CCRx`

Рис. 126. Режим одного импульса.



Например, выдадим положительный импульс на OC1 длиной t_{PULSE} после задержки t_{DELAY} по переднему фронту на ножке TI2.

Берём TI2FP2 как запуск 1:

- Поставим TI2FP2 на TI2 записью CC2S=’01’ в регистре TIMx_CCMR1.
- TI2FP2 ставим на передний фронт записью CC2P=’0’ в регистре TIMx_CCER.
- Выбираем TI2FP2 как запуск контроллером режима ведомого (TRGI) записью TS=’110’ в регистр TIMx_SMCR.
- TI2FP2 используем как старт записью SMS=‘110’ в TIMx_SMCR (режим запуска).

Сигнал OPM определяем записью регистров сравнения (учитываем частоту тактов и предделитель).

- Определяем t_{DELAY} записью в TIMx_CCR1.
- Определяем t_{PULSE} как разность значений перезагрузки и сравнения (TIMx_ARR - TIMx_CCR1).
- Сигнал создаётся переходом из ‘0’ в ‘1’ при сравнении и переходом из ‘1’ в ‘0’ при достижении счётчиком значения перезагрузки. Для этого, включаем режим ШИМ2 записью OC1M=111 в регистре TIMx_CCMR1. Регистры предзагрузки можно включить записью OC1PE=’1’ в регистре TIMx_CCMR1и ARPE в регистре TIMx_CR1, а можно не включать. В этом случае значение сравнения пишется в регистр TIMx_CCR1, значение перезагрузки в регистр TIMx_ARR, даём событие обновления битом UG и ждём внешнего запуска на TI2. В CC1P здесь пишется ‘0’.

В этом примере биты DIR и CMS регистра TIMx_CR1 должны быть сброшены.

Мы ждём только одного импульса (Однократный режим), так что для остановки счётчика по следующему событию обновления (счётчик переходит через значение перезагрузки и обнуляется) нужно поставить бит OPM регистра TIMx_CR1. Если бит OPM регистра TIMx_CR1 обнулён, то включается Повторяющийся режим.

Частный случай: быстрое включение OCx:

В этом режиме появление фронта на входе TIx ставит бит CEN, чем включает счётчик. Затем совпадение счётчика с значением сравнения переключает выходной сигнал. Но для этого нужны несколько тактов, ограничивая минимально возможное значение t_{DELAY}.

Если нужен сигнал с минимальной задержкой, то нужно ставить бит OCxFE в TIMx_CCMRx. Далее включаются OCxREF (и OCx) не глядя на сравнение. Его новый уровень равен уровню при сравнении. OCxFE работает только в режимах ШИМ1 и ШИМ2.

17.3.16.Режим интерфейса кодера

Режим интерфейса кодера выбирается записью SMS=‘001’ в TIMx_SMCR при счёте по фронтам TI2, SMS=’010’ по фронтам TI1 и SMS=’011’ по фронтам TI1 и TI2.

Полярность TI1 и TI2 выбирается битами CC1P и CC2P в регистре TIMx_CCER. Если нужно, то можно использовать входной фильтр.

Для работы с инкрементным кодером используются входы TI1 и TI2. Счётчик тактируется каждой достоверной передачей по TI1FP1 или TI2FP2 (TI1 и TI2 после входного фильтра и выбора полярности, без них TI1FP1= TI1 и TI2FP2= TI2) при стоящем бите CEN в TIMx_CR1. Последовательность сигналов с двух входов создаёт импульсы счёта и сигнал направления. Счёт может быть прямым и обратным, бит DIR в TIMx_CR1 изменяется аппаратно. Бит DIR вычисляется на каждый входной сигнал (TI1 и/или TI2).

Режим интерфейса кодера просто работает по внешним тактам с выбором направления. То есть счётчик непрерывно считает туда-сюда между 0 и перезагружаемым значением в регистре TIMx_ARR. Так что TIMx_ARR нужно писать до старта. Таким способом захват, сравнение, предделитель и запуск вывода продолжают нормально. Режим кодера и Режим внешних тактов 2 несовместимы.

В этом режиме счётчик автоматически следует за скоростью и направлением инкрементного кодера и его содержимое всегда представляет позицию кодера. Направление счёта соответствует направлению вращения подключённого датчика.

Далее приведены возможные комбинации, полагая, что TI1 и TI2 не изменяются одновременно.

Таблица 93. Направления счёта.

Активный сигнал	Уровень оппозитного сигнала (TI1FP1 для TI2, TI2FP2 для TI1)	Фронт TI1FP1		Фронт TI2FP2	
		Передний	Задний	Передний	Задний
Только TI1	Высокий	Обратный	Прямой	Стоит	Стоит
	Низкий	Прямой	Обратный	Стоит	Стоит
Только TI2	Высокий	Стоит	Стоит	Прямой	Обратный
	Низкий	Стоит	Стоит	Обратный	Прямой
TI1 и TI2	Высокий	Обратный	Прямой	Прямой	Обратный
	Низкий	Прямой	Обратный	Обратный	Прямой

Внешний инкрементный кодер может напрямую подключаться к MCU без дополнительной логики. Но обычно для преобразования дифференциального сигнала в цифровой используются компараторы. Это сильно увеличивает помехозащищённость. Третий выход кодера, показывающий механическую нулевую позицию, можно подключить к входу внешнего прерывания и сбросу счётчика.

Пример работы счётчика показывает генерацию сигнала счёта и направления. Также показано как компенсируется дребезг на обоих выбранных фронтах. Он может появиться при нахождении датчика возле точек переключения. Использована такая конфигурация:

- CC1S=’01’ (TIMx_CCMR1, TI1FP1 на TI1).
- CC2S=’01’ (TIMx_CCMR2, TI1FP2 на TI2).
- CC1P=’0’, and IC1F = ‘0000’ (TIMx_CCER, TI1FP1 без инверсии, TI1FP1=TI1).
- CC2P=’0’, and IC2F = ‘0000’ (TIMx_CCER, TI1FP2 без инверсии, TI1FP2= TI2).
- SMS=’011’ (TIMx_SMCR, оба входа по обоим фронтам).
- CEN=’1’ (TIMx_CR1, счётчик включён).

Рис. 127. Работа счётчика.

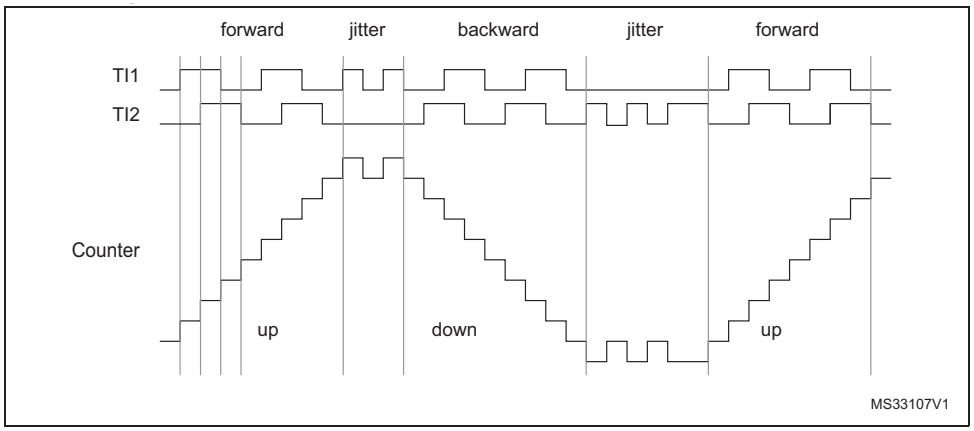
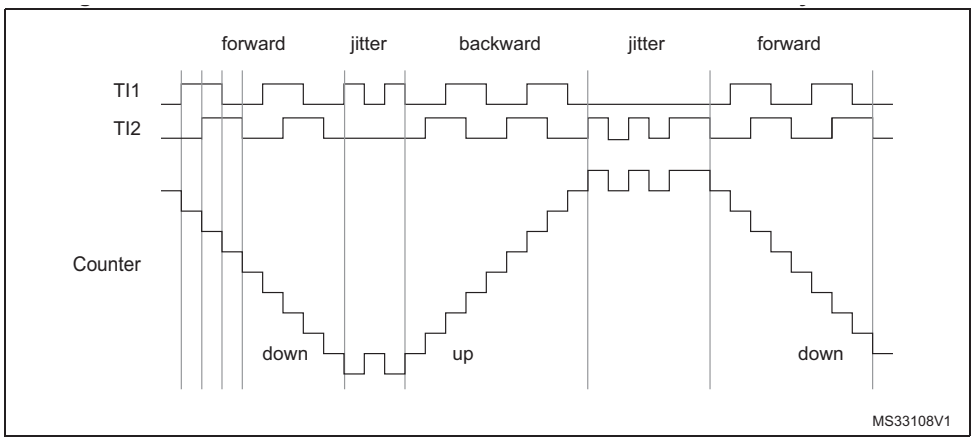


Рис. 128. Работа счётчика с инверсным TI1FP1 и CC1P=’1’.



Таймер в режиме интерфейса кодера показывает текущую позицию счётчика. С помощью второго таймера в режиме захвата, измеряя время между двумя событиями кодера, можно получить динамическую информацию (скорость, ускорение, торможение). Для этого можно использовать третий выход кодера, показывающего механический ноль. В зависимости от времени между двумя событиями счётчик можно читать и регулярно. Это делается регулярной записью счётчика в третий входной регистр (если есть) периодическим сигналом от другого таймера. Можно читать и запросом DMA от часов реального времени.

17.3.17.Функция XOR входов таймера

Бит TI1S в регистре TIMx_CR2 позволяет подключать к входу канала 1 выход вентиля XOR, сочетая три входа TIMx_CH1, TIMx_CH2 and TIMx_CH3.

Выход XOR можно использовать со всеми входными функциями таймера, вроде запуска или захвата.

17.3.18.Работа с датчиками Холла

Это выполняется генерацией ШИМ таймером TIM1 или TIM8 и другим таймером TIMx (TIM2, TIM3, TIM4 или TIM5), так сказать “интерфейсным таймером”. TIMx захватывает 3 соединённые через XOR (бит TI1S в TIMx_CR2) входные ножки (TIMx_CH1, TIMx_CH2, and TIMx_CH3) на вход TI1.

Контроллер режима ведомого ставится в сброс; вход ведомого TI1F_ED. Так что при переключении одного из 3 входов счётчик начинает с 0. Этим создаётся временная база, запущенная любым изменением на входах Холла.

На “интерфейсном таймере” канал 1 ставится на захват, сигнал захвата TRC. Захваченное значение, соответствующее времени между 2 изменениями входных сигналов, показывает скорость мотора.

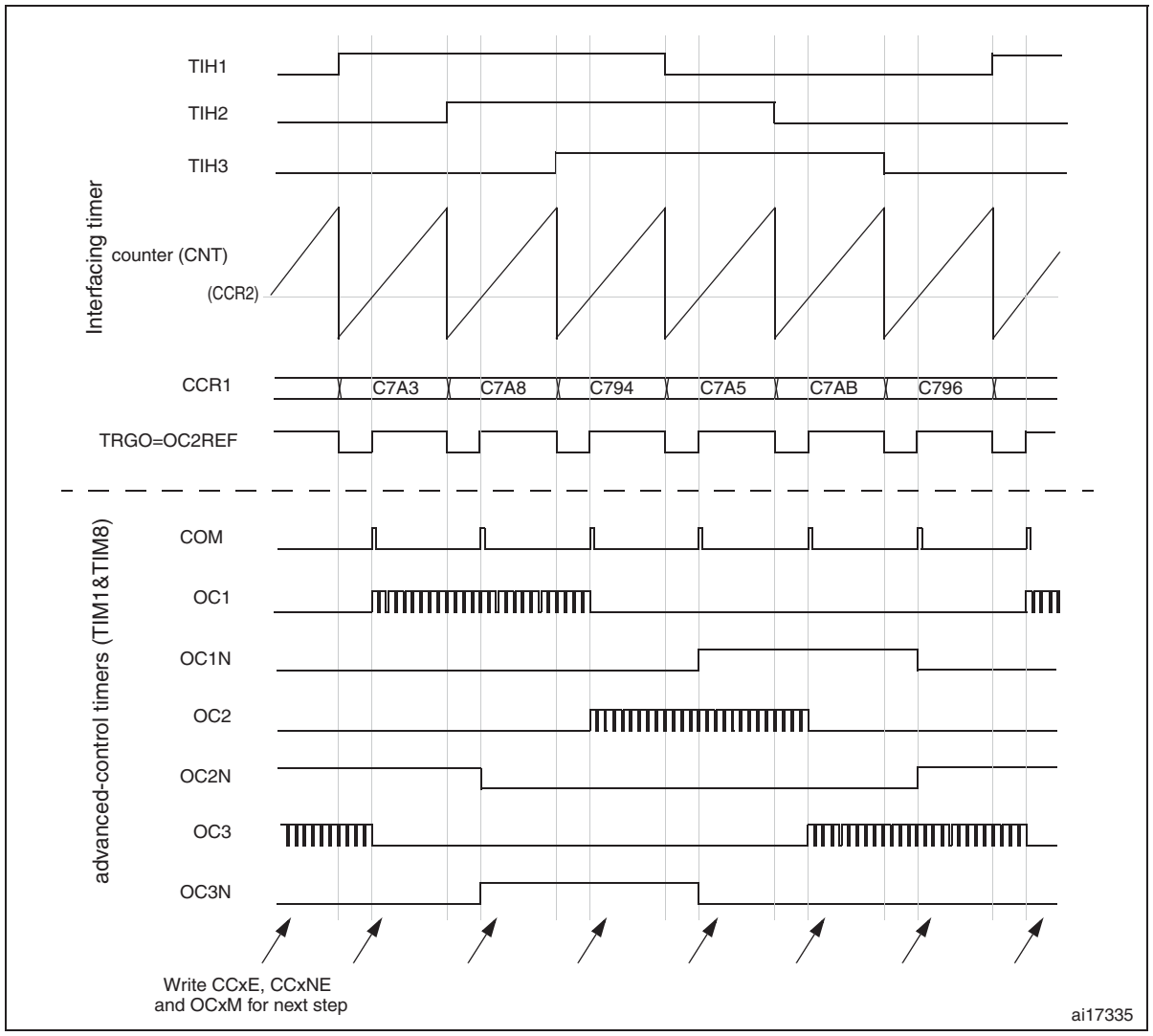
“Интерфейсный таймер” может в режиме выхода выдавать импульс изменения конфигурации продвинутого таймера (TIM1 или TIM8) (событием COM). TIM1 выдаёт ШИМ сигнал для управления мотором. Для этого канал интерфейсного таймера после программируемой задержки должен выдавать положительный импульс (в режиме сравнения выхода или ШИМ). Этот импульс через выход TRGO подаётся на продвинутый таймер (TIM1 или TIM8).

Пример: надо изменить ШИМ конфигурацию TIM1 с программируемой задержкой после каждого изменения входов Холла на таймере TIMx.

- Подключить через XOR три 3 входа таймера на TI1 установкой бита TI1S в TIMx_CR2,
- Создать временную базу: записать в TIMx_ARR максимальное значение (счётчик должен сбрасываться по изменению TI1). Установить предделитель так, чтобы максимальный период счёта превышал время между 2 изменениями датчиков,
- Поставить канал 1 в режим захвата (выбран TRC): записать ‘11’ в биты CC1S регистра TIMx_CCMR1. Можно подключить и цифровой фильтр,
- Поставить канал 1 в режим ШИМ 2 с желаемой задержкой: записать '111’ в биты OC2M и ‘00’ в биты CC2S регистра TIMx_CCMR1,
- Для запуска по TRGO выбрать OC2REF: записать '101’ в биты MMS регистра TIMx_CR2.

В таймере TIM1 правый вход ITR должен стать входом запуска, таймер должен генерировать ШИМ, управляющие сигналы захвата/сравнения предзагружаемые (CCPC=1 в регистре TIMx_CR2) м событие COM управляется от входа запуска (CCUS=1 в регистре TIMx_CR2). Биты управления ШИМ (CCxE, OCxM) пишутся после события COM для следующего шага (это можно сделать в обработчике прерывания по переднему фронту OC2REF).

Рис. 129. Пример интерфейса датчиков Холла.



17.3.19.TIMx и синхронизация внешнего запуска

Есть три режима синхронизации TIMx: Сброс, Вентильный и Запуск.

Режим ведомого: Сброс

Счётчик и предделитель можно инициализировать по входу запуска. Кроме того, если бит URS в TIMx_CR1 сброшен, то генерируется событие UEV. Далее обновляются все предзагружаемые регистры (TIMx_ARR, TIMx_CCRx).

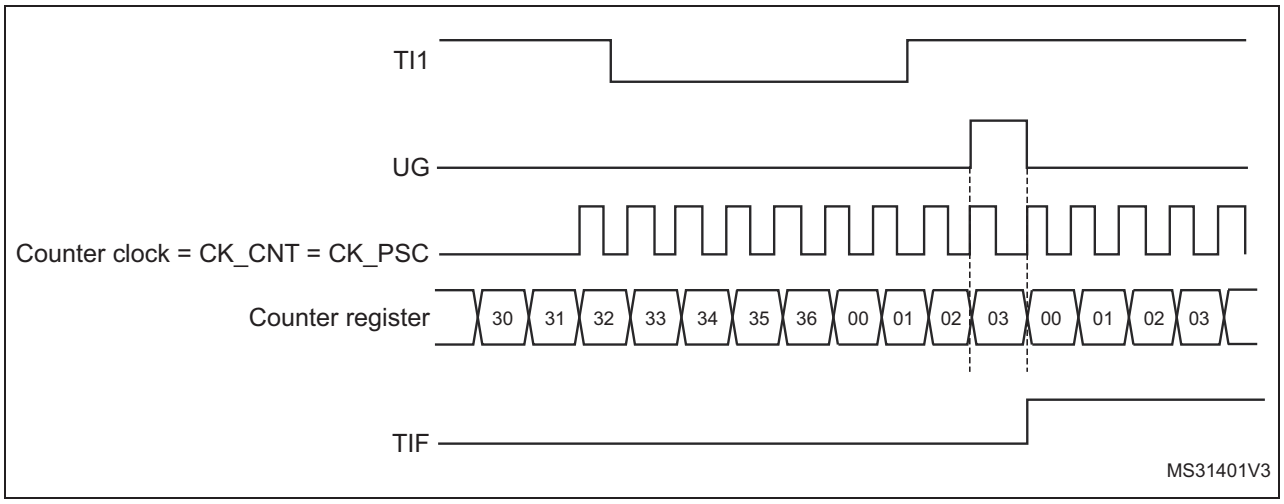
В этом примере счётчик прямого счёта сбрасывается по переднему фронту на входе TI1:

- Ставим канал 1 на передний фронт TI1. Входной фильтр отключён (IC1F=0000). Предделителя нет. Биты CC1S=01 в регистре TIMx_CCMR1 выбирают захват входа. Бит CC1P=0 в регистре TIMx_CCER определяет полярность и передний фронт.
- Ставим таймер в режим сброса записью SMS=100 в регистр TIMx_SMCR. Выбираем вход с TI1 записью TS=101 в регистре TIMx_SMCR.
- Запускаем счётчик записью CEN=1 в регистре TIMx_CR1.

Счётчик начинает считать внутренние такты вплоть переднего фронта TI1. Тогда счётчик сбрасывается с 0, ставится флаг запуска (бит TIF в регистре TIMx_SR) выдаются запросы прерывания и DMA (если разрешены битами TIE и TDE в регистре TIMx_DIER).

На рисунке ниже регистр перезагрузки TIMx_ARR=0x36. Задержка между фронтом TI1 и действительным сбросом появляется из-за схемы ресинхронизации на входе TI1.

Рис. 130. Режим Сброса.



Режим ведомого: Вентильный

Счётчик включается уровнем на выбранном входе.

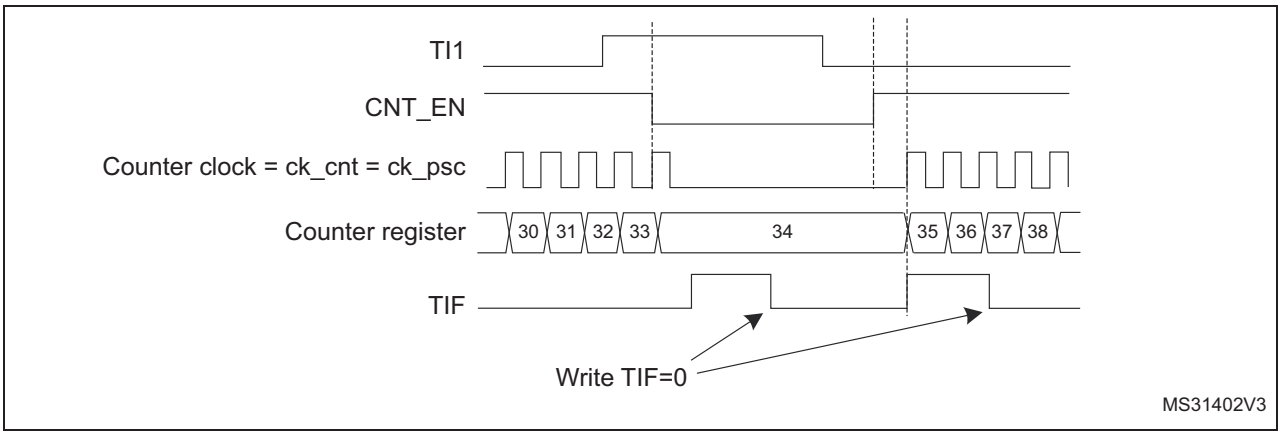
В этом примере счётчик прямого счёта работает только при низком уровне TI1:

- Ставим канал 1 на низкий уровень TI1. Входной фильтр отключён (IC1F=0000). Предделителя нет. Биты CC1S=01 в регистре TIMx_CCMR1 выбирают захват входа. Бит CC1P=1 в регистре TIMx_CCER определяет полярность и низкий уровень.
- Ставим таймер в вентильный режим записью SMS=101 в регистр TIMx_SMCR. Выбираем вход с TI1 записью TS=101 в регистре TIMx_SMCR.
- Запускаем счётчик записью CEN=1 в регистре TIMx_CR1. В этом режиме счётчик не работает при CEN=0, независимо от уровня на входе.

Счётчик начинает считать внутренние такты при низком уровне на TI1 и останавливается при высоком. Флаг запуска (бит TIF в регистре TIMx_SR) ставится при пуске и остановке таймера.

Задержка между фронтом TI1 и действительным сбросом появляется из-за схемы ресинхронизации на входе TI1.

Рис. 131. Вентильный режим.



Режим ведомого: Запуск

Счётчик включается событием на выбранном входе.

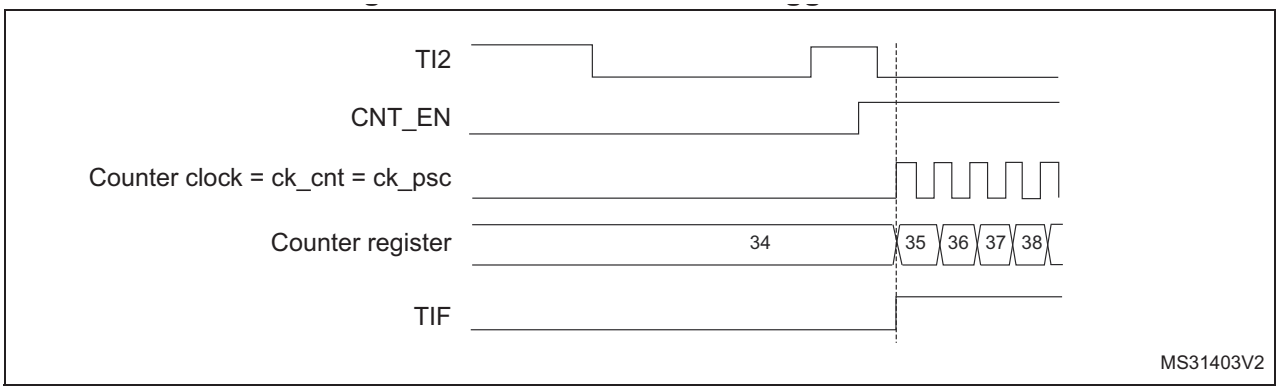
В этом примере счётчик прямого счёта работает по переднему фронту на TI2:

- Ставим канал 2 на передний фронт TI2. Входной фильтр отключён (IC1F=0000). Предделителя нет. Биты CC2S=01 в регистре TIMx_CCMR1 выбирают захват входа. Бит CC2P=1 в регистре TIMx_CCER определяет полярность и низкий уровень.
- Ставим таймер в режим запуска записью SMS=110 в регистр TIMx_SMCR. Выбираем вход с TI2 записью TS=110 в регистре TIMx_SMCR.

Счётчик начинает считать внутренние такты по переднему фронту на TI2 и ставит флаг запуска (бит TIF в регистре TIMx_SR).

Задержка между фронтом TI2 и действительным сбросом появляется из-за схемы ресинхронизации на входе TI2.

Рис. 132. Режим Запуска.



Режим ведомого: Внешние такты 2 + Запуск

К режимам ведомого (кроме Внешних тактов 1 и Кодера) можно добавить режим внешнего тактирования 2. В этом случае сигнал ETR выбирается для внешних тактов, а по другому входу можно подавать запуск (в режимах Сброса, Вентильном или Запуска). Не рекомендуем выбирать ETR как TRGI битами TS в регистре TIMx_SMCR.

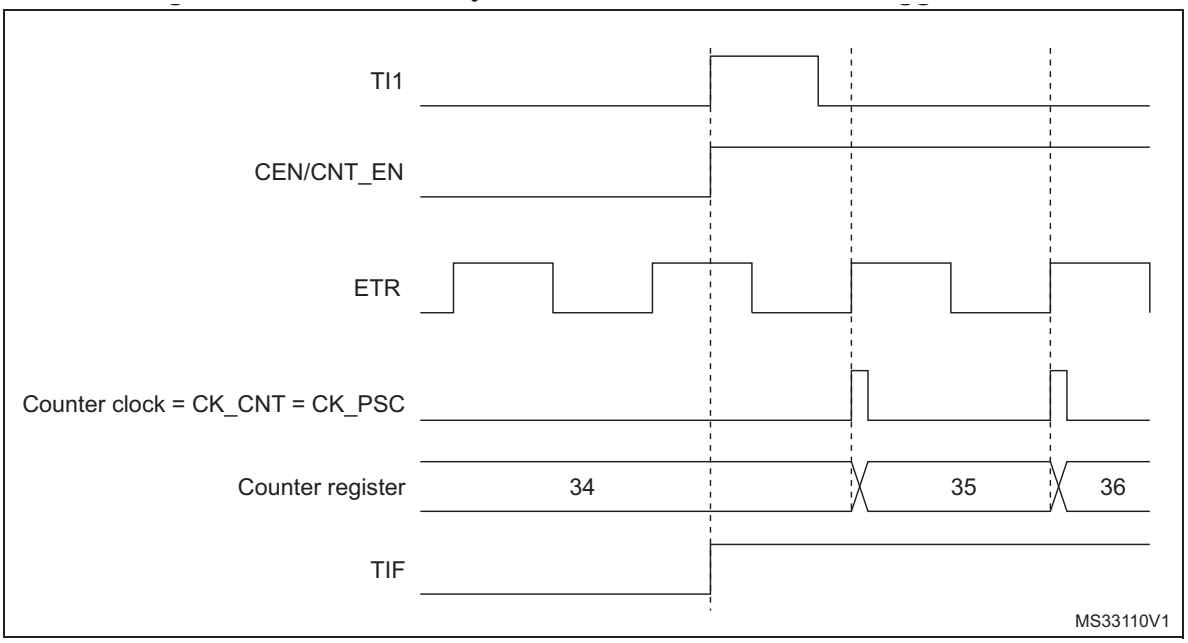
В этом примере счётчик прямого счёта работает по переднему фронту на ETR после переднего фронта запуска на TI1:

1. Ставим внешний запуск в регистре TIMx_SMCR:
   - ETF = 0000: фильтра нет; ETPS = 00: предделитель выключен; ETP = 0: передний фронт на ETR и ECE=1 для внешних тактов 2.
2. Ставим канал 1 на передний фронт TI:
   - IC1F=0000: фильтра нет; предделитель выключен, ничего не делаем; CC1S=01 в TIMx_CCMR1 для захвата входа; CC1P=0 в TIMx_CCER для полярности (и только переднего фронта).
3. Ставим таймер в режим запуска записью SMS=110 в регистр TIMx_SMCR. Выбираем вход с TI1 записью TS=101 в регистре TIMx_SMCR.

Передний фронт TI1 разрешает счёт и ставит флаг TIF. Считаются передние фронты ETR.

Задержка между фронтом ETR и действительным сбросом появляется из-за схемы ресинхронизации на входе ETRP.

Рис. 133. Внешние такты 2 + Запуск.



17.3.20.Синхронизация таймера

Таймеры TIM связаны между собой для синхронизации или сцепления.

NB: Такты ведомого таймера нужно включать до получения событий от ведущего и не должны меняться налету.

17.3.21.Режим отладки

В режиме отладки (ядро Cortex-M3 остановлено), счётчик TIMx либо работает, либо останавливается, в зависимости от бита DBG_TIMx_STOP в модуле DBG.

При стоящем счётчике (DBG_TIMx_STOP = 1 в регистре DBGMCU_APBx_FZ) выходы отключаются (как при сброшенном бите MOE). Выходы можно перевести в неактивное состояние (бит OSSI = 1), или возложить управление на контроллер GPIO (бит OSSI = 0) для перевода их в высокоимпедансное состояние (Hi-Z).

17.4. Регистры TIM1 и TIM8

Они доступны полусловами и словами.

17.4.1. Регистр управления 1 (TIMx_CR1)

Смещение адреса: 0x00

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:10 Резерв, не трогать.

— Биты 9:8 **CKD[1:0]:** Деление тактов.

Это отношение тактов таймера (CK_INT) и тактов задержки и выборки (t_{DTS}) в генераторах задержки и цифровых фильтрах (ETR, TIx),

00: $t_{DTS}=t_{CK\_INT}$

01: $t_{DTS}=2*t_{CK\_INT}$

10: $t_{DTS}=4*t_{CK\_INT}$

11: Резерв

— Бит 7 **ARPE**: Разрешение буфера перезагрузки TIMx_ARR

0: Нельзя

1: Нужно

— Бит 6:5 **CMS[1:0]**: Выбор режима по центру

00: По фронту. Прямой или обратный счёт в зависимости от бита направления (DIR).

01: По центру 1. Попеременный прямой и обратный счёт. Флаг сравнения выхода выводных каналов (CCxS=00 в TIMx_CCMRx) ставится только при обратном счёте.

10: По центру 2. Попеременный прямой и обратный счёт. Флаг сравнения выхода выводных каналов (CCxS=00 в TIMx_CCMRx) ставится только при прямом счёте.

11: По центру 3. Попеременный прямой и обратный счёт. Флаг сравнения выхода выводных каналов (CCxS=00 в TIMx_CCMRx) ставится и при прямом и при обратном счёте.

NB: При включённом таймере (CEN=1) переключать режим по фронту в режим по центру нельзя.

— Бит 4 **DIR**: Направление счёта.

0: Прямой

1: Обратный

NB: Бит читается только в режимах По центру и Кодера.

— Бит 3 **OPM**: Режим одного импульса

0: Счётчик не останавливается

1: Счётчик останавливается по следующему событию обновления (снимается бит CEN)

— Бит 2 **URS**: Источник запроса по событию UEV.

Изменяется программно.

0: Разрешённые запросы прерывания или DMA выдаются по:

– Переполнение/Исчерпание счётчика

– Установка бита UG

– Обновление от контроллера режима ведомого

1: Разрешённые запросы прерывания или DMA выдаются только по Переполнению/Исчерпанию счётчика.

— Бит 1 **UDIS:** Выключение выдачи события обновления UEV.

Изменяется программно.

0: Событие UEV выдаётся по:

– Переполнение/Исчерпание счётчика

– Установка бита UG

– Обновление от контроллера режима ведомого

1: Событие UEV не выдаётся, скрытые регистры (ARR, PSC, CCRx) хранят значение. При стоящем бите UG или аппаратном сбросе от контроллера режима ведомого счётчик и предделитель инициализируются.

— Бит 0 **CEN:** Включение счётчика.

0: Выключен

1: Включён

NB: Режимы Внешнего тактирования, Вентильный и Кодера работают только при заранее установленном бите CEN. Режим Запуска ставит его автоматически.

17.4.2. Регистр управления 2 (TIMx_CR2)

Смещение адреса: 0x04

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

— Бит 15 Резерв, не трогать.

— Бит 14 **OIS4:** Пустое состояние 4 (выход OC4) см. бит OIS1.

— Бит 13 **OIS3N**: Пустое состояние 3 (выход OC3N) см. бит OIS1.

— Бит 12 **OIS3**: Пустое состояние 3 (выход OC3) см. бит OIS1.

— Бит 11 **OIS2N**: Пустое состояние 2 (выход OC2N) см. бит OIS1.

— Бит 10 **OIS2**: Пустое состояние 2 (выход OC2) см. бит OIS1.

— Бит 9 **OIS1N**: Пустое состояние 1 (выход OC1N).

0: OC1N=0 после задержки при MOE=0

1: OC1N=1 после задержки при MOE=0

NB: Нельзя изменять при уровнях LOCK 1, 2 или 3 в регистре TIMx_BDTR.

— Бит 8 **OIS1**: Пустое состояние 1 (выход OC1).

0: OC1=0 после задержки (если есть OC1N) при MOE=0

1: OC1=1 после задержки (если есть OC1N) при MOE=0

NB: Нельзя изменять при уровнях LOCK 1, 2 или 3 в регистре TIMx_BDTR.

— Бит 7 **TI1S**: Выбор TI1.

0: Ножка TIMx_CH1 на входе TI1

1: Ножки TIMx_CH1, CH2 и CH3 на входе TI1 (через XOR)

— Бит 6:4 **MMS[1:0]**: Выбор режима ведущего

Выбор информации синхронизации ведомых таймеров от ведущего (TRGO):

000: **Сброс** - На выход запуска (TRGO) направлен бит UG регистра TIMx_EGR. Если сброс задан входом запуска (контроллер ведомого сброшен), то сигнал TRGO задержан по отношению к реальному сбросу.

001: **Разрешение** - На выход запуска (TRGO) направлен сигнал разрешения CNT_EN. Это полезно при одновременном старте нескольких таймеров или при управлении окном работы ведомого таймера. CNT_EN это объединение по OR бита CEN и входа запуска в Вентильном режиме. При подаче CNT_EN по входу запуска сигнал TRGO задерживается, кроме режима ведущий/ведомый (см. бит MSM в регистре TIMx_SMCR).

010: **Обновление** - На выход запуска (TRGO) направлено событие обновления. Например, ведущий таймер может быть предделителем ведомого.

011: **Импульс сравнения** - После захвата или совпадения выход запуска (TRGO) посылает положительный импульс установки флага CC1IF (даже если он стоит).

100: **Сравнение** - На выход запуска (TRGO) направлен сигнал OC1REF

101: **Сравнение** - На выход запуска (TRGO) направлен сигнал OC2REF

110: **Сравнение** - На выход запуска (TRGO) направлен сигнал OC3REF

111: **Сравнение** - На выход запуска (TRGO) направлен сигнал OC4REF

NB: Тактирование ведомого таймера и ADC нужно включать до получения событий от ведущего таймера и не должно меняться налету.

— Бит 3 **CCDS**: Выбор DMA Захвата/Сравнения

0: Запрос CCx DMA посылается по событию CCx

1: Запросы CCx DMA посылаются по событию обновления

— Бит 2 **CCUS**: Выбор обновления битов Захвата/Обновления

0: Предзагружаемые биты (CCPC=1) обновляются только установкой бита COMG

1: Предзагружаемые биты (CCPC=1) обновляются установкой бита COMG или передним фронтом TRGI

NB: Работает только при наличии инверсных выходов.

— Бит 1 Резерв, не трогать.

— Бит 0 **CCPS:** Предзагрузка битов Захвата/Сравнения.

0: Биты CCxE, CCxNE и OCxM не предзагружены

1: Биты CCxE, CCxNE и OCxM предзагружены, после записи они обновляются только по событию COM (бит COMG ставится по переднему фронту на TRGI, в зависимости от бита CCUS).

NB: Работает только при наличии инверсных выходов.

17.4.3. Регистр управления режима ведомого (TIMx_SMCR)

Смещение адреса: 0x08

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			Res.	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	Res.	rw	rw	rw

— Бит 15 **ETP**: Полярность внешнего запуска

0: ETR не инверсный, активен высоким или передним фронтом.

1: ETR инверсный, активен низким или задним фронтом.

— Бит 14 **ECE**: Разрешение внешнего тактирования

0: Внешнее тактирование 2 выключено

1: Внешнее тактирование 2 включено. Счётчик работает от активного фронта ETRF.

NB:

1: Установка бита ECE равнозначна Внешнему тактированию 1 с подключением TRGI к ETRF (SMS=111 и TS=111).

2: Внешнее тактирование 2 можно использовать одновременно с режимами ведомого: Сброс, Вентильный и Запуска. Тем не менее, TRGI нельзя подключать к ETRF (TS не равен 111).

3: If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.

— Биты 13:12 **ETPS[1:0]**: Предделитель внешнего запуска

Частота сигнала ETRP должна быть не больше 1/4 частоты TIMxCLK. Полезно при измерении быстрых внешних сигналов.

00: Предделитель Выкл.

01: ETRP / 2

10: ETRP / 4

11: ETRP / 8

— Биты 11:8 **ETF[3:0]**: Фильтр внешнего запуска

Определяет частоту выборки и длину цифрового фильтра сигнала ETRP. Цифровой фильтр это счётчик, выдающий сигнал на выход после N последовательных событий на входе:

0000: Без фильтра, выборка на частоте f_{DTS}

0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2

0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4

0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8

0100: $f_{SAMPLING}=f_{DTS}/2$, N=6

0101: $f_{SAMPLING}=f_{DTS}/2$, N=8

0110: $f_{SAMPLING}=f_{DTS}/4$, N=6

0111: $f_{SAMPLING}=f_{DTS}/4$, N=8

1000: $f_{SAMPLING}=f_{DTS}/8$, N=6

1001: $f_{SAMPLING}=f_{DTS}/8$, N=8

1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

— Бит 7 **MSM**: Режим Ведущий/Ведомый
0: Ничего
1: Действие входа запуска (TRGI) задерживается для полной синхронизации текущего таймера и его ведомых (через TRGO). Это полезно при синхронизации нескольких таймеров от одного внешнего события.

— Биты 6:4 **TS[2:0]**: Выбор запуска
000: Внутренний 0 (ITR0)
001: Внутренний 1 (ITR1)
010: Внутренний 2 (ITR2)
011: Внутренний 3 (ITR3)
100: Детектор фронта TI1 (TI1F_ED)
101: Фильтрованный вход 1 (TI1FP1)
110: Фильтрованный вход 2 (TI2FP2)
111: Внешний запуск (ETRF)

NB: Менять их можно только если не используются (например, при SMS=000) во избежание неверного определения фронта.

— Бит 3 Резерв, не трогать.

— Биты 2:0 **SMS[2:0]**: Выбор режима ведомого

При внешнем сигнале активный фронт TRGI подключается к внешнему входу с выбранной полярностью.
000: Режим ведомого выключен - если CEN = '1', то предделитель работает напрямую от внутренних тактов.
001: Режим кодера 1 - Прямой/обратный счёт по фронту TI2FP1 в зависимости от уровня TI1FP2.
010: Режим кодера 2 - Прямой/обратный счёт по фронту TI1FP2 в зависимости от уровня TI2FP1.
011: Режим кодера 3 - Прямой/обратный счёт по фронтам TI1FP1 и TI2FP2 в зависимости от уровня другого входа.
100: Режим сброса - Передний фронт TRGI инициализирует счётчик и запускает обновление регистров.
101: Вентильный режим - Такты счётчика разрешены при высоком TRGI. При низком TRGI счётчик останавливается, но не сбрасывается. Старт и стоп счётчика управляемые.
110: Режим запуска - Счётчик стартует по переднему фронту TRGI (но не сбрасывает). Управляемый только старт.
111: Внешние такты 1 - Счёт передних фронтов TRGI.

NB: Вентильный режим нельзя использовать если входом запуска выбран TI1F_ED (TS='100'). В действительности, TI1F_ED выдаёт 1 импульс на каждую передачу TI1F, тогда как вентильный режим проверяет уровень сигнала запуска.

Такты ведомого таймера нужно включать до приёма событий от ведущего таймера и не должны изменяться налету.

Таблица 82. Подключение внутреннего запуска TIMx.

Ведомый TIM	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM1	TIM5_TRGO	TIM2_TRGO	TIM3_TRGO	TIM4_TRGO
TIM8	TIM1_TRGO	TIM2_TRGO	TIM4_TRGO	TIM5_TRGO

17.4.4. Регистр разрешения прерываний/DMA (TIMx_DIER)

Смещение адреса: 0x0C
По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Содержимое битов:

0: Нельзя

1: Можно

— Бит 15 Резерв, не трогать.

— Бит 14 **TDE**: Запрос DMA запуска

— Бит 13 **COMDE**: Запрос DMA события COM

— Бит 12 **CC4DE**: Запрос DMA захвата/сравнения 4

— Бит 11 **CC3DE**: Запрос DMA захвата/сравнения 3

— Бит 10 **CC2DE**: Запрос DMA захвата/сравнения 2

— Бит 9 **CC1DE**: Запрос DMA захвата/сравнения 1

— Бит 8 **UDE**: Запрос DMA обновления

— Бит 7 **BIE**: Запрос прерывания останова

— Бит 6 **TIE**: Запрос прерывания запуска

— Бит 5 **COMIE**: Запрос прерывания события COM

— Бит 4 **CC4IE**: Запрос прерывания захвата/сравнения 4

— Бит 3 **CC3IE**: Запрос прерывания захвата/сравнения 3

— Бит 2 **CC2IE**: Запрос прерывания захвата/сравнения 2

— Бит 1 **CC1IE**: Запрос прерывания захвата/сравнения 1

— Бит 0 **UIE**: Запрос прерывания обновления

17.4.5. Регистр состояния (TIMx_SR)

Биты ставятся аппаратно, стираются записью 0.

Перезахват это появление события при стоящем его флаге.

Смещение адреса: 0x10

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CC4OF	CC3OF	CC2OF	CC1OF	Res.	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0	Res.	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Содержимое битов:

0: Не было

1: Было

— Биты 15:13 Резерв, не трогать.

— Бит 12 **CC4OF**: Флаг перезахвата захвата/сравнения 4

— Бит 11 **CC3OF**: Флаг перезахвата захвата/сравнения 3

— Бит 10 **CC2OF**: Флаг перезахвата захвата/сравнения 2

— Бит 9 **CC1OF**: Флаг перезахвата захвата/сравнения 1

— Бит 8 Резерв, не трогать.

— Бит 7 **BIF**: Флаг прерывания останова

Стирается при неактивном сигнале останова.

— Бит 6 **TIF**: Флаг прерывания запуска

Ставится аппаратно по активному фронту на входе TRGI при работающем контроллере режима ведомого во всех режимах, кроме Вентильного, (тогда по обоим фронтам).

— Бит 5 **COMIF**: Флаг прерывания события COM

Ставится аппаратно по событию COM (биты - CCxE, CCxNE, OCxM - обновились).

— Бит 4 **CC4IF**: Флаг прерывания захвата/сравнения 4

— Бит 3 **CC3IF**: Флаг прерывания захвата/сравнения 3

— Бит 2 **CC2IF**: Флаг прерывания захвата/сравнения 2

— Бит 1 **CC1IF**: Флаг прерывания захвата/сравнения 1

В режиме вывода CC1:

Есть исключения для выравнивания по центру (см. биты CMS в регистре TIMx_CR1).

В режиме ввода CC1:

— Бит 0 **UIF**: Флаг прерывания обновления

Удержание прерывания обновления ставится при:

– Переполнении или исчерпании счётчика и нулевом счётчике и UDIS=0 в регистре TIMx_CR1.
– При программной инициализации CNT битом UG в регистре TIMx_EGR, если URS=0 и UDIS=0 в регистре TIMx_CR1.
– При инициализации CNT событием запуска (см. регистр TIMx_SMCR), если URS=0 и UDIS=0 в регистре TIMx_CR1.

17.4.6. Регистр генерации событий (TIMx_EGR)

Событие генерируется программной установкой бита, снимается он аппаратно.

Смещение адреса: 0x14

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
								w	w	w	w	w	w	w	w

— Биты 15:8 Резерв, не трогать.
— Бит 7 **BG**: Останов.
1: Событие останова. Бит MOE сбрасывается, флаг BIF ставится. Могут возникнуть прерывание и запрос DMA.
— Бит 6 **TG**: Запуск.
1: Ставит флаг TIF в регистре TIMx_SR. Могут возникнуть прерывание и запрос DMA.
— Бит 5 **COMG**: Событие COM.
1: Стоящий бит CCPC разрешает обновление битов CCxE, CCxNE и OCxM
NB: Работает только у каналов с инверсными выходами.
— Бит 4 **CC4G**: Захват/сравнение 4.
— Бит 3 **CC3G**: Захват/сравнение 3.
— Бит 2 **CC2G**: Захват/сравнение 2.
— Бит 1 **CC1G**: Захват/сравнение 1.
1: Выдаёт событие захвата/сравнения:
В режиме вывода CC1:
Ставит флаг CC1IF. Могут возникнуть прерывание и запрос DMA.
В режиме ввода CC1:
Текущее значение счётчика пишется в TIMx_CCR1. Ставит флаг CC1IF. Могут возникнуть прерывание и запрос DMA. При уже стоящем флаге CC1IF ставится и флаг CC1OF.
— Бит 0 **UG**: Обновление.
1: Инициализирует счётчик и обновление регистров. Текущий счётчик предделителя чистится не влияя на само значение. В режиме по центру или при DIR=0 счётчик обнуляется, иначе перегружается из регистра TIMx_ARR (при DIR=1).

17.4.7. Регистр режима захвата/сравнения 1 (TIMx_CCMR1)

Смещение адреса: 0x18

По сбросу: 0x0000

Каналы могут работать на ввод (захват) и вывод (сравнение), что определяется битами `CCxS`. У остальных битов для ввода и вывода значение отличается. `OCxx` обозначает функции вывода, `ICxx` функции ввода. При этом используются разные каскады.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2 CE	OC2M[2:0]			OC2 PE	OC2 FE	CC2S[1:0]		OC1 CE	OC1M[2:0]			OC1 PE	OC1 FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				IC1F[3:0]				IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Сравнение выхода:
— Бит 15 **OC2CE:** Разрешение очистки сравнения 2.
— Биты 14:12 **OC2M[2:0]**: Режим сравнения выхода 2.
— Бит 11 **OC2PE**: Разрешение предзагрузки сравнения 2.
— Бит 10 **OC2FE**: Быстрое разрешение сравнения 2.
— Биты 9:8 **CC2S[1:0]**: Выбор Захвата/Сравнения 2.
Определяет направление канала (ввод/вывод) и используемый вход.
00: Канал CC2 выходной
01: Канал CC2 входной, IC2 на TI2

10: Канал CC2 входной, IC2 на TI1

11: Канал CC2 входной, IC2 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.

NB: Биты CC2S можно писать только при выключенном канале (CC2E = ‘0’ в TIMx_CCER).

— Бит 7 **OC1CE**: Разрешение очистки сравнения выхода 1.

0: ETRF на OC1REF не влияет

1: Высокий уровень на ETRF стирает OC1REF

— Биты 6:4 **OC1M[2:0]**: Режим сравнения выхода 1.

Определяют поведение выходного опорного сигнала OC1REF, от которого происходят OC1 и OC1N. OC1REF активен высоким, а OC1 и OC1N зависят от битов CC1P и CC1NP.

000: Заморозка - сравнение TIMx_CCR1 и TIMx_CNT на выходы не влияет (режим используется для генерации временной базы).

001: При совпадении TIMx_CNT и TIMx_CCR1 сигнал OC1REF ставится активным (высоким).

010: При совпадении TIMx_CNT и TIMx_CCR1 сигнал OC1REF ставится неактивным (низким).

011: Переключение - при TIMx_CNT=TIMx_CCR1 сигнал OC1REF перебрасывается.

100: Принудительный неактивный - OC1REF ставится низким.

101: Принудительный активный - OC1REF ставится высоким.

110: ШИМ 1 - при прямом счёте канал активен пока TIMx_CNT<TIMx_CCR1. При обратном счёте канал неактивен (OC1REF=‘0’) пока TIMx_CNT>TIMx_CCR1.

111: ШИМ 2 - при прямом счёте канал неактивен пока TIMx_CNT<TIMx_CCR1. При обратном счёте канал активен пока TIMx_CNT>TIMx_CCR1.

NB:

1: Эти биты нельзя изменить пока стоит LOCK уровень 3 (биты LOCK в регистре TIMx_BDTR) и CC1S=’00’ (канал включён на выход).

2: В ШИМ 1 и 2 уровень OCREF изменяется только когда изменяется результат сравнения или при переключении из “Заморозки” в “ШИМ”.

— Бит 3 **OC1PE**: Разрешение регистра предзагрузки.

0: Предзагрузка TIMx_CCR1 выключена. TIMx_CCR1 можно писать в любое время, новое значение начинает действовать незамедлительно.

1: Предзагрузка TIMx_CCR1 включена. Доступ идёт к регистру предзагрузки. TIMx_CCR1 пишется в активный регистр по событию обновления.

NB:

1: Эти биты нельзя изменить пока стоит LOCK уровень 3 (биты LOCK в регистре TIMx_BDTR) и CC1S=’00’ (канал включён на выход).

2: Режим ШИМ без установки регистра предзагрузки можно использовать только в режиме одного импульса (стоит бит OPM в регистре TIMx_CR1). Иначе поведение непредсказуемо.

— Бит 2 **OC1FE**: Разрешение быстрого сравнения.

Ускоряет действие события запуска по входу на выход CC.

0: CC1 работает как обычно (сравнение счётчика и CCR1) даже при включённом запуске. Минимальная задержка переключения выхода CC1 при фронте на входе составляет 5 тактов.

1: Активный фронт запуска по входу действует как сравнение на выходе CC1. Тогда OC ставится в уровень сравнения независимо от его результата. Задержка между импульсом на входе и выходом CC1 сокращается до 3 тактов. OCFE работает только в ШИМ 1 и ШИМ 2.

— Биты 1:0 **CC1S**: Выбор Захвата/Сравнения 1.

Определяет направление канала (ввод/вывод) и используемый вход.

00: Канал CC1 выходной

01: Канал CC1 входной, IC1 на TI1

10: Канал CC1 входной, IC1 на TI2

11: Канал CC1 входной, IC1 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.

NB: Биты CC1S можно писать только при выключенном канале (CC1E = ‘0’ в TIMx_CCER).

Захват входа.

— Биты 15:12 **IC2F**: Фильтр захвата входа 2.

— Биты 11:10 **IC2PSC[1:0]** : Предделитель захвата входа 2.

— Биты 9:8 **CC2S[1:0]**: Выбор Захвата/Сравнения 2.

Определяет направление канала (ввод/вывод) и используемый вход.

00: Канал CC2 выходной

01: Канал CC2 входной, IC2 на TI2

10: Канал CC2 входной, IC2 на TI1

11: Канал CC2 входной, IC2 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.

NB: Биты CC2S можно писать только при выключенном канале (CC2E = ‘0’ в TIMx_CCER).

— Биты 7:4 **IC1F[3:0]**: Фильтр захвата входа 1.

Определяет частоту выборки и длину цифрового фильтра сигнала TI1. Цифровой фильтр это счётчик, выдающий сигнал на выход после N последовательных событий на входе:

0000: Без фильтра, выборка на частоте f_{DTS}

0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2

0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4

0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8

0100: $f_{SAMPLING}=f_{DTS}/2$, N=6

0101: $f_{SAMPLING}=f_{DTS}/2$, N=8

0110: $f_{SAMPLING}=f_{DTS}/4$, N=6

0111: $f_{SAMPLING}=f_{DTS}/4$, N=8

1000: $f_{SAMPLING}=f_{DTS}/8$, N=6

1001: $f_{SAMPLING}=f_{DTS}/8$, N=8

1010: $f_{SAMPLING}=f_{DTS}/16$, N=5

1011: $f_{SAMPLING}=f_{DTS}/16$, N=6

1100: $f_{SAMPLING}=f_{DTS}/16$, N=8

1101: $f_{SAMPLING}=f_{DTS}/32$, N=5

1110: $f_{SAMPLING}=f_{DTS}/32$, N=6

1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

— Биты 3:2 **IC1PSC**: Предделитель захвата входа 1

Значение предделителя на входе CC1 (IC1).

По CC1E=’0’ (регистр TIMx_CCER) предделитель сбрасывается.

00: без предделителя, по каждому фронту на входе

01: каждые 2 события

10: каждые 4 события

11: каждые 8 события

— Биты 1:0 **CC1S**: Выбор Захвата/Сравнения 1.

Определяет направление канала (ввод/вывод) и используемый вход.

00: Канал CC1 выходной

01: Канал CC1 входной, IC1 на TI1

10: Канал CC1 входной, IC1 на TI2

11: Канал CC1 входной, IC1 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.

NB: Биты CC1S можно писать только при выключенном канале (CC1E = ‘0’ в TIMx_CCER).

17.4.8. Регистр режима захвата/сравнения 2 (TIMx_CCMR2)

Смещение адреса: 0x1C

По сбросу: 0x0000

Каналы могут работать на ввод (захват) и вывод (сравнение), что определяется битами `CCxS`. У остальных битов для ввода и вывода значение отличается. `OCxx` обозначает функции вывода, `ICxx` функции ввода. При этом используются разные каскады.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4 CE	OC4M[2:0]			OC4 PE	OC4 FE	CC4S[1:0]		OC3 CE.	OC3M[2:0]			OC3 PE	OC3 FE	CC3S[1:0]	
IC4F[3:0]				IC4PSC[1:0]				IC3F[3:0]				IC3PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Сравнение выхода:

— Бит 15 **OC4CE:** Разрешение очистки сравнения 4.

— Биты 14:12 **OC4M[2:0]**: Режим сравнения выхода 4.

— Бит 11 **OC4PE**: Разрешение предзагрузки сравнения 4.

— Бит 10 **OC4FE**: Быстрое разрешение сравнения 4.

— Биты 9:8 **CC4S[1:0]**: Выбор Захвата/Сравнения 4.

Определяет направление канала (ввод/вывод) и используемый вход.

00: Канал CC4 выходной

01: Канал CC4 входной, IC4 на TI4

10: Канал CC4 входной, IC4 на TI3

11: Канал CC4 входной, IC4 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.

NB: Биты CC4S можно писать только при выключенном канале (CC4E = ‘0’ в TIMx_CCER).

— Бит 7 **OC3CE**: Разрешение очистки сравнения выхода 3.

— Биты 6:4 **OC3M[2:0]**: Режим сравнения выхода 3.

— Бит 3 **OC3PE**: Разрешение регистра предзагрузки.

— Бит 2 **OC3FE**: Разрешение быстрого сравнения.

— Биты 1:0 **CC1S**: Выбор Захвата/Сравнения 1.

Определяет направление канала (ввод/вывод) и используемый вход.

00: Канал CC3 выходной

01: Канал CC3 входной, IC3 на TI3

10: Канал CC3 входной, IC3 на TI4

11: Канал CC3 входной, IC3 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.

NB: Биты CC3S можно писать только при выключенном канале (CC3E = ‘0’ в TIMx_CCER).

Захват входа.

— Биты 15:12 **IC4F**: Фильтр захвата входа 4.

— Биты 11:10 **IC4PSC[1:0]** : Предделитель захвата входа 4.

— Биты 9:8 **CC4S[1:0]**: Выбор Захвата/Сравнения 4.

Определяет направление канала (ввод/вывод) и используемый вход.

00: Канал CC4 выходной

01: Канал CC4 входной, IC4 на TI4

10: Канал CC4 входной, IC4 на TI3

11: Канал CC4 входной, IC4 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.

NB: Биты CC4S можно писать только при выключенном канале (CC4E = ‘0’ в TIMx_CCER).

— Биты 7:4 **IC3F[3:0]**: Фильтр захвата входа 3.

Определяет частоту выборки и длину цифрового фильтра сигнала TI3. Цифровой фильтр это счётчик, выдающий сигнал на выход после N последовательных событий на входе:

0000: Без фильтра, выборка на частоте f_{DTS}

0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2

0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4

0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8

0100: $f_{SAMPLING}=f_{DTS}/2$, N=6

0101: $f_{SAMPLING}=f_{DTS}/2$, N=8

0110: $f_{SAMPLING}=f_{DTS}/4$, N=6

0111: $f_{SAMPLING}=f_{DTS}/4$, N=8

1000: $f_{SAMPLING}=f_{DTS}/8$, N=6

1001: $f_{SAMPLING}=f_{DTS}/8$, N=8

1010: $f_{SAMPLING}=f_{DTS}/16$, N=5

1011: $f_{SAMPLING}=f_{DTS}/16$, N=6

1100: $f_{SAMPLING}=f_{DTS}/16$, N=8

1101: $f_{SAMPLING}=f_{DTS}/32$, N=5

1110: $f_{SAMPLING}=f_{DTS}/32$, N=6

1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

— Биты 3:2 **IC3PSC**: Предделитель захвата входа 3

— Биты 1:0 **CC3S**: Выбор Захвата/Сравнения 3.

Определяет направление канала (ввод/вывод) и используемый вход.

00: Канал CC3 выходной

01: Канал CC3 входной, IC3 на TI3

10: Канал CC3 входной, IC3 на TI4

11: Канал CC3 входной, IC3 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.

NB: Биты CC3S можно писать только при выключенном канале (CC3E = ‘0’ в TIMx_CCER).

17.4.9. Регистр разрешения захвата/сравнения (TIMx_CCER)

Смещение адреса: 0x20

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:14 Резерв, не трогать.

— Бит 13 **CC4P**: Полярность выхода захвата/сравнения 4

— Бит 12 **CC4E**: Разрешение выхода захвата/сравнения 4

— Бит 11 **CC3NP**: Полярность инверсного выхода захвата/сравнения 3

— Бит 10 **CC3NE**: Разрешение инверсного выхода захвата/сравнения 3

— Бит 9 **CC3P**: Полярность выхода захвата/сравнения 3

— Бит 8 **CC3E**:, Разрешение выхода захвата/сравнения 3

— Бит 7 **CC2NP**: Полярность инверсного выхода захвата/сравнения 2

— Бит 6 **CC2NE**: Разрешение инверсного выхода захвата/сравнения 2

— Бит 5 **CC2P**: Полярность выхода захвата/сравнения 2

— Бит 4 **CC2E**: Разрешение выхода захвата/сравнения 2

— Бит 3 **CC1NP**: Полярность инверсного выхода захвата/сравнения 1

0: OC1N активен высоким.

1: OC1N активен низким.

NB: Нельзя писать пока стоит LOCK уровень 2 или 3 (биты LOCK в регистре TIMx_BDTR) и CC1S=”00” (канал выводной).

— Бит 2 **CC1NE**: Разрешение инверсного выхода захвата/сравнения 1

0: Выкл. - OC1N не активен. Уровень OC1N функция битов MOE, OSSI, OSSR, OIS1, OIS1N и CC1E.

1: Вкл. - OC1N выводится на ножку, определённую битами MOE, OSSI, OSSR, OIS1, OIS1N и CC1E.

— Бит 1 **CC1P**: Полярность выхода захвата/сравнения 1

1: Выдаёт событие захвата/сравнения:

В режиме вывода CC1:

0: OC1 активен высоким.

1: OC1 активен низким.

В режиме ввода CC1:

Выбирает для запуска или захвата прямой или инверсный IC1.

0: прямой: захват по переднему фронту IC1. Внешний запуск по IC1 не инвертируется.

1: инверсный: захват по заднему фронту IC1. Внешний запуск по IC1 инвертируется.

NB: Нельзя писать пока стоит LOCK уровень 2 или 3 (биты LOCK в регистре TIMx_BDTR).

— Бит 0 **CC1E**: Разрешение выхода захвата/сравнения 1

В режиме вывода CC1:

0: Выкл. - OC1 не активен. Уровень OC1 функция битов MOE, OSSI, OSSR, OIS1, OIS1N и CC1NE.

1: Вкл. - OC1 выводится на ножку, определённую битами MOE, OSSI, OSSR, OIS1, OIS1N и CC1NE.

В режиме ввода CC1:

0: Захват выключен.

1: Захват включен.

Таблица 83. Управляющие биты OCx and OCxN

Управляющие биты					Состояние выхода(1)	
MOE	OSSI	OSSR	CCxE	CCxNE	Состояние OCx	Состояние OCxN
		0	0	0	Выкл. (таймером не управляется), OCx=0, OCx_EN=0	Выкл. (таймером не управляется), OCxN=0, OCxN_EN=0
		0	0	1	Выкл. (таймером не управляется), OCx=0, OCx_EN=0	OCxREF + Полярность OCxN=OCxREF xor CCxNP, OCxN_EN=1

1	X	0	1	0	OCxREF + Полярность OCx=OCxREF xor CCxP, OCx_EN=1	Выкл. (таймером не управляется) OCxN=0, OCxN_EN=0
		0	1	1	OCREF + Полярность + Задержка OCx_EN=1	Инверсия OCREF + Полярность + Задержка OCxN_EN=1
		1	0	0	Выкл. (таймером не управляется) OCx=CCxP, OCx_EN=0	Выкл. (таймером не управляется) OCxN=CCxNP, OCxN_EN=0
		1	0	1	Включён, неактивное состояние, OCx=CCxP, OCx_EN=1	OCxREF + Полярность OCxN=OCxREF xor CCxNP, OCxN_EN=1
		1	1	0	OCxREF + Полярность OCx=OCxREF xor CCxP, OCx_EN=1	Включён, неактивное состояние, OCxN=CCxNP, OCxN_EN=1
		1	1	1	OCREF + Полярность + Задержка OCx_EN=1	Инверсия OCREF + Полярность + Задержка OCxN_EN=1
0	0	X	0	0	Output Disabled (not driven by the timer) Asynchronously: OCx=CCxP, OCx_EN=0, OCxN=CCxNP, OCxN_EN=0 Then if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and OCxN both in active state.	
	0		0	1		
	0		1	0		
	0		1	1		
	1		0	0	Off-State (output enabled with inactive state) Asynchronously: OCx=CCxP, OCx_EN=1, OCxN=CCxNP, OCxN_EN=1 Then if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and OCxN both in active state	
	1		0	1		
	1		1	0		
	1		1	1		

1. When both outputs of a channel are not used (CCxE = CCxNE = 0), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

NB: Состояние внешних ножек I/O, подключённых к комплементарным OCx и OCxN, зависит от состояния OCx и OCxN и регистров GPIO и AFIO.

17.4.10.Счётчик TIM1 и TIM8 (TIMx_CNT)

Смещение адреса: 0x24

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:0 **CNT[15:0]**: Значение счётчика.

17.4.11.Предделитель TIM1 и TIM8 (TIMx_PSC)

Смещение адреса: 0x28

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:0 **PSC[15:0]**: Значение предделителя

Частота счёта (CK_CNT) равна f_{CK_PSC} / (PSC[15:0] + 1).

Значение PSC пишется в рабочий регистр предделителя по каждому событию обновления (включая очистку счётчика битом UG регистра TIMx_EGR или запуском в режиме Сброса).

17.4.12.Регистр предзагрузки TIM1 и TIM8 (TIMx_ARR)

Смещение адреса: 0x2C

По сбросу: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:0 **ARR[15:0]**: Значение перезагрузки.

Если значение нулевое, то счётчик блокируется.

17.4.13.Счётчик повторения TIM1 и TIM8 (TIMx_RCR)

Смещение адреса: 0x30

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								REP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:8 Резерв, не трогать.

— Биты 7:0 **REP[7:0]**: Значение счётчика.

Регистр позволяет изменять частоту обновления регистров сравнения и частоту выдачи прерываний.

При каждом обнулении обратного счётчика REP_CNT выдаётся событие обновления. Он перегружается значением REP только по событию U_RC.

То есть в режиме ШИМ (REP+1) соответствует:

- числу периодов ШИМ в режиме по фронту
- числу полупериодов ШИМ в режиме по центру.

17.4.14.Регистр 1 захвата/сравнения TIM1 и TIM8 (TIMx_CCR1)

Смещение адреса: 0x34

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

— Биты 15:0 **CCR1[15:0]**: Значение захвата/сравнения

В режиме вывода CC1:

Значение предзагрузки CCR1 надо писать в рабочий регистр захвата/сравнения. Если функция предзагрузки не включена (бит OC1PE в регистре TIMx_CCMR1), то он пишется постоянно, иначе только по событию обновления.

Результат сравнения рабочего регистра и TIMx_CNT подаётся на выход OC1.

В режиме ввода CC1:

CCR1 содержит значение счётчика по последнему событию захвата 1 (IC1). Здесь TIMx_CCR1 только читается.

17.4.15.Регистр 2 захвата/сравнения TIM1 и TIM8 (TIMx_CCR2)

Смещение адреса: 0x38

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

— Биты 15:0 **CCR2[15:0]**: Значение захвата/сравнения

В режиме вывода CC2:

Значение предзагрузки CCR2 надо писать в рабочий регистр захвата/сравнения. Если функция предзагрузки не включена (бит OC2PE в регистре TIMx_CCMR2), то он пишется постоянно, иначе только по событию обновления.

Результат сравнения рабочего регистра и TIMx_CNT подаётся на выход OC2.

В режиме ввода CC2:

CCR2 содержит значение счётчика по последнему событию захвата 2 (IC2). Здесь TIMx_CCR2 только читается.

17.4.16.Регистр 3 захвата/сравнения TIM1 и TIM8 (TIMx_CCR3)

Смещение адреса: 0x3C

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

— Биты 15:0 **CCR3[15:0]**: Значение захвата/сравнения

В режиме вывода CC3:

Значение предзагрузки CCR3 надо писать в рабочий регистр захвата/сравнения. Если функция предзагрузки не включена (бит OC3PE в регистре TIMx_CCMR3), то он пишется постоянно, иначе только по событию обновления.

Результат сравнения рабочего регистра и TIMx_CNT подаётся на выход OC3.

В режиме ввода CC3:

CCR3 содержит значение счётчика по последнему событию захвата 3 (IC3). Здесь TIMx_CCR3 только читается.

17.4.17.Регистр 4 захвата/сравнения TIM1 и TIM8 (TIMx_CCR4)

Смещение адреса: 0x40

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

— Биты 15:0 **CCR4[15:0]**: Значение захвата/сравнения

В режиме вывода CC4:

Значение предзагрузки CCR4 надо писать в рабочий регистр захвата/сравнения. Если функция предзагрузки не включена (бит OC4PE в регистре TIMx_CCMR4), то он пишется постоянно, иначе только по событию обновления.

Результат сравнения рабочего регистра и TIMx_CNT подаётся на выход OC4.

В режиме ввода CC4:

CCR4 содержит значение счётчика по последнему событию захвата 4 (IC4). Здесь TIMx_CCR4 только читается.

17.4.18.Регистр останова и задержки (TIMx_BDTR)

Смещение адреса: 0x44

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

NB:

Поскольку биты AOE, BKP, BKE, OSSI, OSSR и DTG[7:0] могут блокироваться конфигурацией LOCK, то их надо все расставить первой записью в регистр TIMx_BDTR.

— Бит 15 **MOE**: Главное разрешение вывода

Снимается асинхронно аппаратурой пр активном входе останова. Ставится программно или автоматически в зависимости от бита AOE. Работает только с выводными каналами.

0: Выходы OC и OCN выключены или переведены в состояние Простоя.

1: Выходы OC и OCN включаются их битами разрешения (CCxE, CCxNE в регистре TIMx_CCER).

— Бит 14 **AOE**: Автоматическое разрешение вывода

0: MOE ставится только программно

1: MOE ставится программно или автоматически по следующему событию обновления (если неактивен вход останова)

NB: Пока стоит LOCK уровень 1 (биты LOCK в регистре TIMx_BDTR) этот бит изменить невозможно.

— Бит 13 **BKP**: Полярность останова

0: Вход BRK активен низким

1: Вход BRK активен высоким

NB: Пока стоит LOCK уровень 1 (биты LOCK в регистре TIMx_BDTR) этот бит изменить невозможно.

NB: Вступает в действие через 1 такт APB.

— Бит 12 **BKE**: Разрешение останова

0: Входы останова (BRK и сбой CSS) отключены

1: Входы останова (BRK и сбой CSS) включены

NB: Пока стоит LOCK уровень 1 (биты LOCK в регистре TIMx_BDTR) этот бит изменить невозможно.

NB: Вступает в действие через 1 такт APB.

— Бит 11 **OSSR**: Выбор состояния отключения в режиме Run

Используется выводными каналами с комплементарными выходами при MOE=1.

0: Если неактивен, выходы OC/OCN выключены (разрешение OC/OCN =0).

1: Если неактивен, выходы OC/OCN включены с неактивным уровнем когда CCxE=1 или CCxNE=1. Тогда, разрешение OC/OCN =1

NB: Пока стоит LOCK уровень 2 (биты LOCK в регистре TIMx_BDTR) этот бит изменить невозможно.

— Бит 10 **OSSI**: Выбор состояния отключения в режиме Idle (Простой)

Используется выводными каналами при MOE=0.

0: Если неактивен, выходы OC/OCN выключены (разрешение OC/OCN =0).

1: Если неактивен, выходы OC/OCN включены первыми с уровнем Простоя когда CCxE=1 или CCxNE=1. Тогда, разрешение OC/OCN =1

NB: Пока стоит LOCK уровень 2 (биты LOCK в регистре TIMx_BDTR) этот бит изменить невозможно.

— Биты 9:8 **LOCK[1:0]**: Конфигурация защиты записи (LOCK)

00: LOCK OFF - Защиты записи нет.

01: LOCK уровень 1 = Защищены биты DTG в TIMx_BDTR, OISx и OISxN в TIMx_CR2 и BKE/BKP/AOE в регистре TIMx_BDTR.

10: LOCK уровень 2 = LOCK уровень 1 + биты полярности CC (CCxP/CCxNP в TIMx_CCER канала, поставленного на вывод битами CCxS) равно как и биты OSSR OSSI.

11: LOCK уровень 3 = LOCK уровень 2 + биты управления CC (OCxM и OCxPE в TIMx_CCMRx канала, поставленного на вывод битами CCxS).

NB: Биты LOCK можно писать только один раз после сброса. Затем регистр TIMx_BDTR замораживается до следующего сброса.

— Биты 7:0 **DTG[7:0]**: Установка генератора задержки.

Определяет задержку (DT) между комплементарными выходами.

DTG[7:5]=0xx => DT=DTG[7:0] x t_{dtg} с t_{dtg} =t_{DTS}.

DTG[7:5]=10x => DT=(64+DTG[5:0]) x t_{dtg} с T_{dtg}=2 x t_{DTS}.

DTG[7:5]=110 => DT=(32+DTG[4:0]) x t_{dtg} с T_{dtg}=8 x t_{DTS}.

DTG[7:5]=111 => DT=(32+DTG[4:0]) x t_{dtg} с T_{dtg}=16 x t_{DTS}.

Если T_{DTS}=125ns (8MHz), то задержка равна:

0 до 15875 ns шагами по 125 ns,

16 us до 31750 ns шагами по 250 ns,

32 us до 63 us шагами по 1 us,

64 us до 126 us шагами по 2 us

NB: Пока стоит LOCK уровень 1, 2 или 3 (биты LOCK в регистре TIMx_BDTR) эти биты изменить невозможно.

17.4.19.Регистр управления DMA TIM1 и TIM8 (TIMx_DCR)

Смещение адреса: 0x48

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			DBL[4:0]					Reserved			DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

— Биты 15:13 Резерв, не трогать.

— Биты 12:8 **DBL[4:0]**: Длина пакета DMA

Это число передач DMA (таймер обнаруживает передачу пакета после выполнения чтения/записи по адресу регистра TIMx_DMAR).

Адрес TIMx_DMAR:

00000: 1 передача

00001: 2 передачи

00010: 3 передачи

…

10001: 18 передач

— Биты 7:5 Резерв, не трогать.

— Биты 4:0 **DBA[4:0]**: Базовый адрес DMA

Это базовый адрес передач DMA (при доступе через адрес TIMx_DMAR). DBA определяется как смещение от адреса регистра TIMx_CR1.

Пример:

00000: TIMx_CR1,

00001: TIMx_CR2,

00010: TIMx_SMCR,

...

Пример: Есть передача: DBL = 7 передач и DBA = TIMx_CR1. Тогда доступ будет к 7 регистрам, начиная с адреса TIMx_CR1.

17.4.20.Регистр адреса полного доступа DMA (TIMx_DMAR)

Смещение адреса: 0x4C

По сбросу: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAB[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:0 **DMAB[31:0]**: Регистр пакетного доступа DMA

Чтение и запись в регистр DMAR обращаются к регистру по адресу
(адрес TIMx_CR1) + (DBA + индекс DMA) x 4

где (адрес TIMx_CR1) это адрес управляющего регистра 1, DBA это базовый адрес DMA в регистре TIMx_DCR, индекс DMA автоматически изменяется при DMA передачах от 0 до DBL из регистра TIMx_DCR.

Пример пакетного DMA

Здесь пакетом DMA полусловами пишутся регистры CCRx (x = 2, 3, 4).

Нужны следующие шаги:

1. Ставим нужный канал DMA:
   - Адрес периферии DMA пишем в регистр DMAR
   - Адрес памяти это адрес данных в RAM для записи в регистры CCRx.
   - Число передач = 3.
   - Выключаем кольцевой режим.
2. Пишем поля DBA и DBL регистра DCR: DBL = 3 передачи, DBA = 0xE.
3. Разрешаем запрос DMA обновления TIMx (ставим бит UDE в регистре DIER).
4. Включаем TIMx
5. Включаем канал DMA

Здесь каждый регистр CCRx обновляется единожды. Если регистры CCRx нужно обновить дважды, то число передач должно быть 6. Буфер в RAM содержит data1, data2, data3, data4, data5 и data6. В регистры CCRx они передаются так: по первому запросу DMA, data1 в CCR2, data2 в CCR3, data3 в CCR4 и по второму запросу обновления DMA, data4 в CCR2, data5 в CCR3 в data6 iв CCR4.

17.4.21.Карта регистров TIM1 и TIM8

0x00	**TIMx_CR1**	Reserved																CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																	0	0	0	0	0	0	0	0	0	0
0x04	**TIMx_CR2**	Reserved										OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Reserved	CCPC
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x08	**TIMx_SMCR**	Reserved								ETP	ECE	ETP S [1:0]		ETF[3:0]				MSM	TS[2:0]			Reserved	SMS[2:0]			
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	
0x0C	**TIMx_DIER**	Reserved								TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE		
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x10	**TIMx_SR**	Reserved											CC4OF	CC3OF	CC2OF	CC1OF	Reserved	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF	
	Reset value												0	0	0	0		0	0	0	0	0	0	0	0	
0x14	**TIMx_EGR**	Reserved																BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG	
	Reset value																	0	0	0	0	0	0	0	0	
0x18	**TIMx_CCMR1** Output Compare mode	Reserved								OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]		
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	TIMx_CCMR1 Input Capture mode	Reserved								IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]		
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x1C	**TIMx_CCMR2** Output Compare mode	Reserved								O24CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]		
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	TIMx_CCMR2 Input Capture mode	Reserved								IC4F[3:0]				IC4 PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3S [1:0]		
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x20	**TIMx_CCER**	Reserved											CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	**TIMx_CNT**	Reserved								CNT[15:0]																
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x28	**TIMx_PSC**	Reserved								PSC[15:0]																
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

0x2C	**TIMx_ARR**	Reserved								ARR[15:0]																
	Reset value									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	**TIMx_RCR**	Reserved																REP[7:0]								
	Reset value																	0	0	0	0	0	0	0	0	
0x34	**TIMx_CCR1**	Reserved								CCR1[15:0]																
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x38	**TIMx_CCR2**	Reserved								CCR2[15:0]																
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x3C	**TIMx_CCR3**	Reserved								CCR3[15:0]																
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x40	**TIMx_CCR4**	Reserved								CCR4[15:0]																
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x44	**TIMx_BDTR**	Reserved								MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]								
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x48	**TIMx_DCR**	Reserved												DBL[4:0]					Reserved		DBA[4:0]					
	Reset value													0	0	0	0	0			0	0	0	0	0	
0x4C	**TIMx_DMAR**	DMAB[31:0]																								
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

18. Таймеры общего назначения (TIM2 - TIM5)

18.1. Введение в TIM2 - TIM5

Это 16-бит само-перегружаемые таймеры с программируемым предделителем.

Они могут использоваться для измерения длительности входных импульсов (захват входа), генерации сигналов (сравнение выхода и ШИМ).

Длина импульсов и сигналов может модулироваться от нескольких микросекунд до миллисекунд с помощью предделителей таймера и тактов RCC.

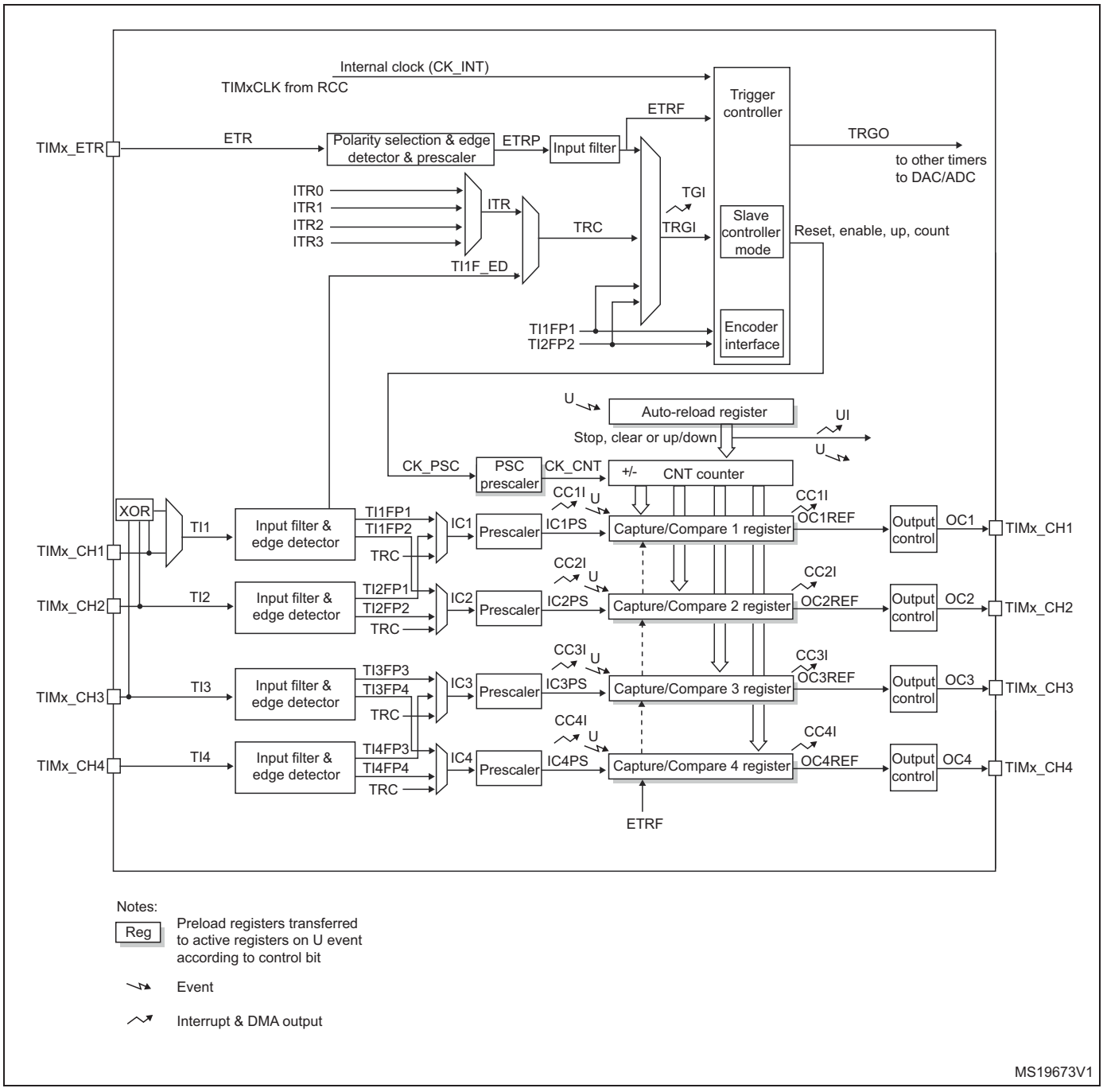
Улучшенные (TIM1 and TIM8) и обычные (TIMx) таймеры полностью независимы и не имеют общих ресурсов. Могут синхронизироваться между собой. См. *Секцию 18.3.15*.

18.2. Основные свойства TIM2 - TIM5

Это:

- 16-бит прямой, обратный, реверсивный счёт с само-перезагрузкой.
- 16-бит программируемый предделитель (можно “налету”) тактов на число от 1 до 65536.
- До 4 независимых каналов для:
  - Захвата входа
  - Сравнение выхода
  - ШИМ генерация (Фронт и Центр)
  - Одиночный импульс
- Инверсные выходы с программируемой задержкой
- Схема синхронизации внешнего управления и объединения таймеров.
- Генерация Прерываний/DMA по следующим событиям:
  - Обновление: переполнение/исчерпание, инициализация счётчика (программно или запуском)
  - Запуск (старт, стоп, инициализация или счёт по сигналу)
  - Захват входа
  - Сравнение выхода
- Поддерживает инкрементный (квадратурный) кодер и датчики Холла для позиционирования.
- Вход запуска от внешних тактов или шагового управления током.

Рис. 134. Блок-схема таймера общего назначения.



18.3. Функциональное описание таймеров TIM2 - TIM5

18.3.1. Узел счёта

Это 16-бит счётчик с регистром само-перезагрузки. Имеет прямой, обратный и реверсивный режимы. Тактирование может поступать через предделитель.

Счётчик, регистр перезагрузки и предделитель доступны программно для чтения и записи даже во время счёта.

Включает:

- Регистр счётчика (TIMx_CNT)
- Регистр предделителя (TIMx_PSC)
- Регистр перезагрузки (TIMx_ARR)

Регистр перезагрузки предзагружаемый (есть видимый и скрытый регистры). Содержимое видимого регистра пишется в скрытый постоянно или по каждому событию обновления (UEV), в зависимости от бита разрешения (ARPE) в регистре TIMx_CR1. Оно посылается программно или при переполнении/исчерпании счётчика при снятом бите UDIS в регистре TIMx_CR1.

Счётчик тактируется выходом предделителя CK_CNT. Разрешается битом CEN в регистре TIMx_CR1.

Счёт начинается через 1 такт после установки бита CEN в регистре TIMx_CR1.

Описание предделителя

Это 16-бит счётчик делителя входной частоты (от 1 до 65536) с видимым регистром TIMx_PSC. Его можно менять налету. Новое значение счётчика пишется по следующему событию обновления.

Рис. 135. Временная диаграмма с изменением предделителя с 1 на 2.

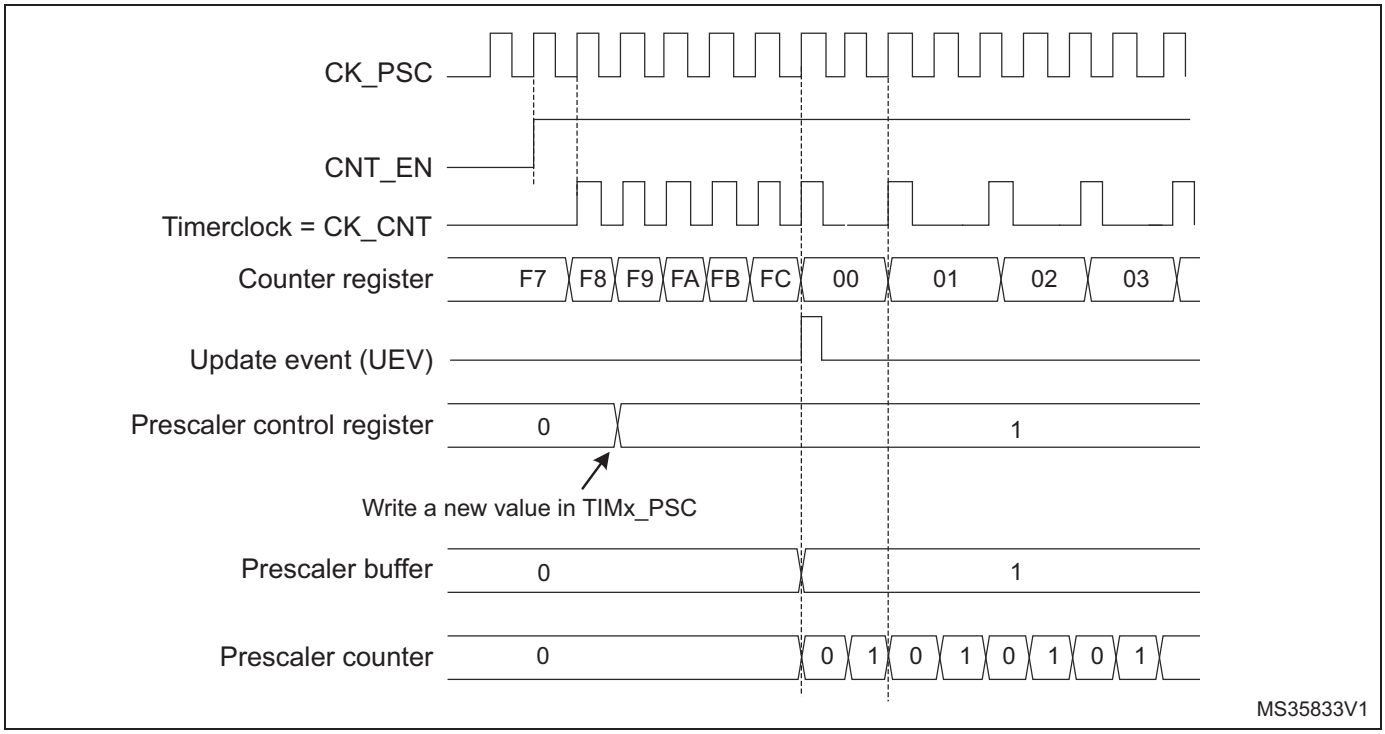
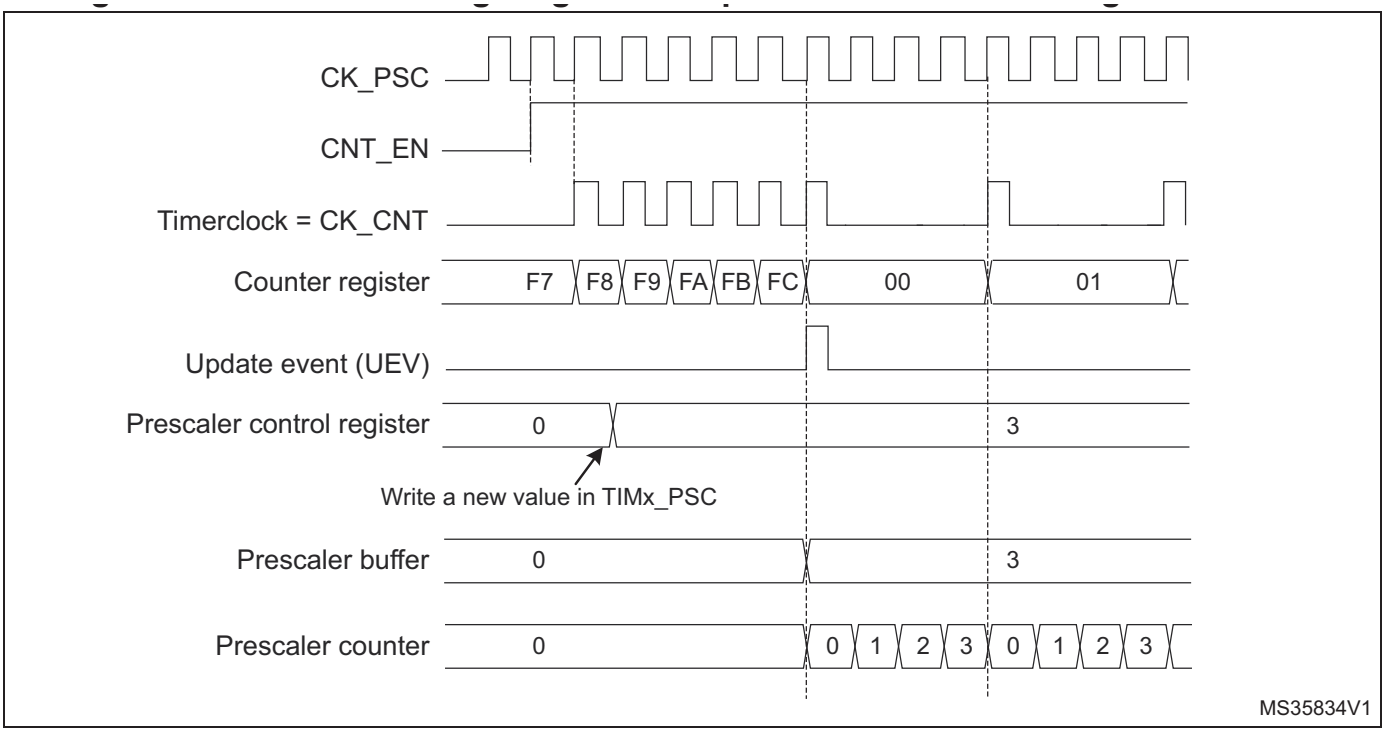


Рис. 136. Временная диаграмма с изменением предделителя с 1 на 4.



18.3.2. Режимы счёта

Прямой счёт

Счёт идёт от 0 до значения перезагрузки (регистр TIMx_ARR), сбрасывается в 0 и выдаёт событие переполнения счётчика.

Событие обновления UEV выдаётся при каждом переполнении счётчика или установкой бита UG в регистре TIMx_EGR (программно или контроллером режима ведомого).

Событие UEV выключается установкой бита UDIS в регистре TIMx_CR1. Этим предупреждается запись скрытого регистра во время изменения регистра предзагрузки. Также событие UEV не появляется при сброшенном бите UDIS 0. Однако, счётчик продолжает считать с 0, равно как и предделитель (не изменив своего значения). Кроме того, если стоит бит URS в регистре TIMx_CR1, то установка бита UG выдаёт событие UEV без установки флага UIF (без прерывания и запроса DMA). Этим предупреждается одновременная выдача прерываний обновления и захвата при очистке счётчика по событию захвата.

По событию обновления обновляются все регистры и ставится флаг обновления (бит UIF в регистре TIMx_SR). Это зависит от бита URS:

- В скрытый регистр перезагрузки пишется содержимое TIMx_ARR,
- Буфер предделителя перегружается из регистра TIMx_PSC.

Примеры ниже используют TIMx_ARR=0x36.

Рис. 137. Временная диаграмма с делителем 1.

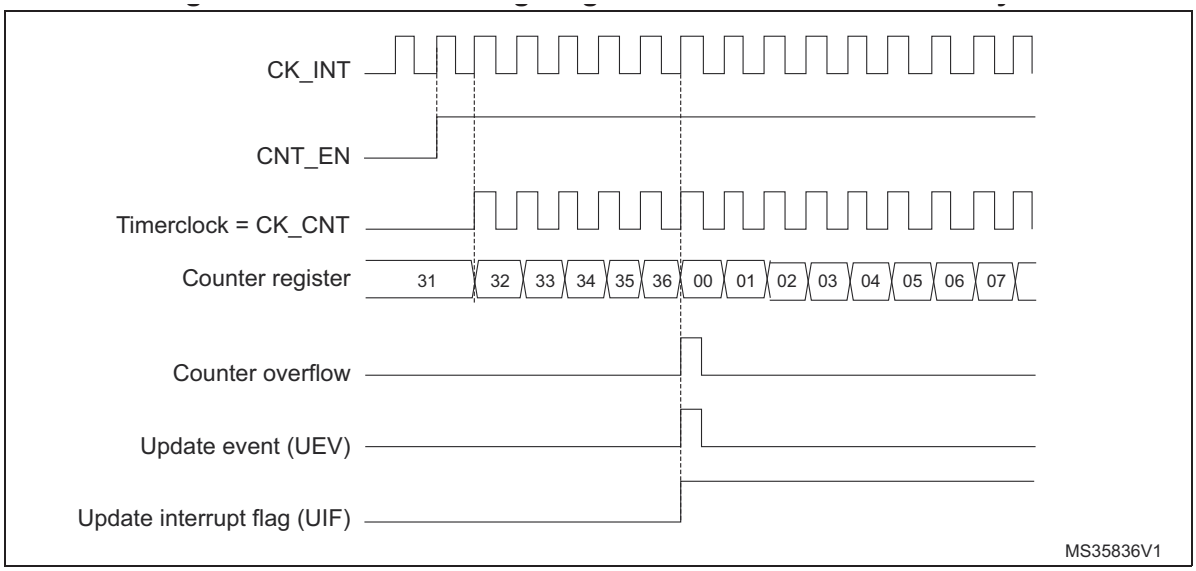


Рис. 138. Временная диаграмма с делителем 2.

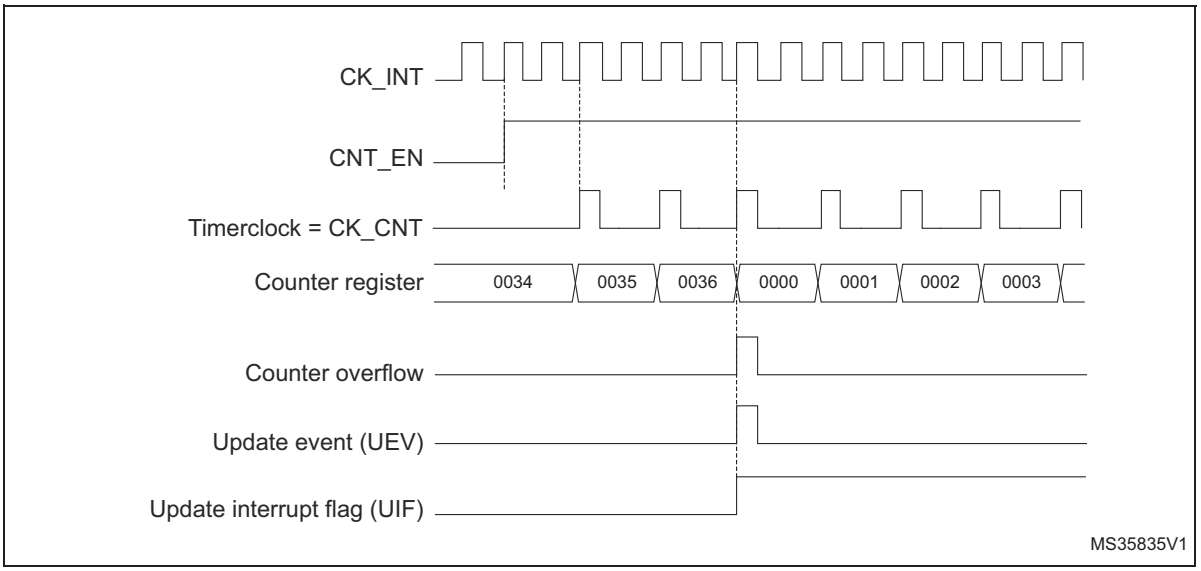


Рис. 139. Временная диаграмма с делителем 4.

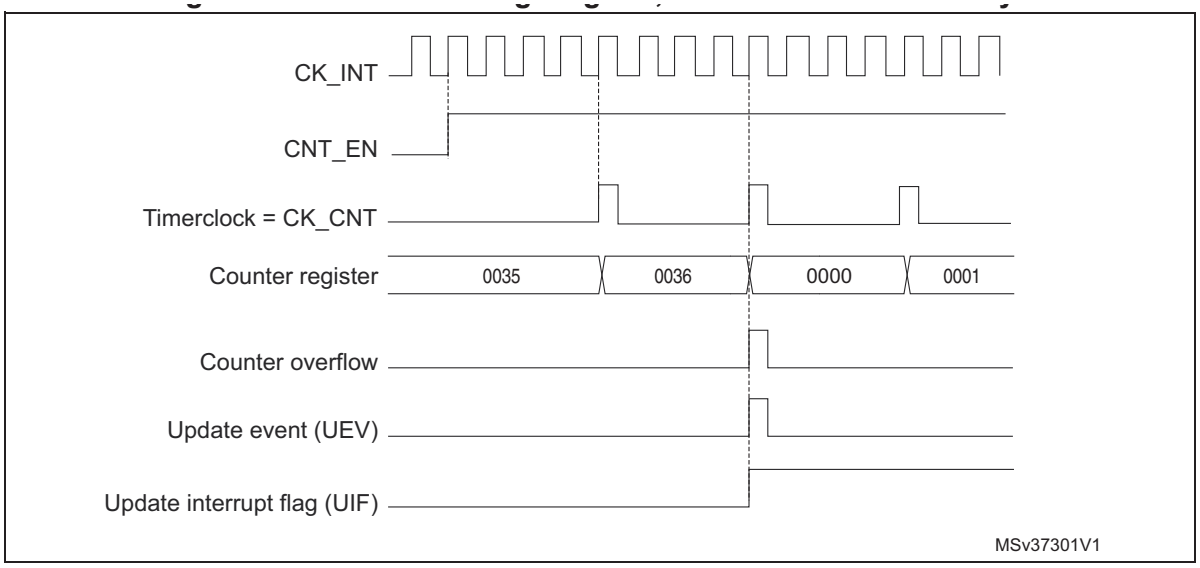


Рис. 140. Временная диаграмма с делителем N.

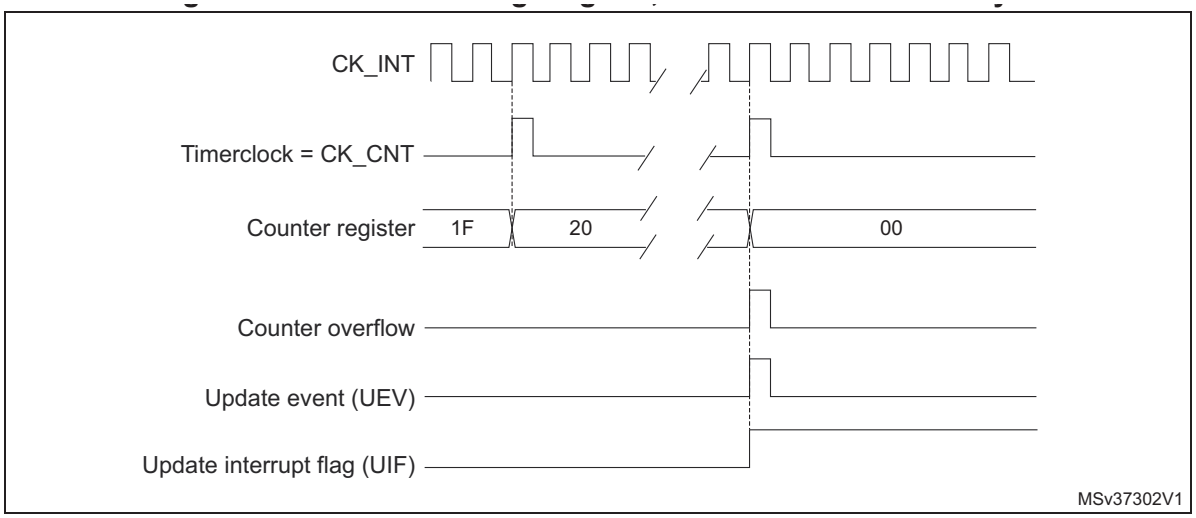


Рис. 141. Временная диаграмма события при ARPE=0 (TIMx_ARR не предзагружен).

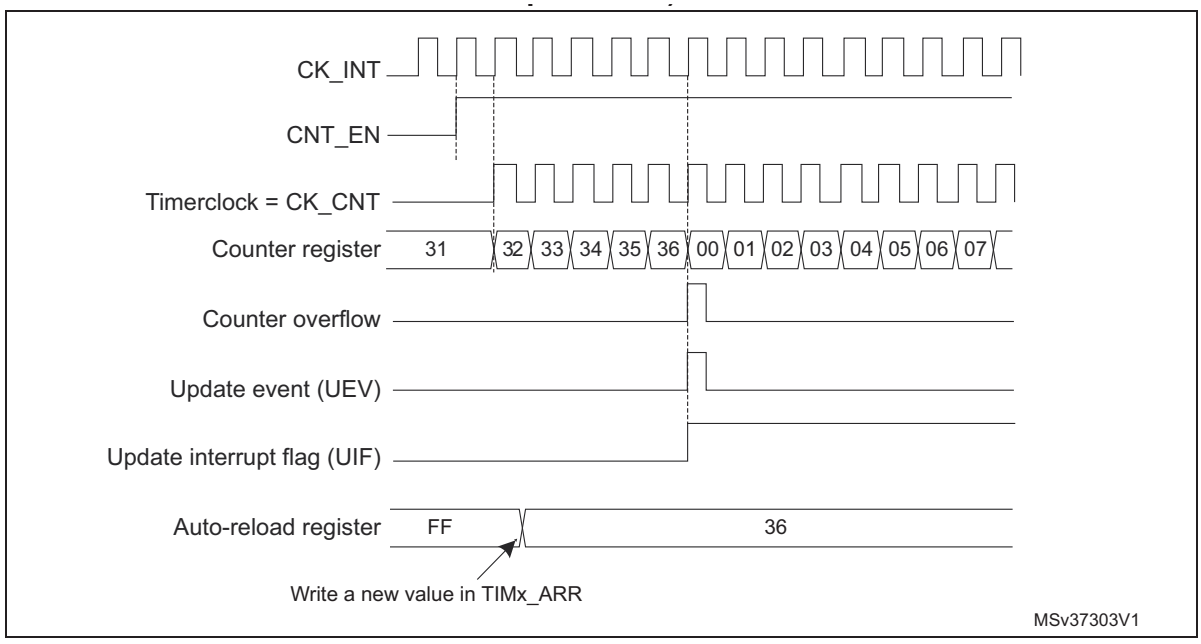
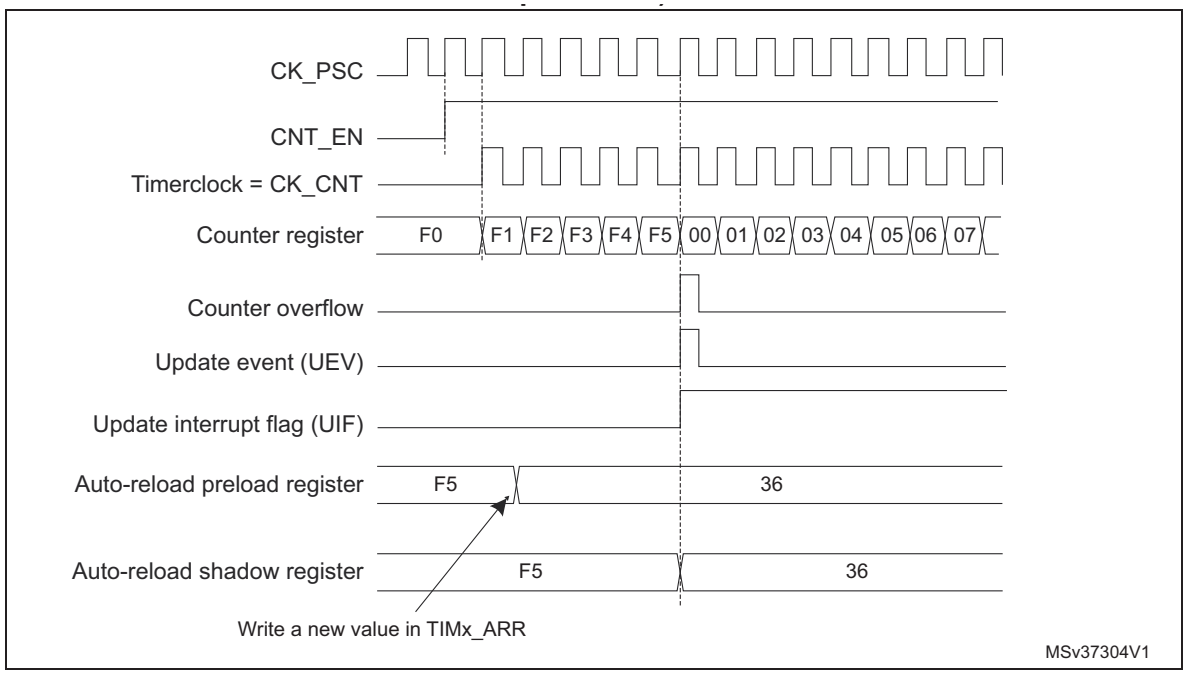


Рис. 142. Временная диаграмма события при ARPE=1 (TIMx_ARR предзагружен).



Обратный счёт

Счёт идёт от значения перезагрузки (регистр TIMx_ARR) до 0, перегружает счётчик и выдаёт событие исчерпания счётчика.

Событие обновления UEV выдаётся при каждом исчерпании счётчика или установкой бита UG в регистре TIMx_EGR (программно или контроллером режима ведомого).

Событие UEV выключается установкой бита UDIS в регистре TIMx_CR1. Этим предупреждается запись скрытого регистра во время изменения регистра предзагрузки. Также событие UEV не появляется при сброшенном бите UDIS 0. Однако, счётчик продолжает считать с значения перезагрузки, также как и предделитель с 0 (не изменив своего значения). Кроме того, если стоит бит URS в регистре TIMx_CR1, то установка бита UG выдаёт событие UEV без установки флага UIF (без прерывания и запроса DMA). Этим предупреждается одновременная выдача прерываний обновления и захвата при очистке счётчика по событию захвата.

По событию обновления обновляются все регистры и ставится флаг обновления (бит UIF в регистре TIMx_SR). Это зависит от бита URS:

- Буфер предделителя перегружается из регистра TIMx_PSC.
- В скрытый регистр перезагрузки пишется содержимое TIMx_ARR. Скрытый регистр перезагрузки обновляется раньше счётчика и работа продолжается с нового значения.

Примеры ниже используют TIMx_ARR=0x36.

Рис. 143. Временная диаграмма с делителем 1.

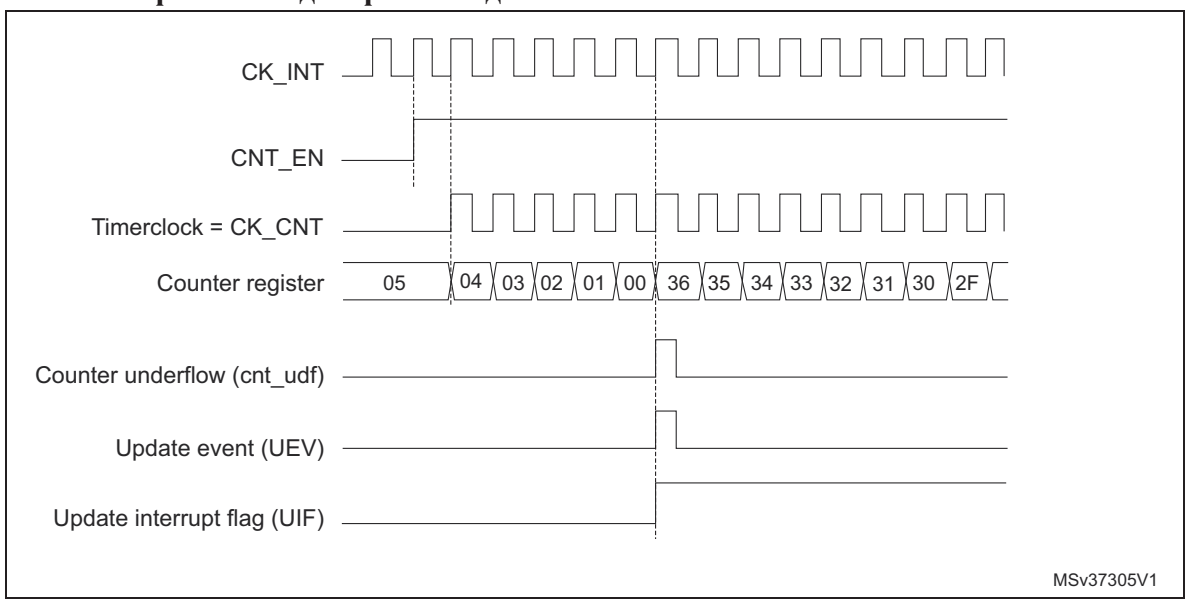


Рис. 144. Временная диаграмма с делителем 2.

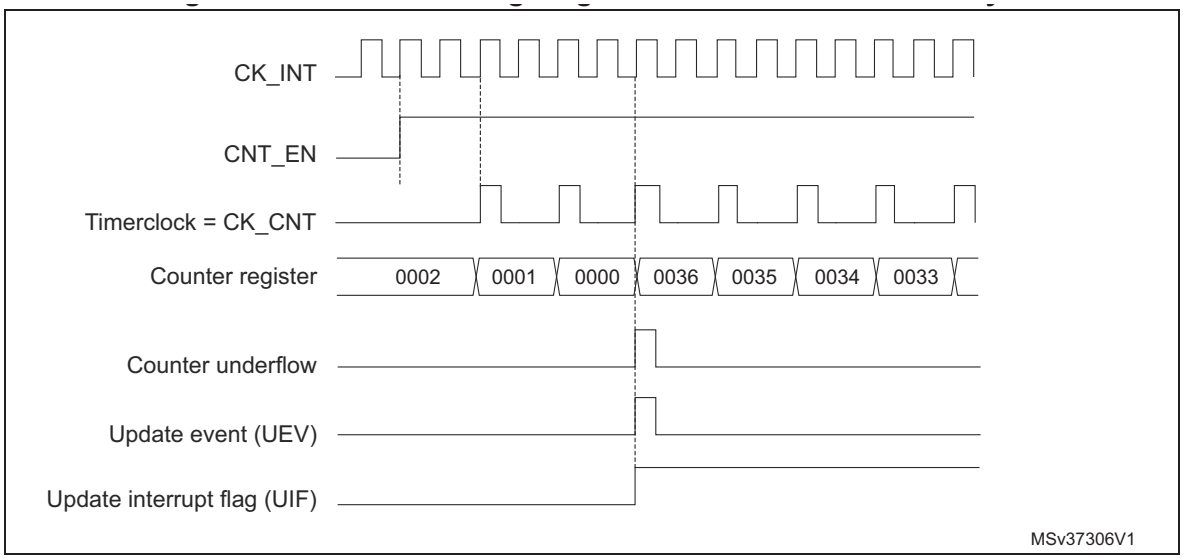


Рис. 145. Временная диаграмма с делителем 4.

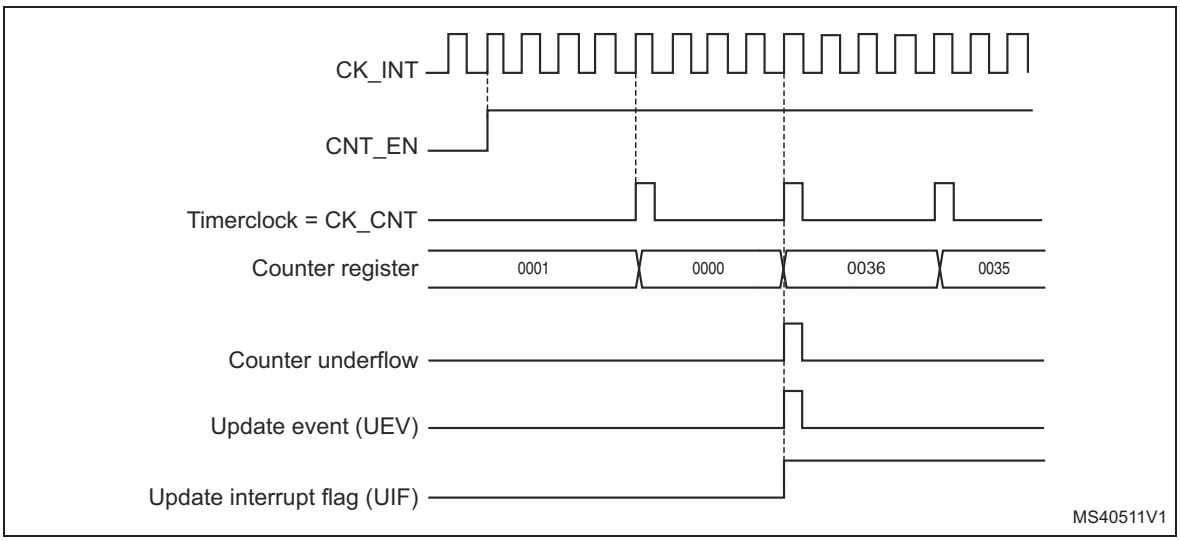


Рис. 146. Временная диаграмма с делителем N.

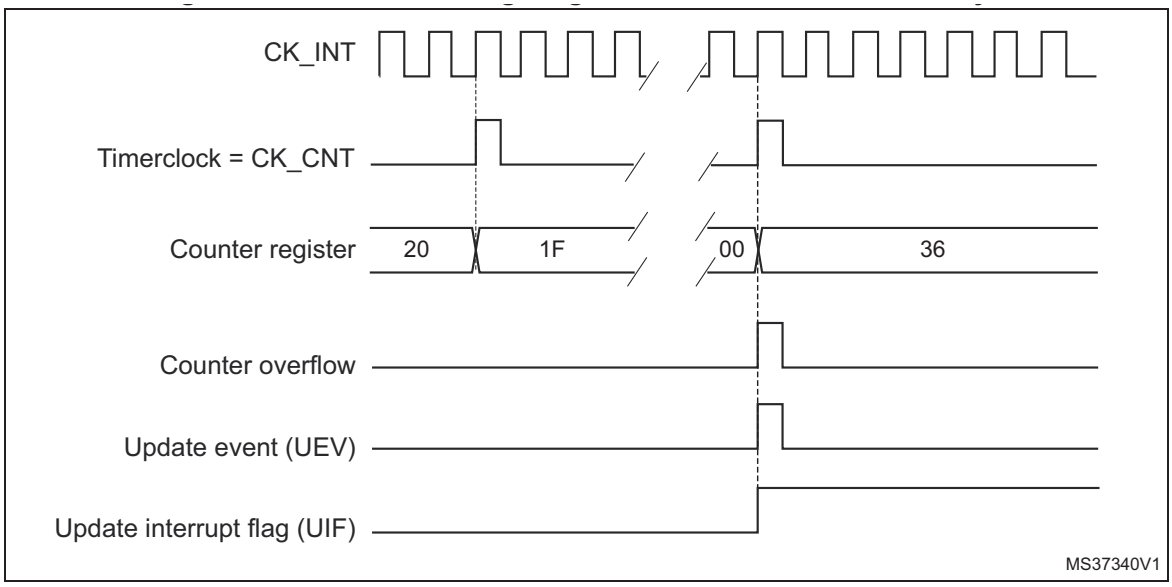
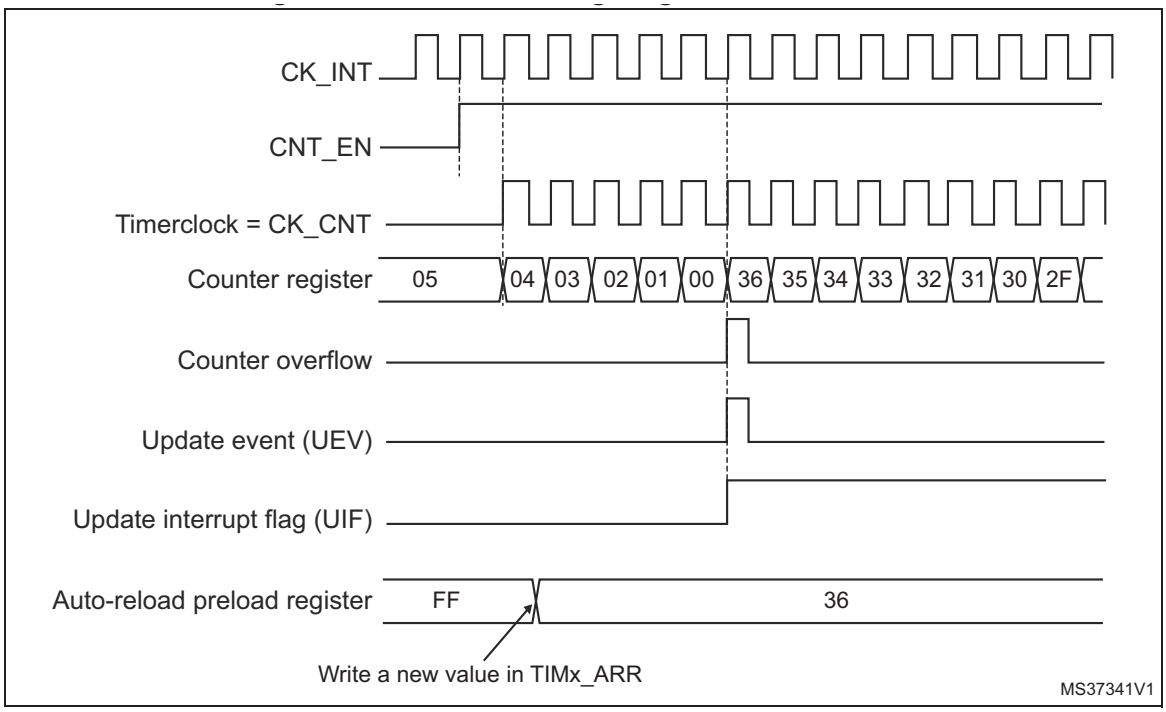


Рис. 147. Временная диаграмма события обновления.



Реверсивный счёт

Счёт идёт от 0 до значения перезагрузки минус 1 (регистр TIMx_ARR-1), выдаёт событие переполнения счётчика, затем считает обратно до 1, выдаёт событие исчерпания счётчика и начиняет счёт с 0.

Реверсивный режим включается если биты CMS в регистре TIMx_CR1 не равны '00'. Флаг прерывания Сравнения выхода выводных каналов ставится когда: завершается обратный счёт (Реверсивный режим 1, CMS = "01"), завершается прямой счёт (Реверсивный режим 2, CMS = "10") завершается прямой и обратный счёт (Реверсивный режим 3, CMS = "11").

В этом режиме бит DIR в регистре TIMx_CR1 писать нельзя. Он изменяется аппаратно и указывает текущее направление счёта.

Событие обновления может выдаваться на каждое переполнение и на каждое исчерпание счётчика установкой бита UG в регистре TIMx_EGR (программно или контроллером режима ведомого). В этом случае счётчик и предделитель начинают счёт с 0.

Событие UEV выключается установкой бита UDIS в регистре TIMx_CR1. Этим предупреждается запись скрытого регистра во время изменения регистра предзагрузки. Также событие UEV не появляется при сброшенном бите UDIS 0. Однако, счётчик продолжает считать с вверх и вниз, основываясь на текущем значении предзагрузки.

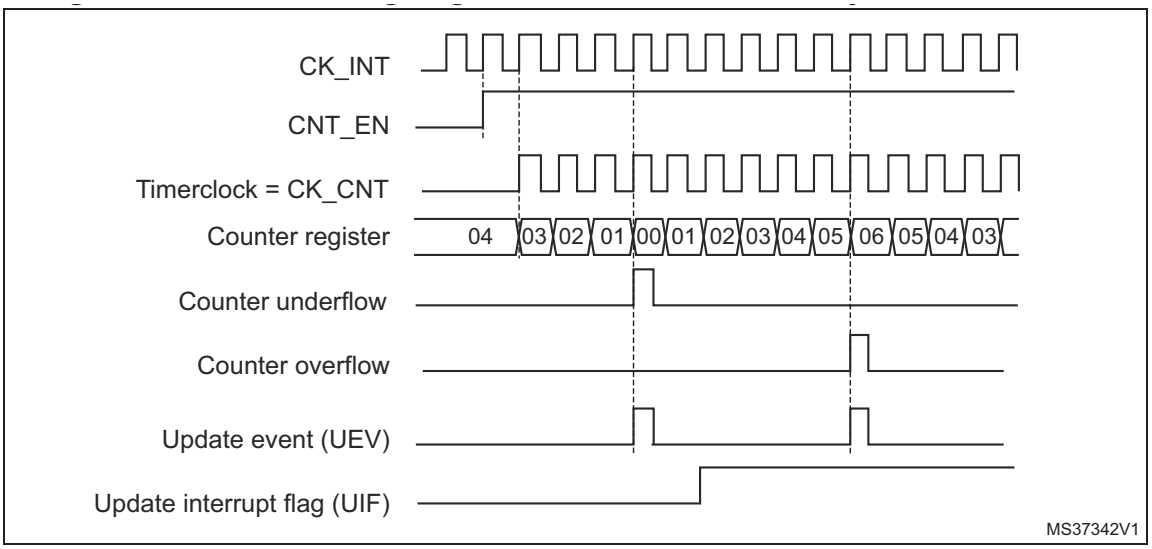
Кроме того, если стоит бит URS в регистре TIMx_CR1, то установка бита UG выдаёт событие UEV без установки флага UIF (без прерывания и запроса DMA). Этим предупреждается одновременная выдача прерываний обновления и захвата при очистке счётчика по событию захвата.

По событию обновления обновляются все регистры и ставится флаг обновления (бит UIF в регистре TIMx_SR). Это зависит от бита URS:

- Буфер предделителя перегружается из регистра TIMx_PSC.
- В скрытый регистр перезагрузки пишется содержимое TIMx_ARR. В случае обновления по переполнению счётчика скрытый регистр перезагрузки обновляется раньше счётчика и работа продолжается с нового значения.

Примеры ниже используют разные частоты тактов.

Рис. 148. Временная диаграмма с делителем 1, TIMx_ARR = 0x6.



1. Реверсивный режим 1.

Рис. 149. Временная диаграмма с делителем 2.

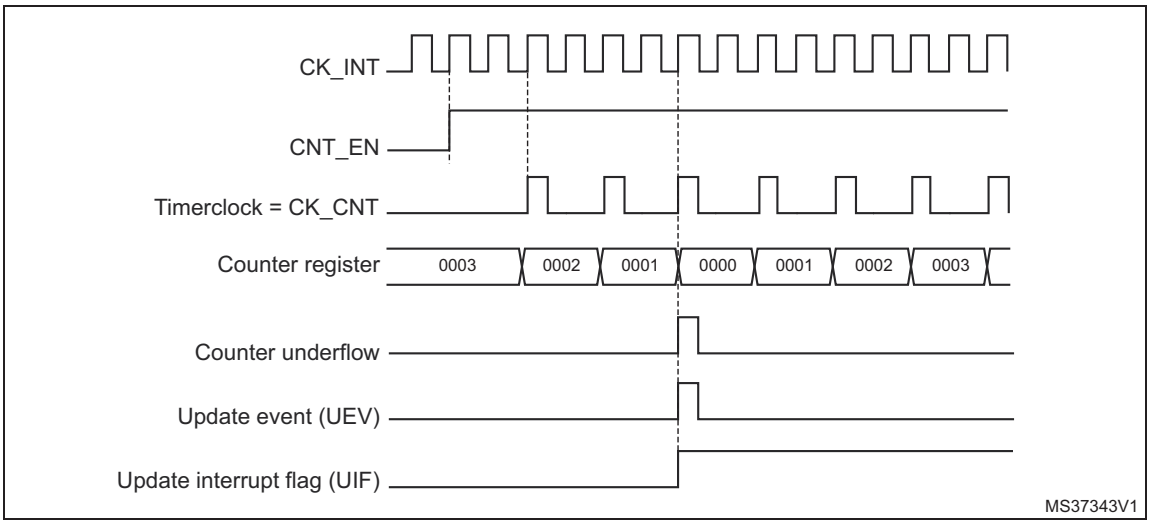
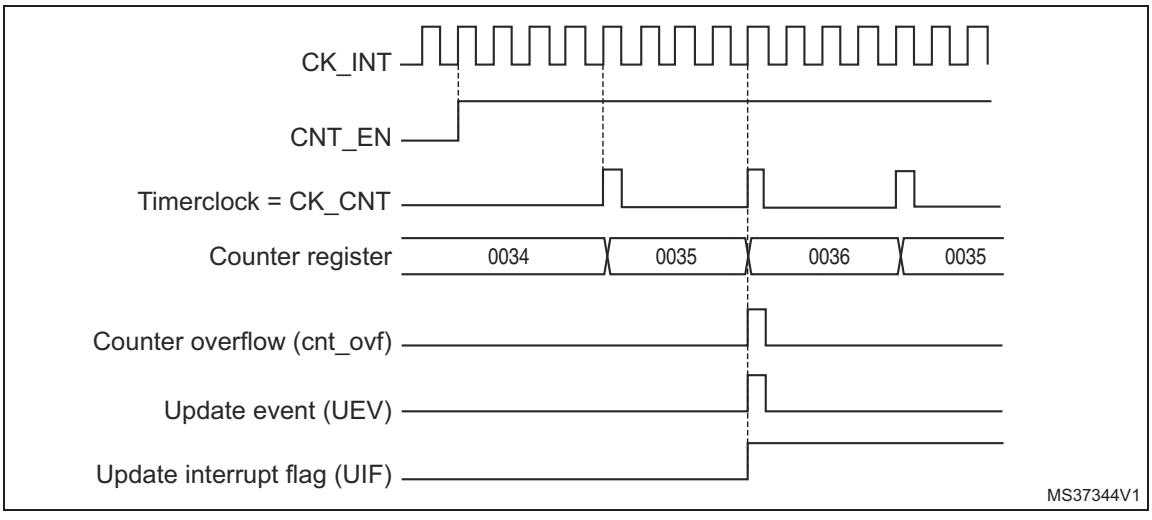


Рис. 150. Временная диаграмма с делителем 4, TIMx_ARR=0x36.



1. Реверсивный режим 2 или 3 с флагом UIF по переполнению.

Рис. 151. Временная диаграмма с делителем N.

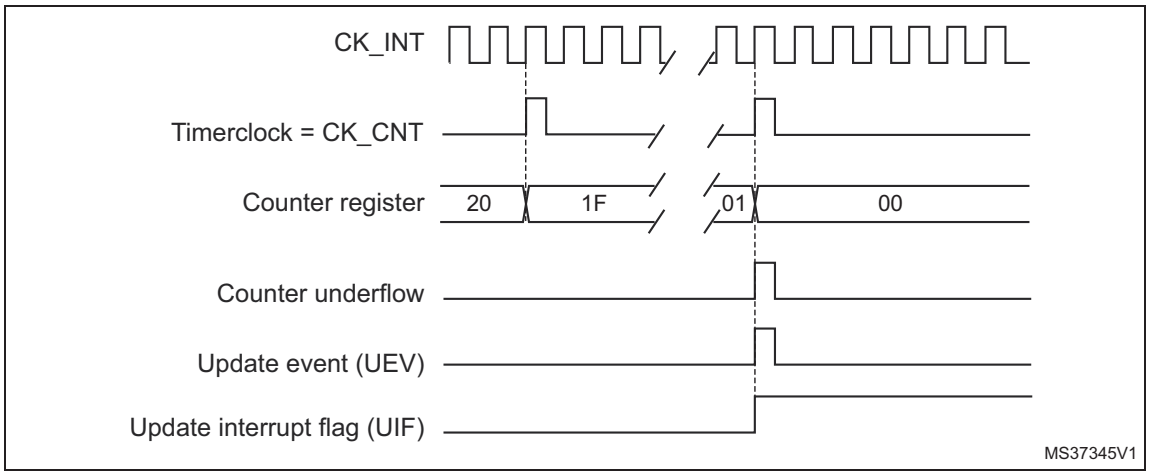


Рис. 152. Временная диаграмма, обновление с ARPE=1 (исчерпание).

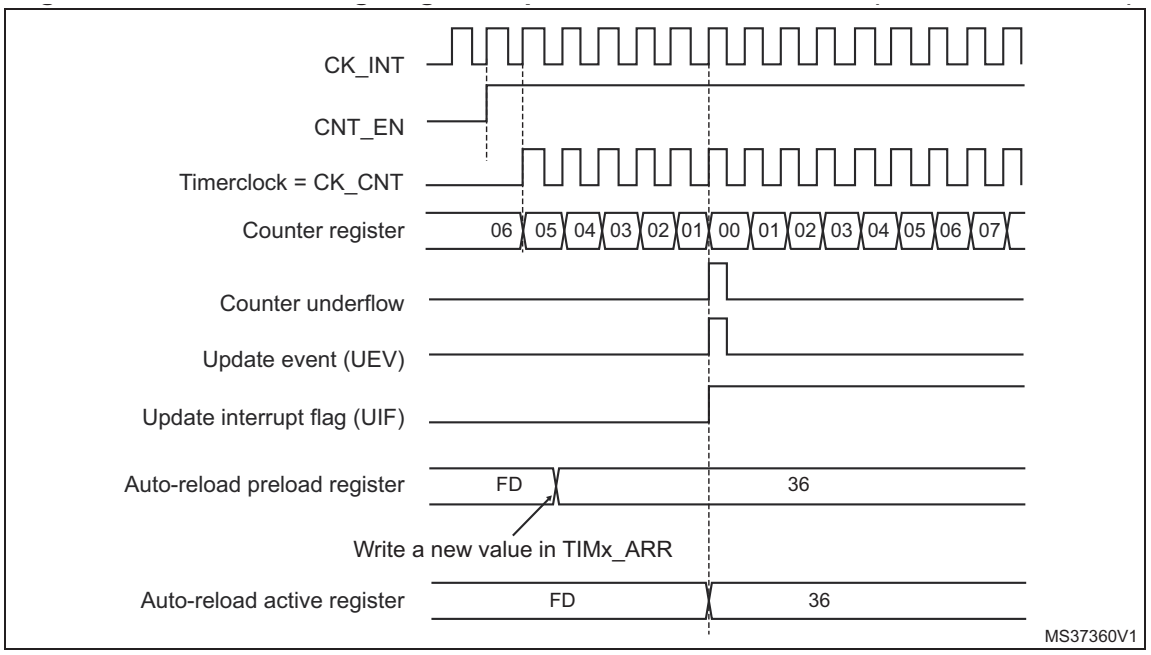
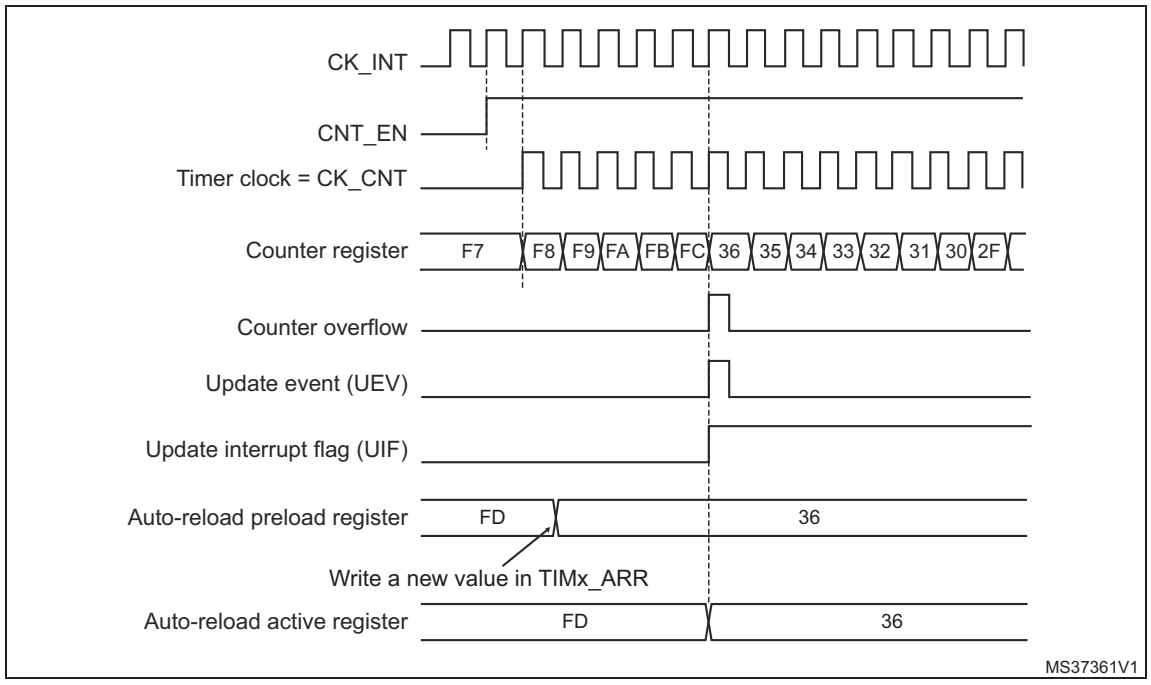


Рис. 153. Временная диаграмма, обновление с ARPE=1 (переполнение).



18.3.3. Выбор тактов

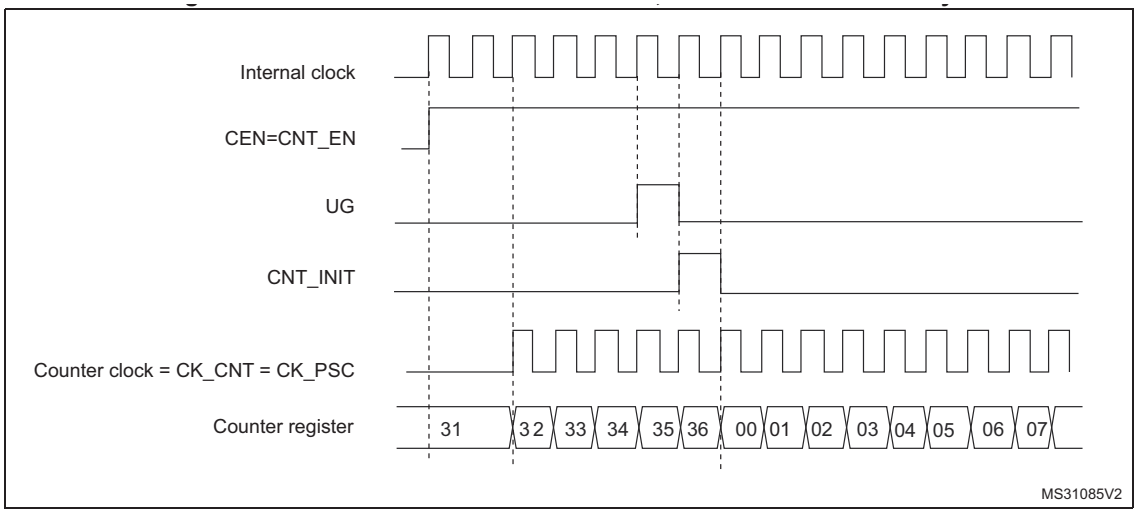
Счётчик может тактироваться из следующих источников:

- Внутренний (CK_INT)
- Внешний режим 1: внешняя ножка TIx
- Внешний режим 2: вход внешнего сигнала ETR
- Входы внутреннего сигнала (ITRx): использование одного таймера предделителем для другого.

Внутренний источник (CK_INT)

Если в ведомом режиме контроллер выключен (SMS=000), то всем управляют только биты CEN, DIR (в регистре TIMx_CR1) и UG (в регистре TIMx_EGR). Они изменяются только программно (кроме UG, снимающегося аппаратно). Сразу после установки бита CEN предделитель начинает получать внутренние такты CK_INT.

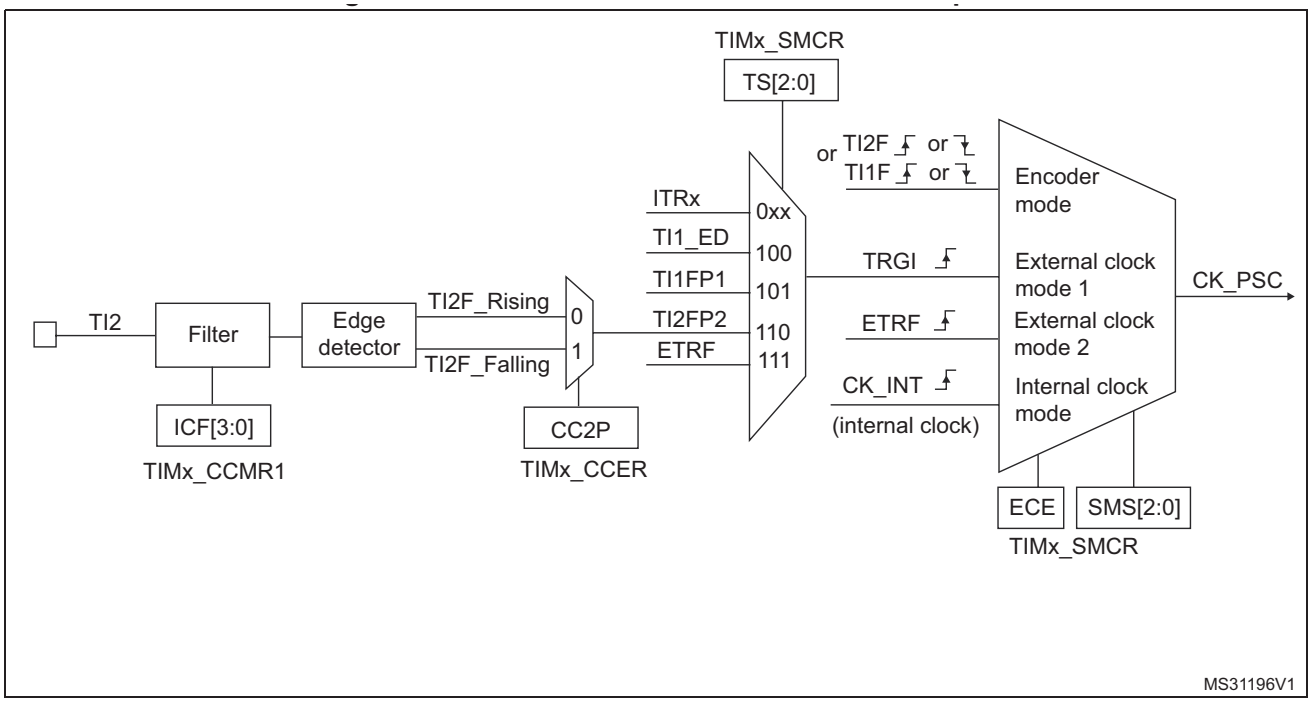
Рис. 154. Схема управления в нормальном режиме прямого счёта без предделителя.



Внешний источник режим 1.

Включается при установке SMS=111 в регистре TIMx_SMCR. Счётчик работает от переднего или заднего фронта выбранного входа.

Рис. 155. Внешний источник TI2.



Например, процедура включения режима прямого счёта по переднему фронту входа TI2:

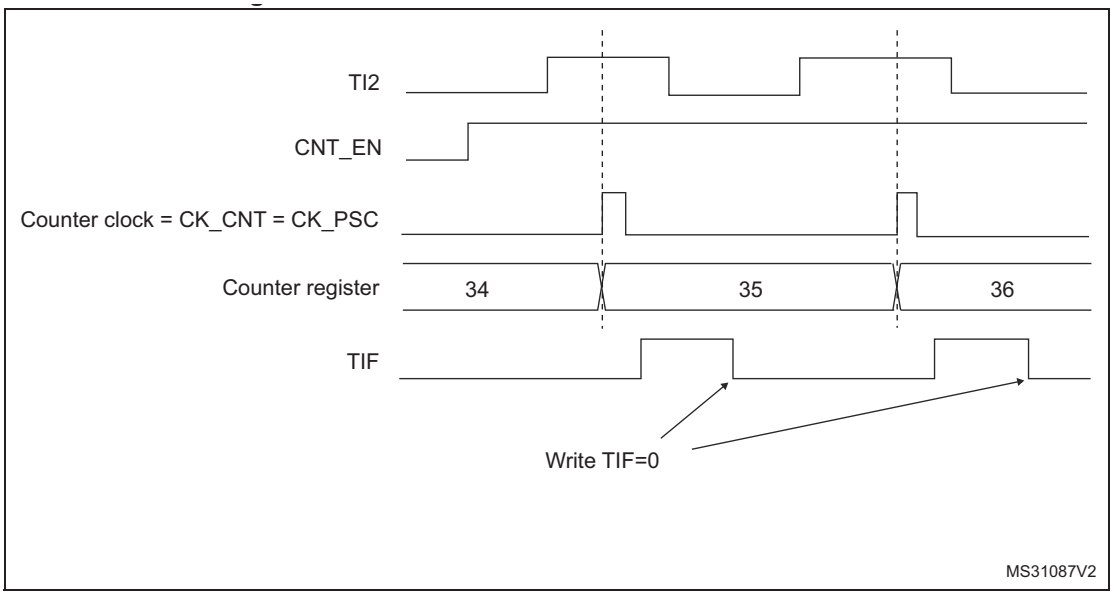
1. Включить определение переднего фронта TI2 канала 2 записью CC2S = ‘01’ в регистре TIMx_CCMR1.
2. Установить длительность входного фильтра битами IC2F[3:0] в регистре TIMx_CCMR1 (если фильтр не нужен, то остаётся IC2F=0000).
3. Выбрать полярность переднего фронта записью CC2P=0 в регистре TIMx_CCER.
4. Включить внешний режим 1 тактирования записью SMS=111 в регистре TIMx_SMCR.
5. Выбрать TI2 как источник сигнала записью TS=110 в регистре TIMx_SMCR.
6. Включить счётчик записью CEN=1 в регистре TIMx_CR1.

Предделитель захвата для запуска не используется, ну и бог с ним.

По переднему фронту на TI2 счётчик один раз тикает и ставит флаг TIF.

Задержка между передним фронтом на TI2 и реальным тиком появляется из-за схемы ресинхронизации на входе TI2.

Рис. 156. Схема управления внешнего режима 1.

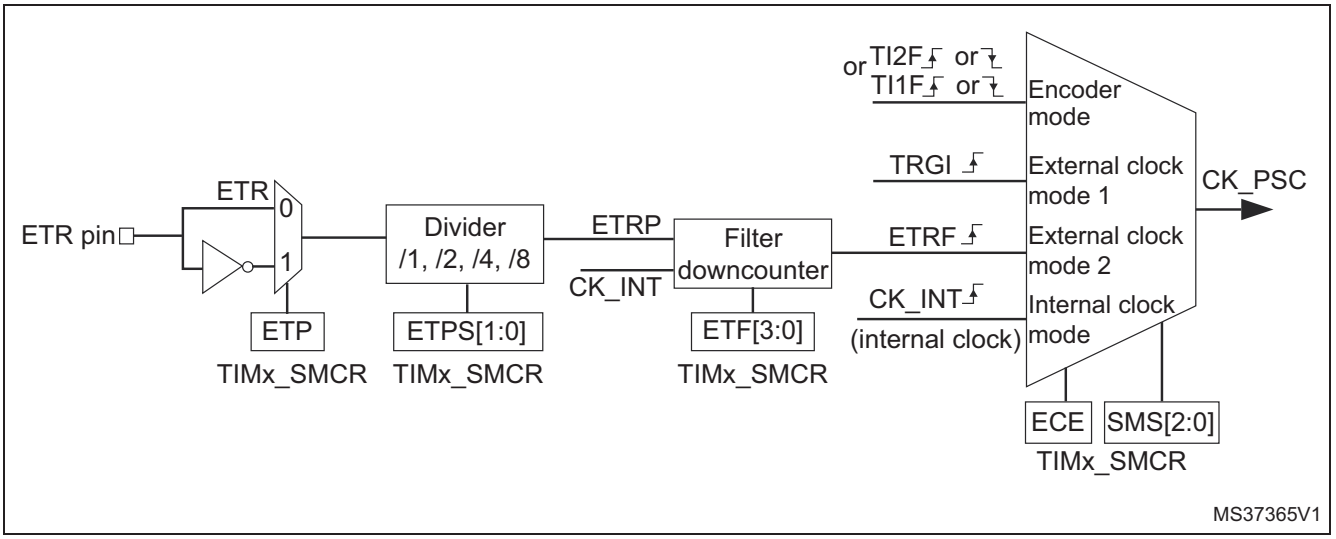


Внешний источник режим 2.

Включается записью ECE=1 в регистре TIMx_SMCR.

Счётчик работает по переднему или заднему фронту входа ETR.

Рис. 157. Блок входа внешнего запуска.



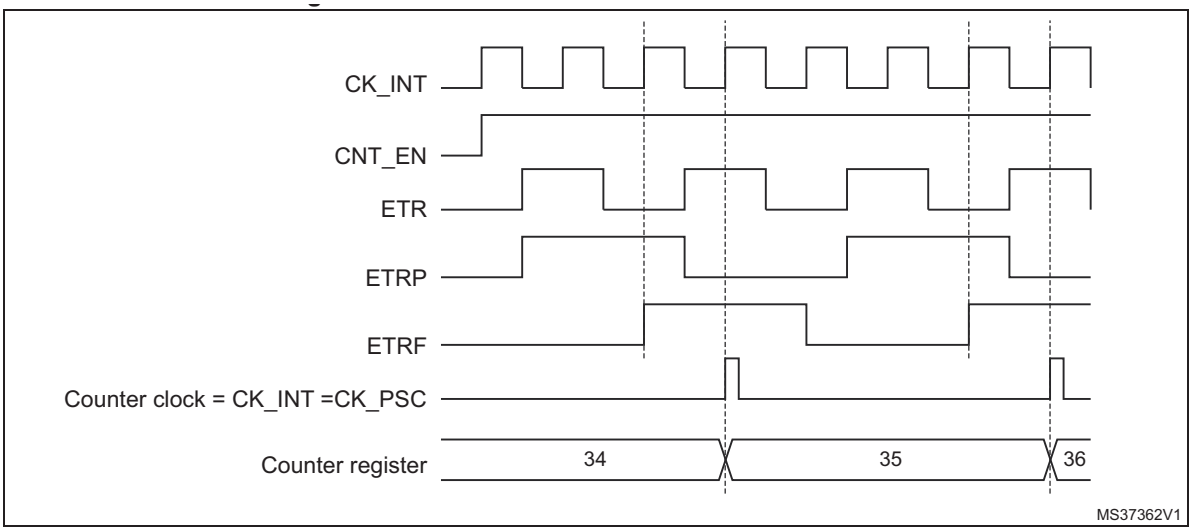
Например, включение режима прямого счёта по каждому 2 переднему фронту на входе ETR:

1. Здесь фильтр не нужен, так что ETF[3:0]=0000 в регистр TIMx_SMCR.
2. В предделитель пишем ETPS[1:0]=01 в регистре TIMx_SMCR
3. Выбираем передний фронт на ножке ETR записью ETP=0 в регистре TIMx_SMCR.
4. Разрешаем внешний режим 2 записью ECE=1 в регистре TIMx_SMCR.
5. Включаем счётчик записью CEN=1 в регистре TIMx_CR1.

Счётчик тикает единожды на каждые 2 передних фронта ETR.

Задержка между передним фронтом на ETR и реальным тиком появляется из-за схемы ресинхронизации на входе сигнала ETRP.

Рис. 158. Схема управления с внешним источником режим 2.

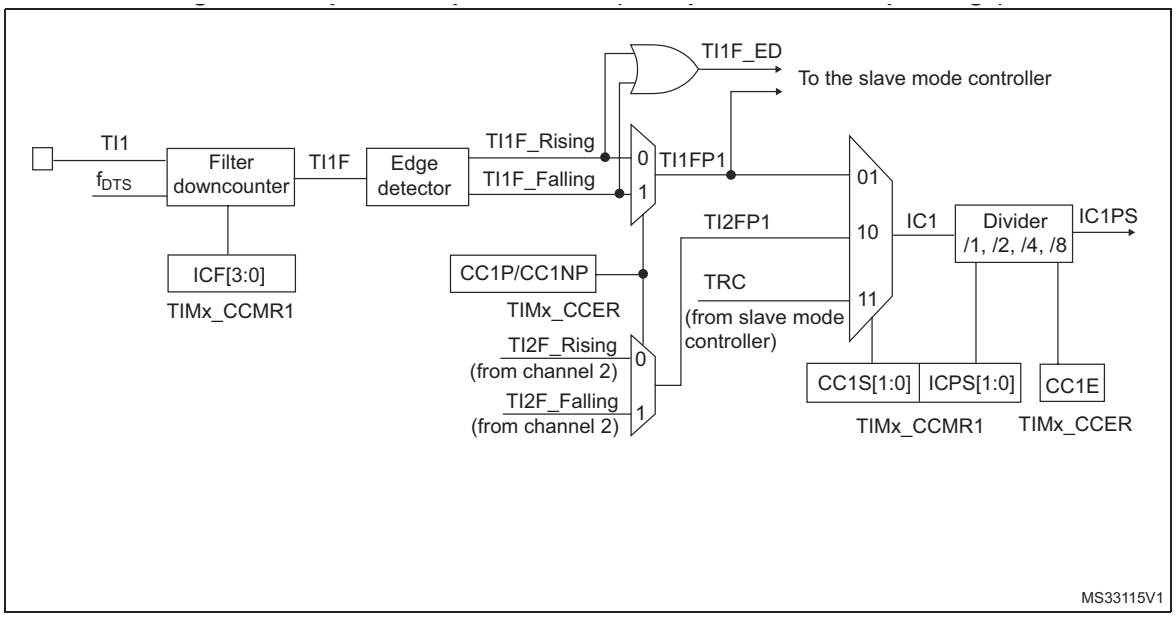


18.3.4. Каналы захвата/сравнения

Каждый канал захвата/сравнения построен вокруг парного регистра (включая невидимый), входного каскада захвата (с цифровым фильтром, мультиплексором и предделителем) и выходного каскада (с компаратором и управлением выходом).

Входной каскад считывает нужный вход TIx для выдачи фильтрованного сигнала TIxF. Далее, детектор фронта с выбором полярности выдаёт сигнал (TIxFPx), что может быть использован как вход запуска контроллером режима ведомого или как команда захвата. Предделитель стоит до регистра захвата (ICxPS).

Рис. 159. Входной каскад канала 1.



Выходной каскад выдаёт промежуточный сигнал, используемый как опорный: OCxRef (активный высокий). Полярность действует в конце цепочки.

Рис. 160. Основная схема канала 1.

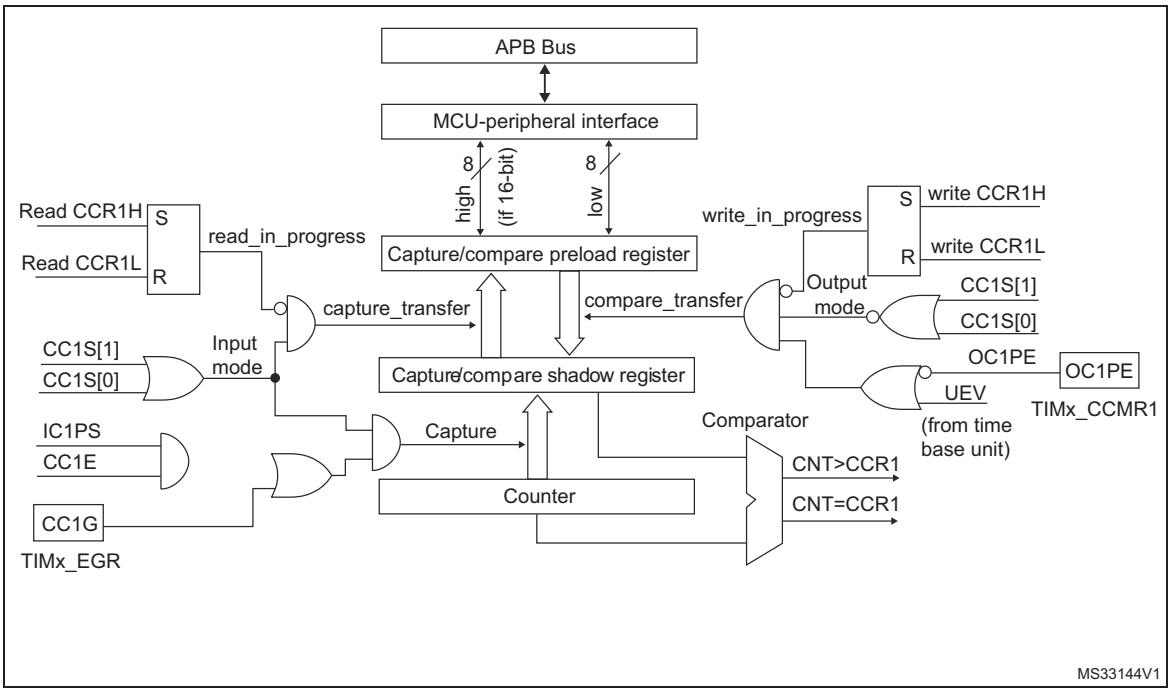
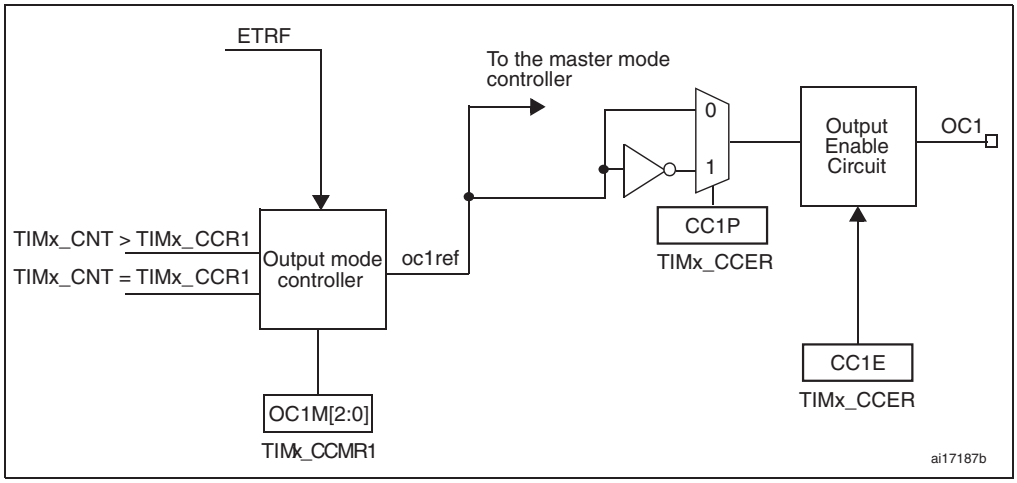


Рис. 161. Выходной каскад канала захвата/сравнения 1.



Блок захвата/сравнения состоит из двух регистров: предзагрузки и невидимого. Снаружи доступен только регистр предзагрузки и по чтению и по записи.

Захват выполняется внутренним регистром и результат пишется в регистр предзагрузки.

При сравнении регистр предзагрузки пишется в невидимый, где и сравнивается со счётчиком.

18.3.5. Режим захвата входа

Регистры `TIMx_CCRx` защёлкивают содержимое счётчика после передачи, обнаруженной соответствующим сигналом `ICx`. При этом ставится флаг `CCXIF` в регистре `TIMx_SR` и ставятся запрос DMA и прерывания. Если захват происходит при стоящем флаге `CCxIF`, то ставится флаг перезахвата `CCxOF` в регистре `TIMx_SR`. Флаг `CCxIF` снимается записью в него нуля или чтением захваченных данных из регистра `TIMx_CCRx`. `CCxOF` снимается записью ‘`0`’.

Далее захватываем содержимое `TIMx_CCR1` по переднему фронту `TI1`. Делаем так:

- Выбираем активный вход: `TIMx_CCR1` подключаем в `TI1` и пишем "`01`" в биты `CC1S` регистра `TIMx_CCMR1`. Биты `CC1S` отличаются от `00`, т. е. канал на вводе и регистр `TIMx_CCR1` доступен только по чтению.
- Ставим длительность входного фильтра битами `ICxF` в регистре `TIMx_CCMRx` с учётом входного сигнала на `TIx`. Допустим, что при переключении входной сигнал нестабилен в течении пяти внутренних тактов. Значит длительность фильтра должна быть больше этих пяти тактов. Мы можем определить передачу на `TI1` за 8 последовательных выборок нового уровня на частоте f_{DTS}. И мы пишем `0011` в биты `IC1F` регистра `TIMx_CCMR1`.
- Выбираем фронт сигнала на `TI1` записью `0` в бит `CC1P` регистра `TIMx_CCER` (передний).

- Ставим предделитель. Здесь он нам не нужен (пишем 00 в биты IC1PS регистра TIMx_CCMR1).
- Разрешаем запись счётчика в регистр захвата установкой бита CC1E в TIMx_CCER.
- Если нужно разрешаем выдачу запросов прерывания битом CC1IE в TIMx_DIER и DMA битом CC1DE в регистре TIMx_DIER.

Если захват входа произошёл, то:

- Регистр TIMx_CCR1 получает значение счётчика активной передачи.
- Ставится флаг прерывания CC1IF. Если он ещё стоял, то ставится и флаг CC1OF.
- В зависимости от бита CC1IE генерируется прерывание.
- В зависимости от бита CC1DE генерируется запрос DMA.

Чтобы обработать перезахват рекомендуется читать регистр данных до опроса флага перезахвата.

NB: Запросы прерывания и DMA IC можно выдавать программно установкой битов CCxG в регистре TIMx_EGR.

18.3.6. Режим ввода ШИМ

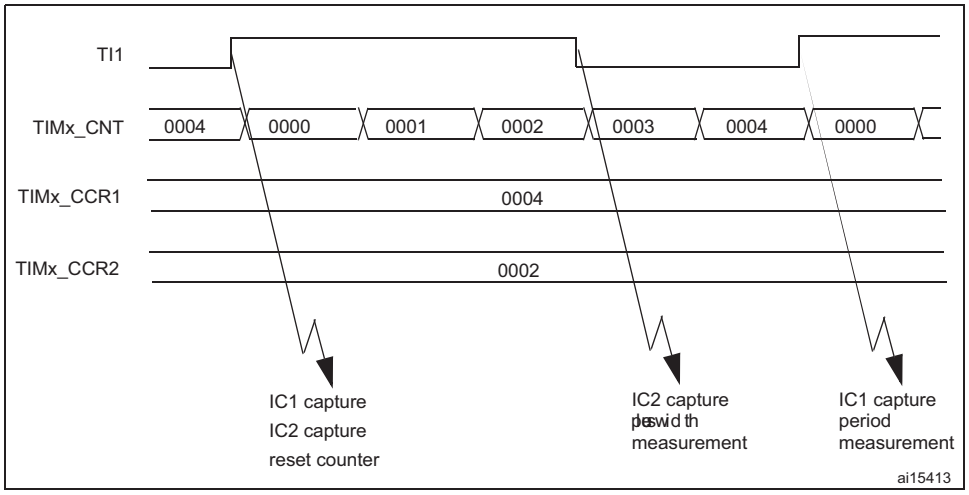
Это часть режима захвата входа. Процедура такая же, кроме:

- Два сигнала ICx направляются на на один вход TIx.
- Полярность этих входов ICx противоположная.
- Сигналом запуска берётся один из двух сигналов TIxFP, а контроллер ведомого режима выключается.

Например, можно измерить период (в регистре TIMx_CCR1) и длительность заполнения (в регистре TIMx_CCR2) ШИМ сигнала на TI1 следующей процедурой (зависит от частоты CK_INT и предделителя):

- Выбрать активный вход для TIMx_CCR1: Записать 01 в биты CC1S регистра TIMx_CCMR1 (TI1).
- Выбрать полярность для TI1FP1 (для захвата в TIMx_CCR1 и очистки счётчика): записать 0 в CC1P (передний фронт).
- Выбрать активный вход для TIMx_CCR2: записать 10 в биты CC2S регистра TIMx_CCMR1 (TI1).
- Выбрать полярность для TI1FP2 (для захвата в TIMx_CCR2): записать 1 в CC2P (задний фронт).
- Выбрать вход запуска: записать 101 в биты TS регистра TIMx_SMCR (TI1FP1).
- Поставить контроллер режима ведомого в сброс: записать 100 в биты SMS регистра TIMx_SMCR.
- Включить захват: поставить биты CC1E и CC2E в регистре TIMx_CCER.

Рис. 162. Времянка режима ввода ШИМ.



1. Режим ШИМ может использоваться только с сигналами TIMx_CH1/TIMx_CH2 так как к контроллеру режима ведомого подключены только TI1FP1 и TI2FP2.

18.3.7. Принудительный вывод

В режиме вывода (биты CCxS = 00 в TIMx_CCMRx), каждый выходной сигнал сравнения (OCxREF и затем OCx/OCxN) можно программно поставить в активное или неактивное состояние, независимо от результата сравнения выходного регистра сравнения и счётчика.

Активными выходные сигналы (OCxREF/OCx) ставятся записью 101 в биты OCxM нужного регистра TIMx_CCMRx. Активный сигнал OCxREF всегда высокий, а OCx противоположен биту полярности CCxP.

Пример: CCxP=0 (OCx активный высокий) => OCx ставится высоким.

Сигнал OCxREF снимается записью 100 в биты OCxM регистра TIMx_CCMRx.

Но сравнение внутреннего регистра TIMx_CCRx со счётчиком всё равно выполняется, биты ставятся, генерируются запросы прерывания и DMA. См. ниже.

18.3.8. Режим сравнения выхода

Используется для управления выходным сигналом и определения истечения периода времени.

При совпадении регистра захвата/сравнения со счётчиком эта схема:

- Ставит выходную ножку в заданное состояние (биты OCxM в регистре TIMx_CCMRx) и полярность (бит CCxP в регистре TIMx_CCER). Ножка может хранить состояние(OCxM=000), стать активной (OCxM=001), неактивной (OCxM=010) или переключиться (OCxM=011).
- Ставит флаг прерывания (бит CCxIF в регистре TIMx_SR).
- При установленной маске прерывания (бит CCxIE в регистре TIMx_DIER) выдаёт запрос.
- При установленном разрешении DMA (бит CCxDE в регистре TIMx_DIER) выдаёт нужный (бит CCDS в регистре TIMx_CR2) запрос.

В регистр TIMx_CCRx можно писать как через регистр предзагрузки, так и без него (см. бит OCxPE в регистре TIMx_CCMRx).

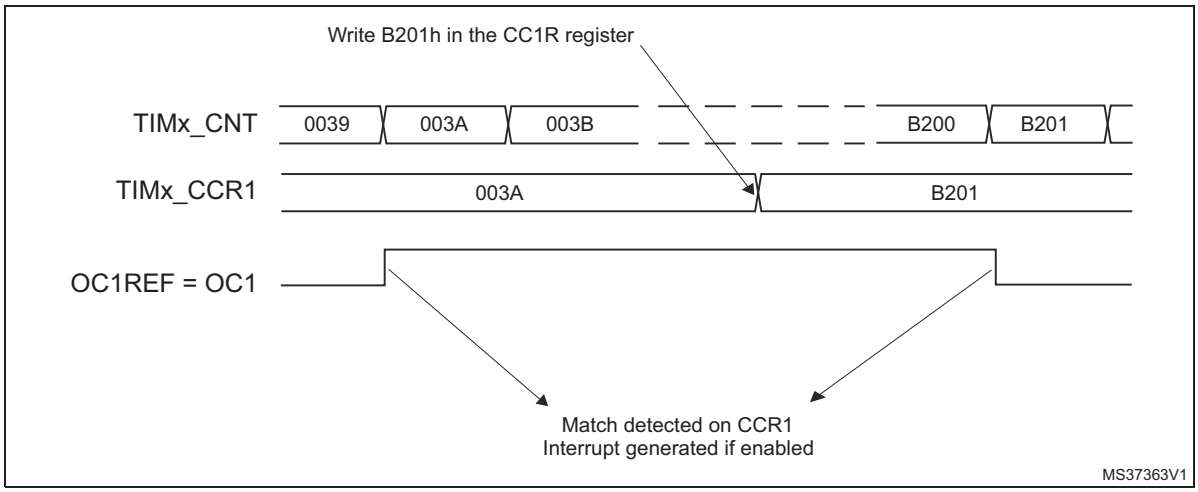
В режиме сравнения выхода бит события UEV на выходы OCxREF и OCx не влияет. Разрешение составляет один счёт счётчика. Этот режим можно использовать для выдачи одного импульса.

Процедура:

1. Выбрать такты счётчика (внутренние, внешние, предделитель).
2. Записать желаемое в регистры TIMx_ARR и TIMx_CCRx.
3. Если нужно прерывание, то установить бит CCxIE.
4. Выбрать режим выхода. Например:
   - Записать OCxM = 011 для переключения ножки OCx при совпадении CNT и CCRx
   - Записать OCxPE = 0 для отключения регистра предзагрузки
   - Записать CCxP = 0 ради высокого активного уровня
   - Записать CCxE = 1 для разрешения вывода
5. Включить счётчик установкой бита CEN в регистре TIMx_CR1.

При запрещённом регистре предзагрузки (OCxPE='0') регистр TIMx_CCRx можно обновлять в любое время, иначе он будет обновлён только по событию UEV.

Рис. 163. Режим сравнения выхода, переключение ножки OC1.



18.3.9. Режим ШИМ

Частота сигнала ШИМ определяется регистром TIMx_ARR, а коэффициент заполнения регистром TIMx_CCRx.

Режим ШИМ может включаться независимо по одному на каждый выход OCx записью ‘110’ (ШИМ1) или ‘111’ (ШИМ2) в биты OCxM регистра TIMx_CCMRx. Соответствующий регистр предзагрузки должно включать битом OCxPE в регистре TIMx_CCMRx и, соответсвенно, регистр авто-загрузки (в режимах прямого и реверсивного счёта) нужно включить битом ARPE в регистре TIMx_CR1.

Поскольку регистр предзагрузки пишется в теневой регистр только по событию обновления, то все регистры нужно заранее инициализировать установкой бита UG в регистре TIMx_EGR.

Полярность OCx (активный высокий или низкий) выбирается битом CCxP в регистре TIMx_CCER. Выходы OCx включаются битами CCxE, CCxNE, MOE, OSSI и OSSR в регистрах TIMx_CCER и TIMx_BDTR.

В режиме ШИМ (1 или 2), TIMx_CNT и TIMx_CCRx сравниваются на выполнение условия TIMx_CCRx ≤ TIMx_CNT или TIMx_CNT ≤ TIMx_CCRx (в зависимости от направления счёта).

Но для совместимости с ETRF (OCREF очищается внешним сигналом ETR до следующего периода ШИМ), сигнал OCREF выставляется только:

- когда изменяется результат сравнения, или
- при переключении сравнения выхода (биты OCxM в регистре TIMx_CCMRx register) из “заморозки” (без сравнения, OCxM=‘000) в один из режимов ШИМ (OCxM=‘110 или ‘111).

Это программное управление ШИМ при работающем счётчике.

Бит CMS в регистре TIMx_CR1 определяет генерацию ШИМ по фронту или по центру.

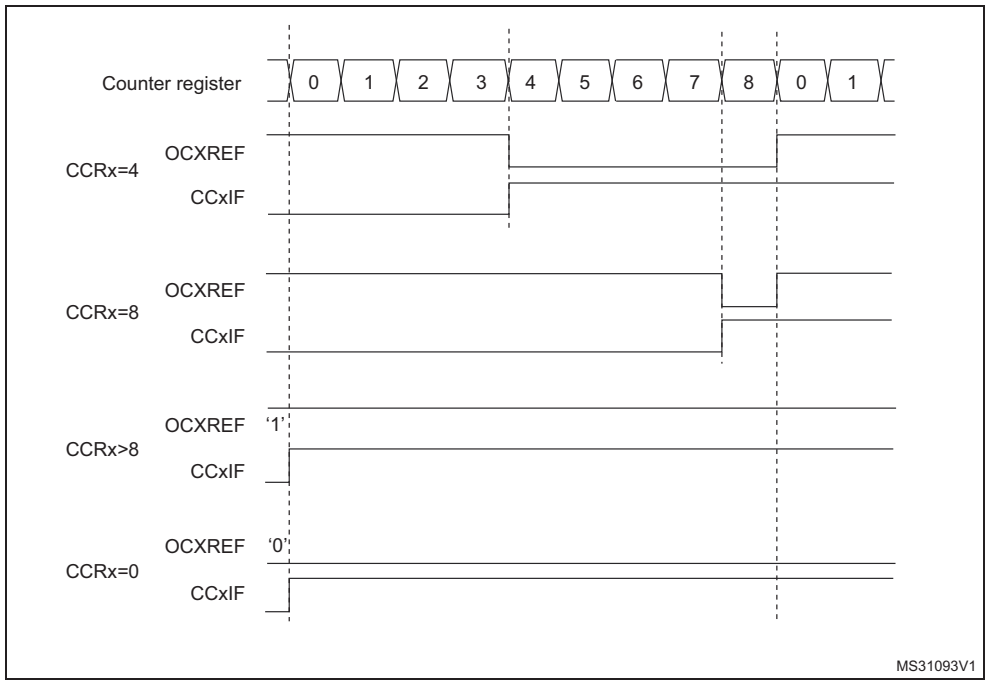
ШИМ по фронту

Прямой счёт

Он определяется сброшенным битом DIR в регистре TIMx_CR1.

В примере ниже стоит режим ШИМ1. Опорный сигнал OCxREF остаётся высоким при TIMx_CNT < TIMx_CCRx. Если регистр TIMx_CCRx больше значения авто-загрузки (TIMx_ARR) то OCxREF равен ‘1’. При равенстве значений сравнения, OCxREF равен ‘0’. В примере ниже TIMx_ARR=8.

Рис. 164. ШИМ по фронту (TIMx_ARR=8).



Обратный счёт

Он определяется стоящим битом DIR в регистре TIMx_CR1.

В режиме ШИМ1 опорный сигнал OCxREF остаётся низким при TIMx_CNT > TIMx_CCRx. Если регистр TIMx_CCRx больше значения авто-загрузки (TIMx_ARR) то OCxREF равен ‘1’. При равенстве значений сравнения, OCxREF равен ‘1’. 0% ШИМ в этом режиме недоступен.

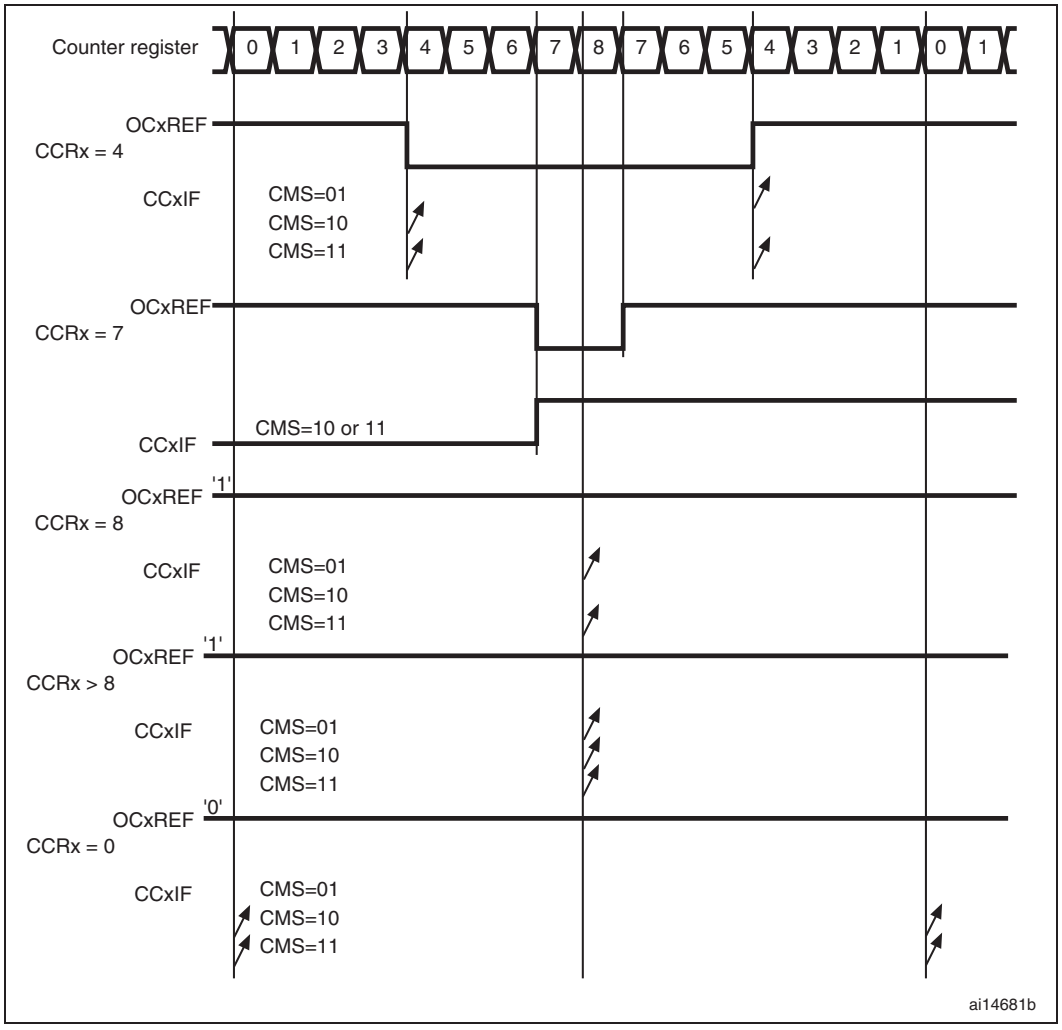
ШИМ по центру

Этот режим включается когда биты CMS в регистре TIMx_CR1 не равны ‘00’ (остальные конфигурации на сигналы OCxREF/OCx влияют также). Флаг сравнения ставится в любых режимах счёта (прямом, обратном и реверсивном) в зависимости от битов CMS в регистре TIMx_CR1. Бит направления (DIR) в регистр TIMx_CR1 меняется аппаратно и не дай бог менять его программно.

В примере ниже:

- TIMx_ARR=8,
- Стоит режим ШИМ1,
- Флаг ставится при обратном счёте в режиме ШИМ1 по центру (CMS=01 в TIMx_CR1).

ШИМ по центру (TIMx_ARR=8).



Советы:

- При старте режима по центру используется текущее направление счёта (зависит от бита DIR в регистре TIMx_CR1). Более того, в это время биты DIR и CMS программно менять нельзя.
- Запись в счётчик работающего режима ой как не рекомендуется. В частности:
  - При TIMx_CNT>TIMx_ARR направление счёта не изменяется и продолжает считать в том же направлении.
  - При записи 0 или изменении TIMx_ARR направление изменяется, но событие UEV не выдаётся.
- Самый безопасный способ это программно запустить обновление (битом UG в регистре TIMx_EGR) прямо перед стартом, а не писать в работающий счётчик.

18.3.10.Режим одного импульса (OPM)

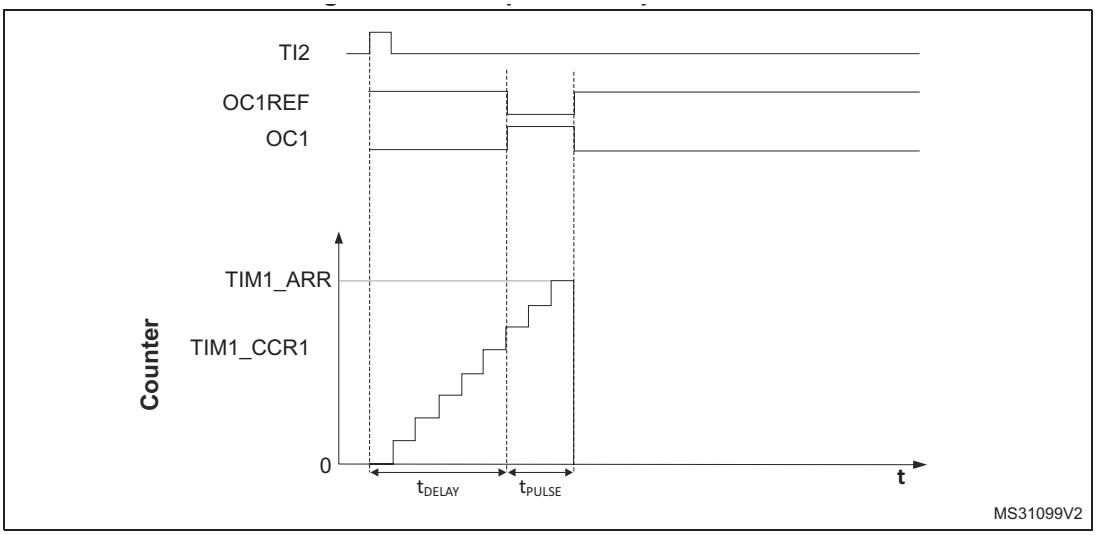
OPM это частный случай предыдущих режимов. Он позволяет счётчику запускаться по сигналу и генерировать импульс программируемой длины с программируемой задержкой.

Запускать счётчик можно из контроллера ведомого режима. Создавать сигнал можно в режиме сравнения выхода и ШИМ. Режим включается установкой бита OPM в регистре TIMx_CR1. Так счётчик автоматически останавливается по следующему событию UEV.

Импульс генерируется правильно только если значение сравнения отличается от начального значения счётчика. Перед стартом (таймер ждёт запуска) конфигурация должна быть:

- При прямом счёте: CNT < CCRx ≤ ARR (в частности, 0 < CCRx)
- При обратном счёте: CNT > CCRx

Рис. 166. Режим одного импульса.



Например, выдадим положительный импульс на OC1 длиной t_{PULSE} после задержки t_{DELAY} по переднему фронту на ножке TI2.

Берём TI2FP2 как запуск 1:

- Поставим TI2FP2 на TI2 записью CC2S=’01’ в регистре TIMx_CCMR1.
- TI2FP2 ставим на передний фронт записью CC2P=’0’ в регистре TIMx_CCER.
- Выбираем TI2FP2 как запуск контроллером режима ведомого (TRGI) записью TS=’110’ в регистр TIMx_SMCR.
- TI2FP2 используем как старт записью SMS=‘110’ в TIMx_SMCR (режим запуска).

Сигнал OPM определяем записью регистров сравнения (учитываем частоту тактов и предделитель).

- Определяем t_{DELAY} записью в TIMx_CCR1.
- Ставим t_{PULSE} как разность значений перезагрузки и сравнения (TIMx_ARR - TIMx_CCR1+1).
- Сигнал создаётся переходом из ‘0’ в ‘1’ при сравнении и переходом из ‘1’ в ‘0’ при достижении счётчиком значения перезагрузки. Для этого, включаем режим ШИМ2 записью OC1M=111 в регистре TIMx_CCMR1. Регистры предзагрузки можно включить записью OC1PE=’1’ в регистре TIMx_CCMR1и ARPE в регистре TIMx_CR1, а можно не включать. В этом случае значение сравнения пишется в регистр TIMx_CCR1, значение перезагрузки в регистр TIMx_ARR, даём событие обновления битом UG и ждём внешнего запуска на TI2. В CC1P здесь пишется ‘0’.

В этом примере биты DIR и CMS регистра TIMx_CR1 должны быть сброшены.

Мы ждём только одного импульса (Однократный режим), так что для остановки счётчика по следующему событию обновления (счётчик переходит через значение перезагрузки и обнуляется) нужно поставить бит OPM регистра TIMx_CR1. Если бит OPM регистра TIMx_CR1 обнулён, то включается Повторяющийся режим.

Частный случай: быстрое включение OCx:

В этом режиме появление фронта на входе TIx ставит бит CEN, чем включает счётчик. Затем совпадение счётчика с значением сравнения переключает выходной сигнал. Но для этого нужны несколько тактов, ограничивая минимально возможное значение t_{DELAY}.

Если нужен сигнал с минимальной задержкой, то нужно ставить бит OCxFE в TIMx_CCMRx. Далее включаются OCxREF (и OCx) не глядя на сравнение. Его новый уровень равен уровню при сравнении. OCxFE работает только в режимах ШИМ1 и ШИМ2.

18.3.11.Очистка OCxREF по внешнему событию

Сигнал OCxREF для заданных каналов можно сделать низким, подав высокий уровень на вход ETRF (бит разрешения OCxCE в TIMx_CCMRx стоит в ‘1’). Он остаётся низким вплоть до следующего события UEV.

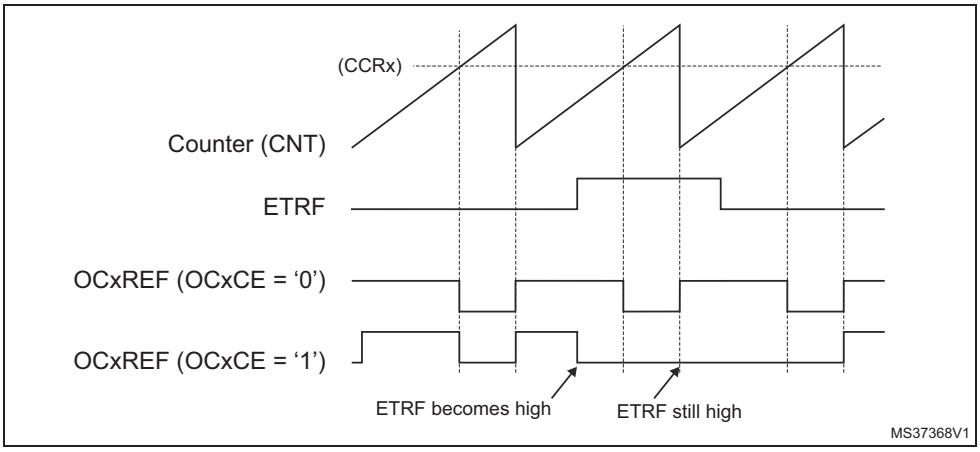
Эта функция работает в режимах сравнения выхода и ШИМ, но не в принудительном режиме.

Например, сигнал ETR можно подключить к выходу компаратора для управления током. В этом случае ETR нужно конфигурировать так:

1. Выключить предделитель внешнего запуска: записать ’00’ в биты ETPS[1:0] в TIMx_SMCR.
2. Выключить режим 2 внешнего тактирования: снять бит ECE в TIMx_SMCR.
3. Поставить полярность и фильтр внешнего запуска (ETP и ETF) как нужно.

В примере ниже приведён сигнал OCxREF при переходе входа ETRF в высокий уровень для обоих значений OCxCE. Здесь TIMx в режиме ШИМ.

Рис. 167. Очистка OCxREF.



18.3.12.Режим интерфейса кодера

Режим интерфейса кодера выбирается записью SMS=‘001’ в TIMx_SMCR при счёте по фронтам TI2, SMS=’010’ по фронтам TI1 и SMS=’011’ по фронтам TI1 и TI2.

Полярность TI1 и TI2 выбирается битами CC1P и CC2P в регистре TIMx_CCER. Если нужно, то можно использовать входной фильтр.

Для работы с инкрементным кодером используются входы TI1 и TI2. Счётчик тактируется каждой достоверной передачей по TI1FP1 или TI2FP2 (TI1 и TI2 после входного фильтра и выбора полярности, без них TI1FP1= TI1 и TI2FP2= TI2) при стоящем бите CEN в TIMx_CR1. Последовательность сигналов с двух входов создаёт импульсы счёта и сигнал направления. Счёт может быть прямым и обратным, бит DIR в TIMx_CR1 изменяется аппаратно. Бит DIR вычисляется на каждый входной сигнал (TI1 и/или TI2).

Режим интерфейса кодера просто работает по внешним тактам с выбором направления. То есть счётчик непрерывно считает туда-сюда между 0 и перезагружаемым значением в регистре TIMx_ARR. Так что TIMx_ARR нужно писать до старта. Таким способом захват, сравнение, предделитель и запуск вывода продолжают нормально. Режим кодера и Режим внешних тактов 2 несовместимы.

В этом режиме счётчик автоматически следует за скоростью и направлением инкрементного кодера и его содержимое всегда представляет позицию кодера. Направление счёта соответствует направлению вращения подключённого датчика.

Далее приведены возможные комбинации, полагая, что TI1 и TI2 не изменяются одновременно.

Таблица 97. Направления счёта.

Активный сигнал	Уровень оппозитного сигнала (TI1FP1 для TI2, TI2FP2 для TI1)	Фронт TI1FP1		Фронт TI2FP2	
		Передний	Задний	Передний	Задний

Только TI1	Высокий	Обратный	Прямой	Стоит	Стоит
	Низкий	Прямой	Обратный	Стоит	Стоит
Только TI2	Высокий	Стоит	Стоит	Прямой	Обратный
	Низкий	Стоит	Стоит	Обратный	Прямой
TI1 и TI2	Высокий	Обратный	Прямой	Прямой	Обратный
	Низкий	Прямой	Обратный	Обратный	Прямой

Внешний инкрементный кодер может напрямую подключаться к MCU без дополнительной логики. Но обычно для преобразования дифференциального сигнала в цифровой используются компараторы. Это сильно увеличивает помехозащищённость. Третий выход кодера, показывающий механическую нулевую позицию, можно подключить к входу внешнего прерывания и сбросу счётчика.

Пример работы счётчика показывает генерацию сигнала счёта и направления. Также показано как компенсируется дребезг на обоих выбранных фронтах. Он может появиться при нахождении датчика возле точек переключения. Использована такая конфигурация:

- CC1S=’01’ (TIMx_CCMR1, TI1FP1 на TI1).
- CC2S=’01’ (TIMx_CCMR2, TI1FP2 на TI2).
- CC1P=’0’, and IC1F = ‘0000’ (TIMx_CCER, TI1FP1 без инверсии, TI1FP1=TI1).
- CC2P=’0’, and IC2F = ‘0000’ (TIMx_CCER, TI1FP2 без инверсии, TI1FP2= TI2).
- SMS=’011’ (TIMx_SMCR, оба входа по обоим фронтам).
- CEN=’1’ (TIMx_CR1, счётчик включён).

Рис. 168. Работа счётчика.

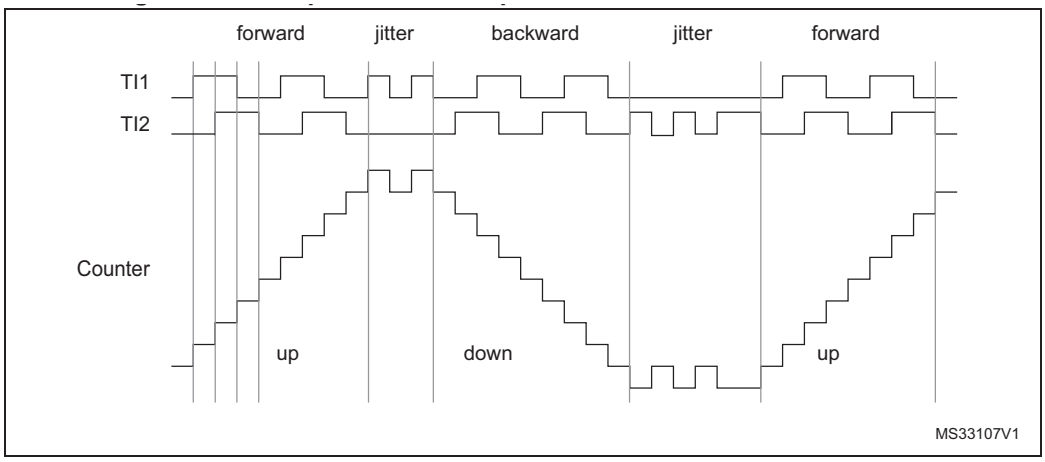
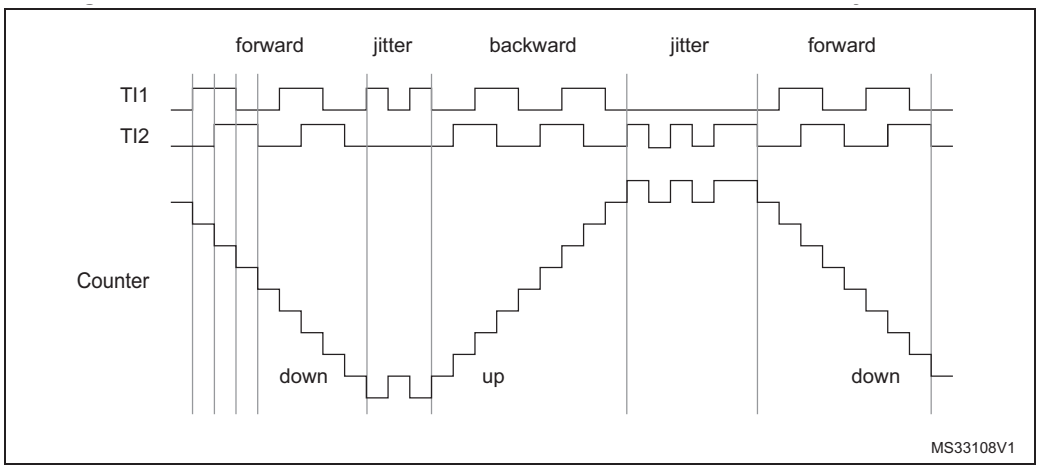


Рис. 169. Работа счётчика с инверсным TI1FP1 и CC1P=’1’.



Таймер в режиме интерфейса кодера показывает текущую позицию счётчика. С помощью второго таймера в режиме захвата, измеряя время между двумя событиями кодера, можно получить динамическую информацию (скорость, ускорение, торможение). Для этого можно использовать третий выход кодера, показывающего механический ноль. В зависимости от времени между двумя

событиями счётчик можно читать и регулярно. Это делается регулярной записью счётчика в третий входной регистр (если есть) периодическим сигналом от другого таймера. Можно читать и запросом DMA от часов реального времени.

18.3.13.Функция XOR входов таймера

Бит TI1S в регистре TIMx_CR2 позволяет подключать к входу канала 1 выход вентиля XOR, сочетая три входа TIMx_CH1, TIMx_CH2 and TIMx_CH3.

Выход XOR можно использовать со всеми входными функциями таймера, вроде запуска или захвата.

18.3.14.Синхронизация внешнего запуска TIMx

Есть три режима синхронизации TIMx: Сброс, Вентильный и Запуск.

Режим ведомого: Сброс

Счётчик и предделитель можно инициализировать по входу запуска. Кроме того, если бит URS в TIMx_CR1 сброшен, то генерируется событие UEV. Далее обновляются все предзагружаемые регистры (TIMx_ARR, TIMx_CCRx).

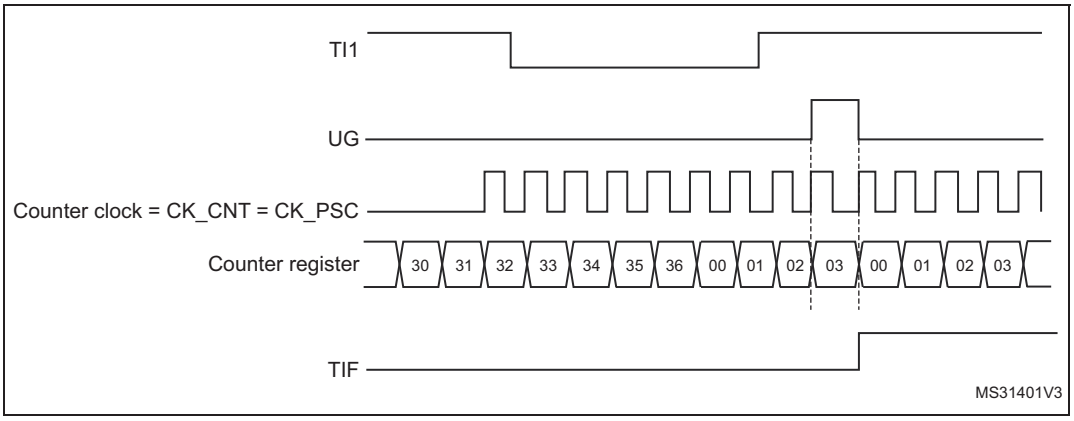
В этом примере счётчик прямого счёта сбрасывается по переднему фронту на входе TI1:

- Ставим канал 1 на передний фронт TI1. Входной фильтр отключён (IC1F=0000). Предделителя нет. Биты CC1S=01 в регистре TIMx_CCMR1 выбирают захват входа. Бит CC1P=0 в регистре TIMx_CCER определяет полярность и передний фронт.
- Ставим таймер в режим сброса записью SMS=100 в регистр TIMx_SMCR. Выбираем вход с TI1 записью TS=101 в регистре TIMx_SMCR.
- Запускаем счётчик записью CEN=1 в регистре TIMx_CR1.

Счётчик начинает считать внутренние такты вплоть переднего фронта TI1. Тогда счётчик сбрасывается с 0, ставится флаг запуска (бит TIF в регистре TIMx_SR) выдаются запросы прерывания и DMA (если разрешены битами TIE и TDE в регистре TIMx_DIER).

На рисунке ниже регистр перезагрузки TIMx_ARR=0x36. Задержка между фронтом TI1 и действительным сбросом появляется из-за схемы ресинхронизации на входе TI1.

Рис. 170. Режим Сброса.



Режим ведомого: Вентильный

Счётчик включается уровнем на выбранном входе.

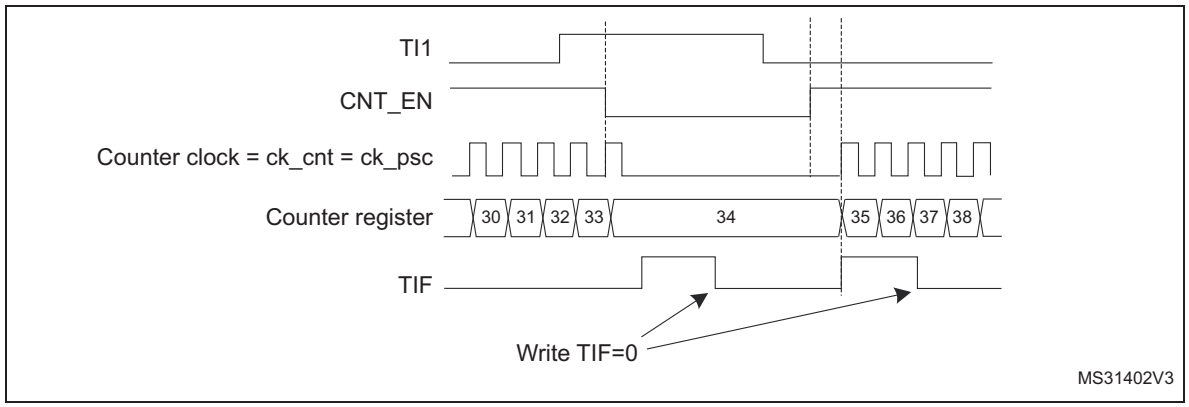
В этом примере счётчик прямого счёта работает только при низком уровне TI1:

- Ставим канал 1 на низкий уровень TI1. Входной фильтр отключён (IC1F=0000). Предделителя нет. Биты CC1S=01 в регистре TIMx_CCMR1 выбирают захват входа. Бит CC1P=1 в регистре TIMx_CCER определяет полярность и низкий уровень.
- Ставим таймер в вентильный режим записью SMS=101 в регистр TIMx_SMCR. Выбираем вход с TI1 записью TS=101 в регистре TIMx_SMCR.
- Запускаем счётчик записью CEN=1 в регистре TIMx_CR1. В этом режиме счётчик не работает при CEN=0, независимо от уровня на входе.

Счётчик начинает считать внутренние такты при низком уровне на TI1 и останавливается при высоком. Флаг запуска (бит TIF в регистре TIMx_SR) ставится при пуске и остановке таймера.

Задержка между фронтом TI1 и действительным сбросом появляется из-за схемы ресинхронизации на входе TI1.

Рис. 171. Вентильный режим.



Режим ведомого: Запуск

Счётчик включается событием на выбранном входе.

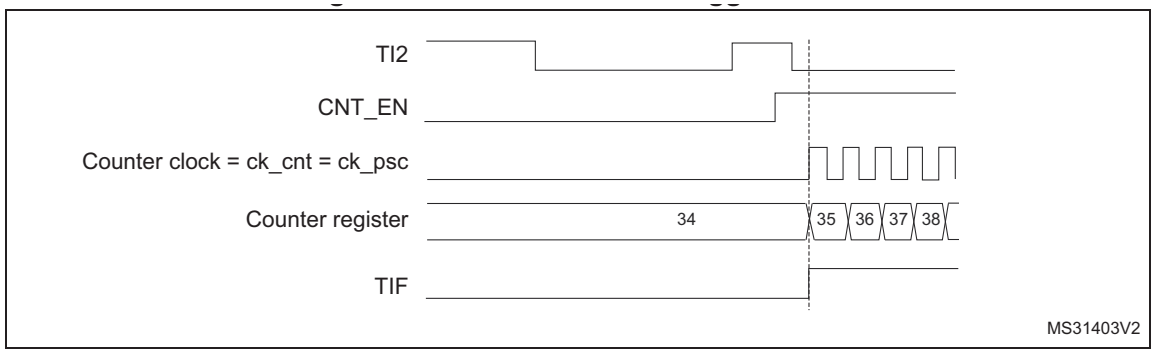
В этом примере счётчик прямого счёта работает по переднему фронту на TI2:

- Ставим канал 2 на передний фронт TI2. Входной фильтр отключён (IC1F=0000). Предделителя нет. Биты CC2S=01 в регистре TIMx_CCMR1 выбирают захват входа. Бит CC2P=1 в регистре TIMx_CCER определяет полярность и низкий уровень.
- Ставим таймер в режим запуска записью SMS=110 в регистр TIMx_SMCR. Выбираем вход с TI2 записью TS=110 в регистре TIMx_SMCR.

Счётчик начинает считать внутренние такты по переднему фронту на TI2 и ставит флаг запуска (бит TIF в регистре TIMx_SR).

Задержка между фронтом TI2 и действительным сбросом появляется из-за схемы ресинхронизации на входе TI2.

Рис. 172. Режим Запуска.



Режим ведомого: Внешние такты 2 + Запуск

К режимам ведомого (кроме Внешних тактов 1 и Кодера) можно добавить режим внешнего тактирования 2. В этом случае сигнал ETR выбирается для внешних тактов, а по другому входу можно подавать запуск (в режимах Сброса, Вентильном или Запуска). Не рекомендуем выбирать ETR как TRGI битами TS в регистре TIMx_SMCR.

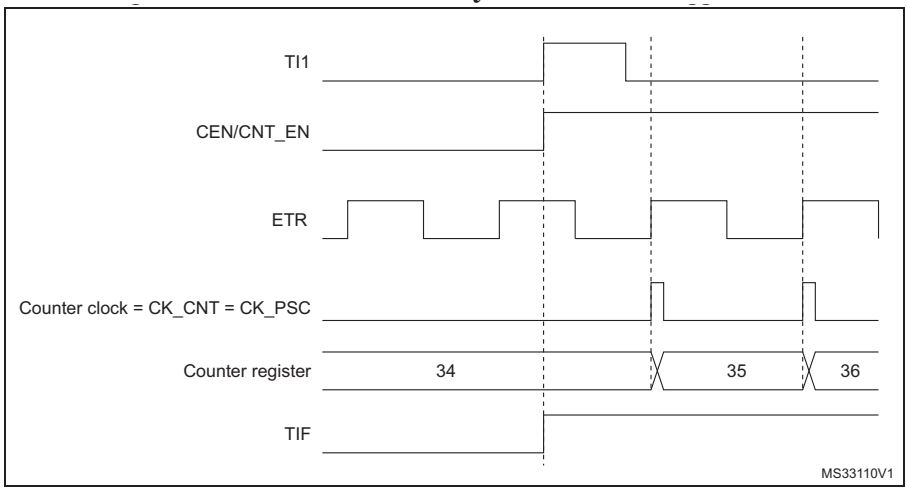
В этом примере счётчик прямого счёта работает по переднему фронту на ETR после переднего фронта запуска на TI1:

1. Ставим внешний запуск в регистре TIMx_SMCR:
   - ETF = 0000: фильтра нет; ETPS = 00: предделитель выключен; ETP = 0: передний фронт на ETR и ECE=1 для внешних тактов 2.
2. Ставим канал 1 на передний фронт TI:
   - IC1F=0000: фильтра нет; предделитель выключен, ничего не делаем; CC1S=01 в TIMx_CCMR1 для захвата входа; CC1P=0 в TIMx_CCER для полярности (и только переднего фронта).
3. Ставим таймер в режим запуска записью SMS=110 в регистр TIMx_SMCR. Выбираем вход с TI1 записью TS=101 в регистре TIMx_SMCR.

Передний фронт TI1 разрешает счёт и ставит флаг TIF. Считаются передние фронты ETR.

Задержка между фронтом ETR и действительным сбросом появляется из-за схемы ресинхронизации на входе ETRP.

Рис. 173. Внешние такты 2 + Запуск.

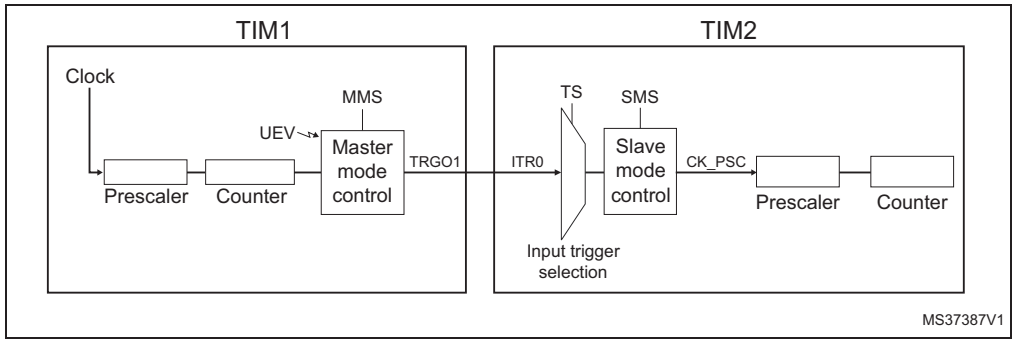


18.3.15.Синхронизация таймера

Таймеры TIM связаны между собой для синхронизации или сцепления. Таймер в режиме Ведущего может сбрасывать, запускать, останавливать другой таймер в режиме Ведомого.

NB: Такты ведомого таймера нужно включать до получения событий от ведущего и не должны меняться налету.

Рис. 174. Таймер как предделитель другого.



Здесь TIM1 работает предделителем TIM2:

- Ставим TIM1 ведущим для выдачи периодического сигнала запуска по каждому событию UEV. При MMS=010 в регистре TIM1_CR2 передний фронт появляется на выходе TRGO1.
- Подключаем выход TRGO1 от TIM1 идёт на TIM2, стоящий в режиме ведомого с внутренним запуском по ITR0. Ставится битами TS=000 в регистре TIM2_SMCR.
- Ставим контоллер ведомого в режим внешних тактов 1 (SMS=111 в регистре TIM2_SMCR). Отныне TIM2 тактируется передним фронтом сигнала переполнения TIM1.
- Наконец-то включаем оба таймера их битами CEN в регистрах TIMx_CR1.

NB: Если на TIM1 OCx стоит внешним запуском (MMS=1xx), то его передний фронт тактирует счётчик TIM2.

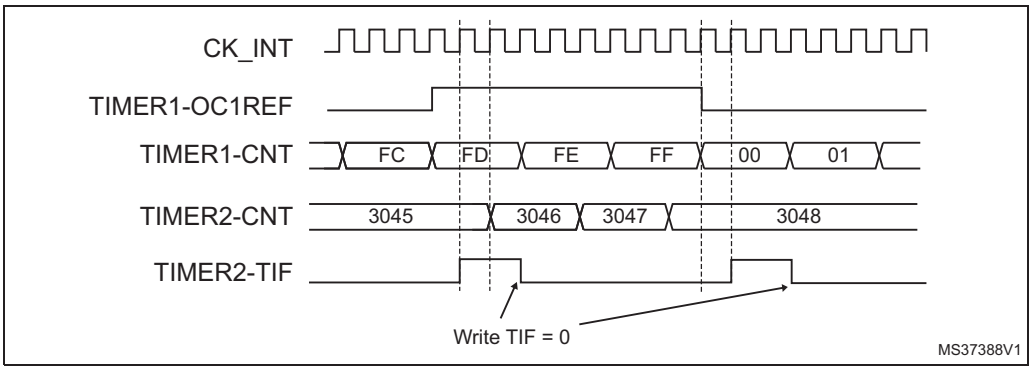
Один таймер разрешает другой.

В этом примере мы разрешаем TIM2 выходом сравнения TIM1. TIM2 считает предделённые внутренние такты только при высоком OC1REF от TIM1. Оба счётчика тактируются после предделителя CK_INT ($f_{CK\_CNT} = f_{CK\_INT}/3$).

- Ведущий TIM1 посылает OC1REF на выход как запуск (MMS=100 в TIM1_CR2).
- Конфигурируем OC1REF у TIM1 (регистр TIM1_CCMR1).
- TIM2 получает запуск от TIM1 (TS=000 в регистре TIM2_SMCR).
- Ставим TIM2 в вентильный режим (SMS=101 в регистре TIM2_SMCR).
- Разрешаем TIM2 установкой бита CEN (регистр TIM2_CR1).
- Пускаем TIM1 установкой бита CEN (регистр TIM1_CR1).

NB: Такты счётчика 2 не синхронизированы со счётчиком 1, режим влияет только на разрешение счётчика TIM2.

Рис. 175. Вентильный TIM2 по OC1REF от TIM1.

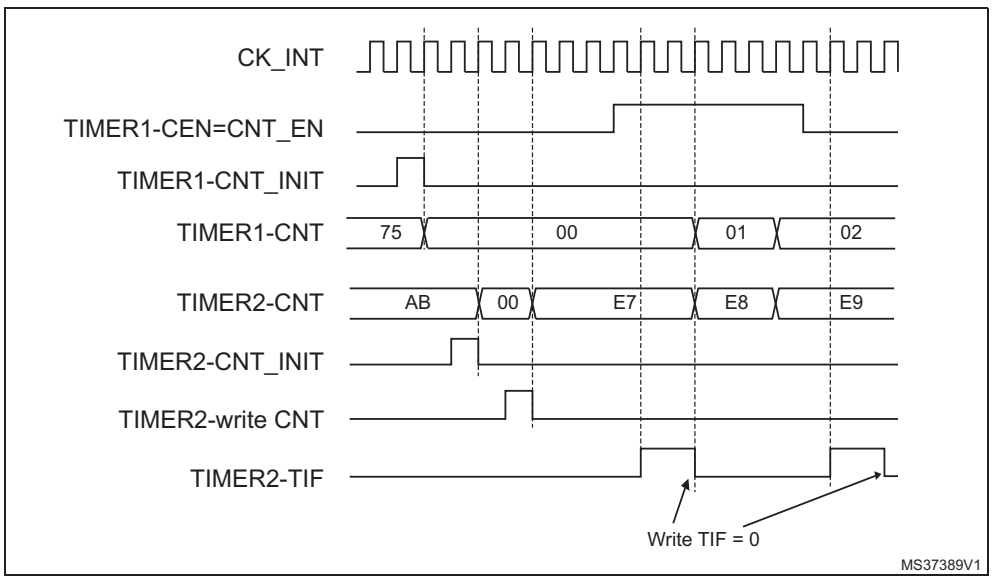


Здесь счётчик и предделитель TIM2 не инициализируются, а работают с текущего состояния. Для этого оба таймера сбрасываются до старта TIM1. Затем можно записать в счётчики любое значение. Сбросить таймеры можно программно битом UG в регистрах TIMx_EGR.

В следующем примере синхронизируем TIM1 и TIM2. Ведущий TIM1 начинает с 0. Ведомый TIM2 начинает с 0xE7. Оба предделителя совпадают. TIM2 2 останавливается при выключении TIM1 записью '0' в бит CEN регистра TIM1_CR1:

- Ведущий TIM1 посылает OC1REF на выход как запуск (MMS=100 в TIM1_CR2).
- Конфигурируем OC1REF у TIM1 (регистр TIM1_CCMR1).
- TIM2 получает запуск от TIM1 (TS=000 в регистре TIM2_SMCR).
- Ставим TIM2 в вентильный режим (SMS=101 в регистре TIM2_SMCR).
- Сбрасываем TIM1 записью '1' в бит UG (регистр TIM1_EGR).
- Сбрасываем TIM2 записью '1' в бит UG (регистр TIM2_EGR).
- Пишем 0xE7 в счётчик TIM2 (TIM2_CNTL).
- Разрешаем TIM2 установкой бита CEN (регистр TIM2_CR1).
- Пускаем TIM1 установкой бита CEN (регистр TIM1_CR1).
- Останавливаем TIM1 снятием бита CEN (регистр TIM1_CR1).

Рис. 176. Вентильный TIM2 с разрешением от TIM1.



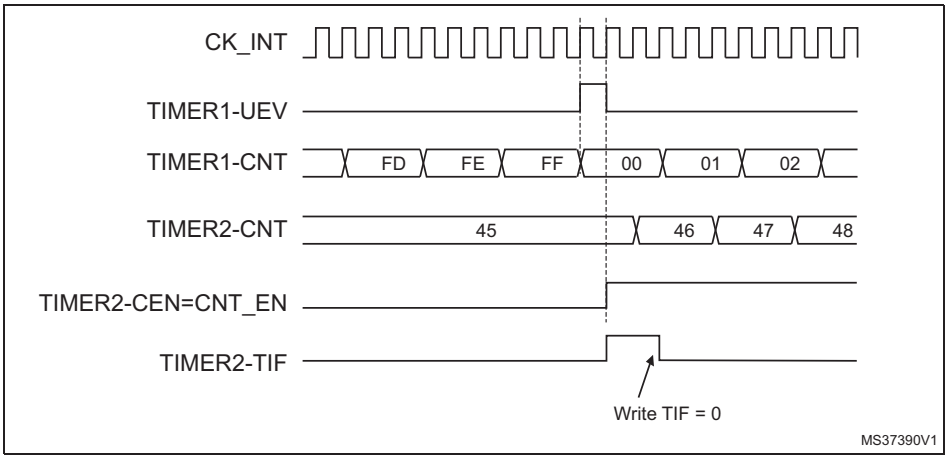
Запуск одного таймера от другого.

В этом примере мы разрешаем TIM2 событием обновления TIM1. TIM2 начинает счёт с текущего состояния (не нулевого) предделёнными внутренними тактами только по событию обновления от TIM1. Когда TIM2 получает сигнал запуска, его бит CEN автоматически встаёт и счётчик работает вплоть до записи ‘0 в бит CEN регистра TIM2_CR1. Оба счётчика тактируются после предделителя CK_INT ($f_{CK\_CNT} = f_{CK\_INT}/3$).

- Ведущий TIM1 посылает UEV на выход как запуск (MMS=010 в TIM1_CR2).
- Ставим период TIM1 (регистр TIM1_ARR).

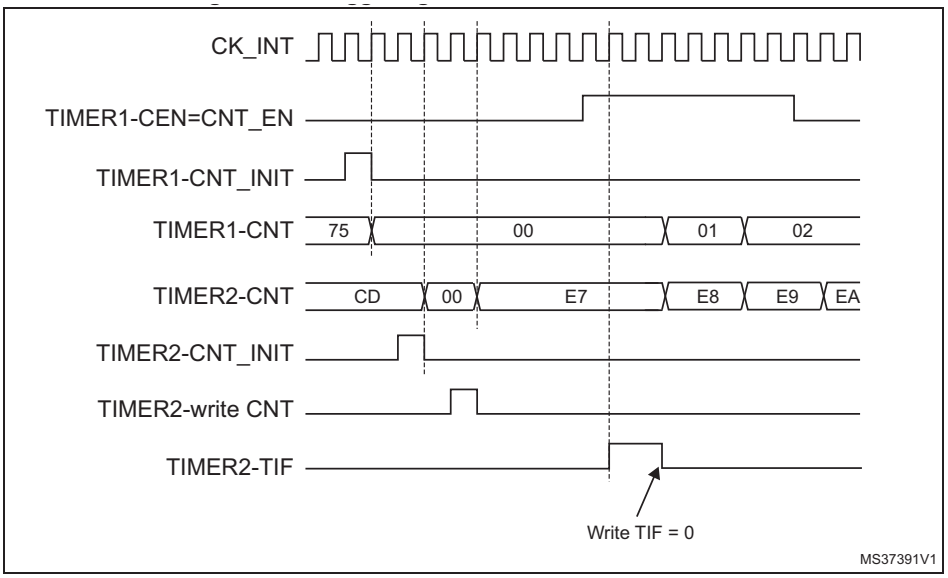
- TIM2 получает запуск от TIM1 (TS=000 в регистре TIM2_SMCR).
- Ставим TIM2 в режим запуска (SMS=110 в регистре TIM2_SMCR).
- Пускаем TIM1 установкой бита CEN (регистр TIM1_CR1).

Рис. 177. Запуск TIM2 событием обновления от TIM1.



Как и в предыдущем примере, можно инициализировать оба счётчика до старта.

Рис. 178. Запуск TIM2 разрешением TIM1.



Синхронный старт 2 таймеров по внешнему запуску.

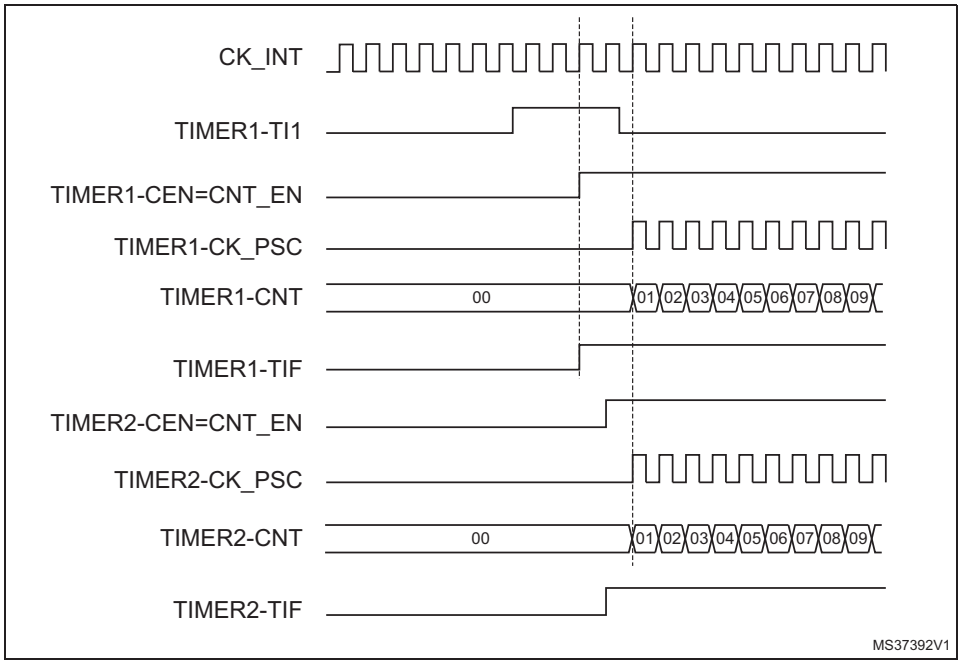
В этом примере передний фронт на входе TI1 TIM1 разрешает и TIM1 и TIM2. Для выравнивания счётчиков TIM1 нужно ставить в режим Ведущий/Ведомый (ведомый для TI1, ведущий для TIM2):

- Ставим TIM1 ведущим для посылки его Разрешения на выход запуска (MMS=001 в TIM1_CR2).
- Ставим TIM1 ведомым для получения запуска от TI1 (TS=100 в TIM1_SMCR).
- Ставим TIM1 в режим запуска (SMS=110 в TIM1_SMCR).
- Ставим TIM1 Ведущим/Ведомым записью MSM=1 (регистра TIM1_SMCR).
- Ставим TIM2 на получение запуска от TIM1 (TS=000 в TIM2_SMCR).
- Ставим TIM2 в режим запуска (SMS=110 в TIM2_SMCR).

По переднему фронту на TI1 (TIM1) оба счётчика начинают работать синхронно по внутренним тактам и оба ставят флаги TIF.

NB: В этом примере оба таймера инициализируются до старта установкой битов UG. Оба счётчика начинают с 0, но можно установить смещение записью в их регистры счётчиков (TIMx_CNT). Ниже видно задержку между CNT_EN и CK_PSC TIM1.

Рис. 179. Запуск TIM1 и TIM2 сигналом TI1 TIM1.



18.3.16.Режим отладки

В режиме отладки (ядро Cortex-M3 остановлено), счётчик TIMx либо работает, либо останавливается, в зависимости от бита DBG_TIMx_STOP в модуле DBG.

При стоящем счётчике (DBG_TIMx_STOP = 1 в регистре DBGMCU_APBx_FZ) выходы отключаются (как при сброшенном бите MOE). Выходы можно перевести в неактивное состояние (бит OSSI = 1), или возложить управление на контроллер GPIO (бит OSSI = 0) для перевода их в высокоимпедансное состояние (Hi-Z).

18.4. Регистры TIM1 и TIM5

32-бит регистры надо писать словами. Остальные регистры надо писать полусловами или словами. Читать можно байтами, полусловами или словами.

18.4.1. Регистр управления 1 (TIMx_CR1)

Смещение адреса: 0x00

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:10 Резерв, не трогать.

— Биты 9:8 **CKD[1:0]:** Деление тактов.

Это отношение тактов таймера (CK_INT) и тактов задержки и выборки (t_{DTS}) в генераторах задержки и цифровых фильтрах (ETR, TIx),

00: $t_{DTS}=t_{CK\_INT}$

01: $t_{DTS}=2*t_{CK\_INT}$

10: $t_{DTS}=4*t_{CK\_INT}$

11: Резерв

— Бит 7 **ARPE**: Разрешение буфера перезагрузки TIMx_ARR

0: Нельзя

1: Нужно

— Бит 6:5 **CMS[1:0]**: Выбор режима по центру

00: По фронту. Прямой или обратный счёт в зависимости от бита направления (DIR).

01: По центру 1. Попеременный прямой и обратный счёт. Флаг сравнения выхода выводных каналов (CCxS=00 в TIMx_CCMRx) ставится только при обратном счёте.

10: По центру 2. Попеременный прямой и обратный счёт. Флаг сравнения выхода выводных каналов (CCxS=00 в TIMx_CCMRx) ставится только при прямом счёте.

11: По центру 3. Попеременный прямой и обратный счёт. Флаг сравнения выхода выводных каналов (CCxS=00 в TIMx_CCMRx) ставится и при прямом и при обратном счёте.

NB: При включённом таймере (CEN=1) переключать режим по фронту в режим по центру нельзя.

— Бит 4 **DIR**: Направление счёта.

0: Прямой

1: Обратный

NB: Бит читается только в режимах По центру и Кодера.

— Бит 3 **OPM**: Режим одного импульса

0: Счётчик не останавливается

1: Счётчик останавливается по следующему событию обновления (снимается бит CEN)

— Бит 2 **URS**: Источник запроса по событию UEV.

Изменяется программно.

0: Разрешённые запросы прерывания или DMA выдаются по:

– Переполнение/Исчерпание счётчика

– Установка бита UG

– Обновление от контроллера режима ведомого

1: Разрешённые запросы прерывания или DMA выдаются только по Переполнению/Исчерпанию счётчика.

— Бит 1 **UDIS:** Выключение выдачи события обновления UEV.

Изменяется программно.

0: Событие UEV выдаётся по:

– Переполнение/Исчерпание счётчика

– Установка бита UG

– Обновление от контроллера режима ведомого

1: Событие UEV не выдаётся, скрытые регистры (ARR, PSC, CCRx) хранят значение. При стоящем бите UG или аппаратном сбросе от контроллера режима ведомого счётчик и предделитель инициализируются.

— Бит 0 **CEN:** Включение счётчика.

0: Выключен

1: Включён

NB: Режимы Внешнего тактирования, Вентильный и Кодера работают только при заранее установленном бите CEN. Режим Запуска ставит его автоматически. В режиме Одного импульса бит CEN снимается аппаратно по событию обновления.

18.4.2. Регистр управления 2 (TIMx_CR2)

Смещение адреса: 0x04

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TI1S	MMS[2:0]			CCDS	Reserved		
								rw	rw	rw	rw	rw			

— Биты 15:8 Резерв, не трогать.

— Бит 7 **TI1S:** Выбор ножки для TI1

0: Ножка TIMx_CH1

1: Ножки TIMx_CH1, CH2 и CH3 (через XOR)

— Биты 6:4 **MMS[2:0]:** Режим ведущего

Выбор информации для синхронизации ведомых таймеров (TRGO):

000: **Сброс** - Бит UG в регистре TIMx_EGR. Если на входе запуска идёт сигнал сброса (контроллер режима ведомого в сбросе), то сигнал на TRGO задерживается относительно реального сброса.

001: **Разрешение** - Сигнал разрешения счётчика (CNT_EN). Это полезно для одновременного запуска нескольких таймеров или управления окном ведомого таймера. Сигнал разрешения это объединение по OR бита CEN с входом запуска вентильного режима.

При подаче CNT_EN по входу запуска сигнал TRGO задерживается, кроме режима ведущий/ведомый (см. бит MSM в регистре TIMx_SMCR).

010: **Обновление** - На выход запуска (TRGO) направлено событие обновления. Например, ведущий таймер может быть предделителем ведомого.

011: **Импульс сравнения** - После захвата или совпадения на выход запуска (TRGO) посылается положительный импульс установки флага CC1IF (даже если он стоит).

100: **Сравнение** - Сигнал OC1REF

101: **Сравнение** - Сигнал OC2REF

110: **Сравнение** - Сигнал OC3REF

111: **Сравнение** - Сигнал OC4REF

NB: Тактирование ведомого таймера и ADC нужно включать до получения событий от ведущего таймера и не должно меняться налету.

— Бит 3 **CCDS:** Выбор DMA Захвата/Сравнения

0: Запрос CCx DMA посылается по событию CCx

1: Запросы CCx DMA посылаются по событию обновления

— Биты 2:0 Резерв, не трогать.

18.4.3. Регистр управления режима ведомого (TIMx_SMCR)

Смещение адреса: 0x08

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			Res.	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	Res.	rw	rw	rw

— Бит 15 **ETP**: Полярность внешнего запуска

0: ETR не инверсный, активен высоким или передним фронтом.

1: ETR инверсный, активен низким или задним фронтом.

— Бит 14 **ECE**: Разрешение внешнего тактирования

0: Внешнее тактирование 2 выключено

1: Внешнее тактирование 2 включено. Счётчик работает от активного фронта ETRF.

NB:

1: Установка бита ECE равнозначна Внешнему тактированию 1 с подключением TRGI к ETRF (SMS=111 и TS=111).

2: Внешнее тактирование 2 можно использовать одновременно с режимами ведомого: Сброс, Вентильный и Запуска. Тем не менее, TRGI нельзя подключать к ETRF (TS не равен 111).

3: If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.

— Биты 13:12 **ETPS[1:0]**: Предделитель внешнего запуска

Частота сигнала ETRP должна быть не больше 1/4 частоты TIMxCLK. Полезно при измерении быстрых внешних сигналов.

00: Предделитель Выкл.

01: ETRP / 2

10: ETRP / 4

11: ETRP / 8

— Биты 11:8 **ETF[3:0]**: Фильтр внешнего запуска

Определяет частоту выборки и длину цифрового фильтра сигнала ETRP. Цифровой фильтр это счётчик, выдающий сигнал на выход после N последовательных событий на входе:

0000: Без фильтра, выборка на частоте f_{DTS}

0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2

0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4

0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8

0100: $f_{SAMPLING}=f_{DTS}/2$, N=6

0101: $f_{SAMPLING}=f_{DTS}/2$, N=8

0110: $f_{SAMPLING}=f_{DTS}/4$, N=6

0111: $f_{SAMPLING}=f_{DTS}/4$, N=8

1000: $f_{SAMPLING}=f_{DTS}/8$, N=6

1001: $f_{SAMPLING}=f_{DTS}/8$, N=8

1010: $f_{SAMPLING}=f_{DTS}/16$, N=5

1011: $f_{SAMPLING}=f_{DTS}/16$, N=6

1100: $f_{SAMPLING}=f_{DTS}/16$, N=8

1101: $f_{SAMPLING}=f_{DTS}/32$, N=5

1110: $f_{SAMPLING}=f_{DTS}/32$, N=6

1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

— Бит 7 **MSM**: Режим Ведущий/Ведомый

0: Ничего

1: Действие входа запуска (TRGI) задерживается для полной синхронизации текущего таймера и его ведомых (через TRGO). Это полезно при синхронизации нескольких таймеров от одного внешнего события.

— Биты 6:4 **TS[2:0]**: Выбор запуска

000: Внутренний 0 (ITR0)

001: Внутренний 1 (ITR1)

010: Внутренний 2 (ITR2)

011: Внутренний 3 (ITR3)

100: Детектор фронта TI1 (TI1F_ED)

101: Фильтрованный вход 1 (TI1FP1)

110: Фильтрованный вход 2 (TI2FP2)

111: Внешний запуск (ETRF)

NB: Менять их можно только если не используются (например, при SMS=000) во избежание неверного определения фронта.

— Бит 3 Резерв, не трогать.

— Биты 2:0 **SMS[2:0]**: Выбор режима ведомого

При внешнем сигнале активный фронт TRGI подключается к внешнему входу с выбранной полярностью.

000: Режим ведомого выключен - если CEN = '1', то предделитель работает напрямую от внутренних тактов.

001: Режим кодера 1 - Прямой/обратный счёт по фронту TI2FP1 в зависимости от уровня TI1FP2.

010: Режим кодера 2 - Прямой/обратный счёт по фронту TI1FP2 в зависимости от уровня TI2FP1.

011: Режим кодера 3 - Прямой/обратный счёт по фронтам TI1FP1 и TI2FP2 в зависимости от уровня другого входа.

100: Режим сброса - Передний фронт TRGI инициализирует счётчик и запускает обновление регистров.

101: Вентильный режим - Такты счётчика разрешены при высоком TRGI. При низком TRGI счётчик останавливается, но не сбрасывается. Старт и стоп счётчика управляемые.

110: Режим запуска - Счётчик стартует по переднему фронту TRGI (но не сбрасывает). Управляемый только старт.

111: Внешние такты 1 - Счёт передних фронтов TRGI.

NB: Вентильный режим нельзя использовать если входом запуска выбран TI1F_ED (TS='100'). В действительности, TI1F_ED выдаёт 1 импульс на каждую передачу TI1F, тогда как вентильный режим проверяет уровень сигнала запуска.

Такты ведомого таймера нужно включать до приёма событий от ведущего таймера и не должны изменяться налету.

Таблица 98. Подключение внутреннего запуска TIMx.

Ведомый TIM	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM2	TIM1	TIM8	TIM3	TIM4
TIM3	TIM1	TIM2	TIM5	TIM4
TIM4	TIM1	TIM2	TIM3	TIM8
TIM5	TIM2	TIM3	TIM4	TIM8

18.4.4. Регистр разрешения прерываний/DMA (TIMx_DIER)

Смещение адреса: 0x0C

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	Res	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw		rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

Содержимое битов:

0: Нельзя

1: Можно

— Бит 15 Резерв, не трогать.

— Бит 14 **TDE**: Запрос DMA запуска

— Бит 13 Резерв, не трогать.

— Бит 12 **CC4DE**: Запрос DMA захвата/сравнения 4
— Бит 11 **CC3DE**: Запрос DMA захвата/сравнения 3
— Бит 10 **CC2DE**: Запрос DMA захвата/сравнения 2
— Бит 9 **CC1DE**: Запрос DMA захвата/сравнения 1
— Бит 8 **UDE**: Запрос DMA обновления
— Бит 7 Резерв, не трогать.
— Бит 6 **TIE**: Запрос прерывания запуска
— Бит 5 Резерв, не трогать.
— Бит 4 **CC4IE**: Запрос прерывания захвата/сравнения 4
— Бит 3 **CC3IE**: Запрос прерывания захвата/сравнения 3
— Бит 2 **CC2IE**: Запрос прерывания захвата/сравнения 2
— Бит 1 **CC1IE**: Запрос прерывания захвата/сравнения 1
— Бит 0 **UIE**: Запрос прерывания обновления

18.4.5. Регистр состояния (TIMx_SR)

Биты ставятся аппаратно, стираются записью 0.
Перезахват это появление события при стоящем его флаге.
Смещение адреса: 0x10
По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CC4OF	CC3OF	CC2OF	CC1OF	Reserved		TIF	Res	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Содержимое битов:
0: Не было
1: Было

— Биты 15:13 Резерв, не трогать.
— Бит 12 **CC4OF**: Флаг перезахвата захвата/сравнения 4
— Бит 11 **CC3OF**: Флаг перезахвата захвата/сравнения 3
— Бит 10 **CC2OF**: Флаг перезахвата захвата/сравнения 2
— Бит 9 **CC1OF**: Флаг перезахвата захвата/сравнения 1
— Бит 8:7 Резерв, не трогать.
— Бит 6 **TIF**: Флаг прерывания запуска
Ставится аппаратно по активному фронту на входе TRGI при работающем контроллере режима ведомого во всех режимах, кроме Вентильного, (тогда по обоим фронтам).
— Бит 5 Резерв, не трогать.
— Бит 4 **CC4IF**: Флаг прерывания захвата/сравнения 4
— Бит 3 **CC3IF**: Флаг прерывания захвата/сравнения 3
— Бит 2 **CC2IF**: Флаг прерывания захвата/сравнения 2
— Бит 1 **CC1IF**: Флаг прерывания захвата/сравнения 1
В режиме вывода CC1:
Есть исключения для выравнивания по центру (см. биты CMS в регистре TIMx_CR1).
В режиме ввода CC1:
— Бит 0 **UIF**: Флаг прерывания обновления
Удержание прерывания обновления ставится при:
– Переполнении или исчерпании счётчика и нулевом счётчике и UDIS=0 в регистре TIMx_CR1.
– При программной инициализации CNT битом UG в регистре TIMx_EGR, если URS=0 и UDIS=0 в регистре TIMx_CR1.
– При инициализации CNT событием запуска (см. регистр TIMx_SMCR), если URS=0 и UDIS=0 в регистре TIMx_CR1.

18.4.6. Регистр генерации событий (TIMx_EGR)

Событие генерируется программной установкой бита, снимается он аппаратно.
Смещение адреса: 0x14
По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
									w		w	w	w	w	w

— Биты 15:7 Резерв, не трогать.

— Бит 6 **TG**: Запуск.

1: Ставит флаг TIF в регистре TIMx_SR. Могут возникнуть прерывание и запрос DMA.

— Бит 5 Резерв, не трогать.

— Бит 4 **CC4G**: Захват/сравнение 4.

— Бит 3 **CC3G**: Захват/сравнение 3.

— Бит 2 **CC2G**: Захват/сравнение 2.

— Бит 1 **CC1G**: Захват/сравнение 1.

1: Выдаёт событие захвата/сравнения:

В режиме вывода CC1:

Ставит флаг CC1IF. Могут возникнуть прерывание и запрос DMA.

В режиме ввода CC1:

Текущее значение счётчика пишется в TIMx_CCR1. Ставит флаг CC1IF. Могут возникнуть прерывание и запрос DMA. При уже стоящем флаге CC1IF ставится и флаг CC1OF.

— Бит 0 **UG**: Обновление.

1: Инициализирует счётчик и обновление регистров. Текущий счётчик предделителя чистится не влияя на само значение. В режиме по центру или при DIR=0 счётчик обнуляется, иначе перегружается из регистра TIMx_ARR (при DIR=1).

18.4.7. Регистр режима захвата/сравнения 1 (TIMx_CCMR1)

Смещение адреса: 0x18

По сбросу: 0x0000

Каналы могут работать на ввод (захват) и вывод (сравнение), что определяется битами `CCxS`. У остальных битов для ввода и вывода значение отличается. `OCxx` обозначает функции вывода, `ICxx` функции ввода. При этом используются разные каскады.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				IC1F[3:0]				IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Сравнение выхода:

— Бит 15 **OC2CE:** Разрешение очистки сравнения 2.

— Биты 14:12 **OC2M[2:0]**: Режим сравнения выхода 2.

— Бит 11 **OC2PE**: Разрешение предзагрузки сравнения 2.

— Бит 10 **OC2FE**: Быстрое разрешение сравнения 2.

— Биты 9:8 **CC2S[1:0]**: Выбор Захвата/Сравнения 2.

Определяет направление канала (ввод/вывод) и используемый вход.

00: Канал CC2 выходной

01: Канал CC2 входной, IC2 на TI2

10: Канал CC2 входной, IC2 на TI1

11: Канал CC2 входной, IC2 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.

NB: Биты CC2S можно писать только при выключенном канале (CC2E = ‘0’ в TIMx_CCER).

— Бит 7 **OC1CE**: Разрешение очистки сравнения выхода 1.

0: ETRF на OC1REF не влияет

1: Высокий уровень на ETRF стирает OC1REF

— Биты 6:4 **OC1M[2:0]**: Режим сравнения выхода 1.

Определяют поведение выходного опорного сигнала OC1REF, от которого происходят OC1 и OC1N. OC1REF активен высоким, а OC1 и OC1N зависят от битов CC1P и CC1NP.

000: Заморозка - сравнение TIMx_CCR1 и TIMx_CNT на выходы не влияет (режим используется для генерации временной базы).

001: При совпадении TIMx_CNT и TIMx_CCR1 сигнал OC1REF ставится активным (высоким).

010: При совпадении TIMx_CNT и TIMx_CCR1 сигнал OC1REF ставится неактивным (низким).

011: Переключение - при TIMx_CNT=TIMx_CCR1 сигнал OC1REF перебрасывается.

100: Принудительный неактивный - OC1REF ставится низким.

101: Принудительный активный - OC1REF ставится высоким.

110: ШИМ 1 - при прямом счёте канал активен пока TIMx_CNT<TIMx_CCR1. При обратном счёте канал неактивен (OC1REF='0') пока TIMx_CNT>TIMx_CCR1.

111: ШИМ 2 - при прямом счёте канал неактивен пока TIMx_CNT<TIMx_CCR1. При обратном счёте канал активен пока TIMx_CNT>TIMx_CCR1.

NB:

1: Эти биты нельзя изменить пока стоит LOCK уровень 3 (биты LOCK в регистре TIMx_BDTR) и CC1S='00' (канал включён на выход).

2: В ШИМ 1 и 2 уровень OCREF изменяется только когда изменяется результат сравнения или при переключении из "Заморозки" в "ШИМ".

— Бит 3 **OC1PE**: Разрешение регистра предзагрузки.

0: Предзагрузка TIMx_CCR1 выключена. TIMx_CCR1 можно писать в любое время, новое значение начинает действовать незамедлительно.

1: Предзагрузка TIMx_CCR1 включена. Доступ идёт к регистру предзагрузки. TIMx_CCR1 пишется в активный регистр по событию обновления.

NB:

1: Эти биты нельзя изменить пока стоит LOCK уровень 3 (биты LOCK в регистре TIMx_BDTR) и CC1S='00' (канал включён на выход).

2: Режим ШИМ без установки регистра предзагрузки можно использовать только в режиме одного импульса (стоит бит OPM в регистре TIMx_CR1). Иначе поведение непредсказуемо.

— Бит 2 **OC1FE**: Разрешение быстрого сравнения.

Ускоряет действие события запуска по входу на выход CC.

0: CC1 работает как обычно (сравнение счётчика и CCR1) даже при включённом запуске. Минимальная задержка переключения выхода CC1 при фронте на входе составляет 5 тактов.

1: Активный фронт запуска по входу действует как сравнение на выходе CC1. Тогда OC ставится в уровень сравнения независимо от его результата. Задержка между импульсом на входе и выходом CC1 сокращается до 3 тактов. OCFE работает только в ШИМ 1 и ШИМ 2.

— Биты 1:0 **CC1S**: Выбор Захвата/Сравнения 1.

Определяет направление канала (ввод/вывод) и используемый вход.

00: Канал CC1 выходной

01: Канал CC1 входной, IC1 на TI1

10: Канал CC1 входной, IC1 на TI2

11: Канал CC1 входной, IC1 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.

NB: Биты CC1S можно писать только при выключенном канале (CC1E = '0' в TIMx_CCER).

Захват входа.

— Биты 15:12 **IC2F**: Фильтр захвата входа 2.

— Биты 11:10 **IC2PSC[1:0]** : Предделитель захвата входа 2.

— Биты 9:8 **CC2S[1:0]**: Выбор Захвата/Сравнения 2.

Определяет направление канала (ввод/вывод) и используемый вход.

00: Канал CC2 выходной

01: Канал CC2 входной, IC2 на TI2

10: Канал CC2 входной, IC2 на TI1

11: Канал CC2 входной, IC2 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.

NB: Биты CC2S можно писать только при выключенном канале (CC2E = '0' в TIMx_CCER).

— Биты 7:4 **IC1F[3:0]**: Фильтр захвата входа 1.

Определяет частоту выборки и длину цифрового фильтра сигнала TI1. Цифровой фильтр это счётчик, выдающий сигнал на выход после N последовательных событий на входе:

0000: Без фильтра, выборка на частоте f_{DTS}

0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2

0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4

0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8

0100: $f_{SAMPLING}=f_{DTS}/2$, N=6

0101: $f_{SAMPLING}=f_{DTS}/2$, N=8

0110: $f_{SAMPLING}=f_{DTS}/4$, N=6

0111: $f_{SAMPLING}=f_{DTS}/4$, N=8

1000: $f_{SAMPLING}=f_{DTS}/8$, N=6
1001: $f_{SAMPLING}=f_{DTS}/8$, N=8
1010: $f_{SAMPLING}=f_{DTS}/16$, N=5
1011: $f_{SAMPLING}=f_{DTS}/16$, N=6
1100: $f_{SAMPLING}=f_{DTS}/16$, N=8
1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

— Биты 3:2 **IC1PSC**: Предделитель захвата входа 1
Значение предделителя на входе CC1 (IC1).
По CC1E=’0’ (регистр TIMx_CCER) предделитель сбрасывается.
00: без предделителя, по каждому фронту на входе
01: каждые 2 события
10: каждые 4 события
11: каждые 8 события

— Биты 1:0 **CC1S**: Выбор Захвата/Сравнения 1.
Определяет направление канала (ввод/вывод) и используемый вход.
00: Канал CC1 выходной
01: Канал CC1 входной, IC1 на TI1
10: Канал CC1 входной, IC1 на TI2
11: Канал CC1 входной, IC1 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.
NB: Биты CC1S можно писать только при выключенном канале (CC1E = ‘0’ в TIMx_CCER).

18.4.8. Регистр режима захвата/сравнения 2 (TIMx_CCMR2)

Смещение адреса: 0x1C

По сбросу: 0x0000

Каналы могут работать на ввод (захват) и вывод (сравнение), что определяется битами `CCxS`. У остальных битов для ввода и вывода значение отличается. `OCxx` обозначает функции вывода, `ICxx` функции ввода. При этом используются разные каскады.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4 CE	OC4M[2:0]			OC4 PE	OC4 FE	CC4S[1:0]		OC3 CE.	OC3M[2:0]			OC3 PE	OC3 FE	CC3S[1:0]	
IC4F[3:0]				IC4PSC[1:0]				IC3F[3:0]				IC3PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Сравнение выхода:
— Бит 15 **OC4CE:** Разрешение очистки сравнения 4.
— Биты 14:12 **OC4M[2:0]**: Режим сравнения выхода 4.
— Бит 11 **OC4PE**: Разрешение предзагрузки сравнения 4.
— Бит 10 **OC4FE**: Быстрое разрешение сравнения 4.
— Биты 9:8 **CC4S[1:0]**: Выбор Захвата/Сравнения 4.
Определяет направление канала (ввод/вывод) и используемый вход.
00: Канал CC4 выходной
01: Канал CC4 входной, IC4 на TI4
10: Канал CC4 входной, IC4 на TI3
11: Канал CC4 входной, IC4 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.
NB: Биты CC4S можно писать только при выключенном канале (CC4E = ‘0’ в TIMx_CCER).
— Бит 7 **OC3CE**: Разрешение очистки сравнения выхода 3.
— Биты 6:4 **OC3M[2:0]**: Режим сравнения выхода 3.
— Бит 3 **OC3PE**: Разрешение регистра предзагрузки.
— Бит 2 **OC3FE**: Разрешение быстрого сравнения.
— Биты 1:0 **CC1S**: Выбор Захвата/Сравнения 1.
Определяет направление канала (ввод/вывод) и используемый вход.
00: Канал CC3 выходной
01: Канал CC3 входной, IC3 на TI3

10: Канал CC3 входной, IC3 на TI4

11: Канал CC3 входной, IC3 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.

NB: Биты CC3S можно писать только при выключенном канале (CC3E = ‘0’ в TIMx_CCER).

Захват входа.

— Биты 15:12 **IC4F**: Фильтр захвата входа 4.

— Биты 11:10 **IC4PSC[1:0]** : Предделитель захвата входа 4.

— Биты 9:8 **CC4S[1:0]**: Выбор Захвата/Сравнения 4.

Определяет направление канала (ввод/вывод) и используемый вход.

00: Канал CC4 выходной

01: Канал CC4 входной, IC4 на TI4

10: Канал CC4 входной, IC4 на TI3

11: Канал CC4 входной, IC4 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.

NB: Биты CC4S можно писать только при выключенном канале (CC4E = ‘0’ в TIMx_CCER).

— Биты 7:4 **IC3F[3:0]**: Фильтр захвата входа 3.

Определяет частоту выборки и длину цифрового фильтра сигнала TI3. Цифровой фильтр это счётчик, выдающий сигнал на выход после N последовательных событий на входе:

0000: Без фильтра, выборка на частоте f_{DTS}

0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2

0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4

0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8

0100: $f_{SAMPLING}=f_{DTS}/2$, N=6

0101: $f_{SAMPLING}=f_{DTS}/2$, N=8

0110: $f_{SAMPLING}=f_{DTS}/4$, N=6

0111: $f_{SAMPLING}=f_{DTS}/4$, N=8

1000: $f_{SAMPLING}=f_{DTS}/8$, N=6

1001: $f_{SAMPLING}=f_{DTS}/8$, N=8

1010: $f_{SAMPLING}=f_{DTS}/16$, N=5

1011: $f_{SAMPLING}=f_{DTS}/16$, N=6

1100: $f_{SAMPLING}=f_{DTS}/16$, N=8

1101: $f_{SAMPLING}=f_{DTS}/32$, N=5

1110: $f_{SAMPLING}=f_{DTS}/32$, N=6

1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

— Биты 3:2 **IC3PSC**: Предделитель захвата входа 3

— Биты 1:0 **CC3S**: Выбор Захвата/Сравнения 3.

Определяет направление канала (ввод/вывод) и используемый вход.

00: Канал CC3 выходной

01: Канал CC3 входной, IC3 на TI3

10: Канал CC3 входной, IC3 на TI4

11: Канал CC3 входной, IC3 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.

NB: Биты CC3S можно писать только при выключенном канале (CC3E = ‘0’ в TIMx_CCER).

18.4.9. Регистр разрешения захвата/сравнения (TIMx_CCER)

Смещение адреса: 0x20

По сбросу: 0x0000

15 14	13	12	11 10	9	8	7 6	5	4	3 2	1	0
Reserved	CC4P	CC4E	Reserved	CC3P	CC3E	Reserved	CC2P	CC2E	Reserved	CC1P	CC1E
	rw	rw		rw	rw		rw	rw		rw	rw

— Биты 15:14 Резерв, не трогать.

— Бит 13 **CC4P**: Полярность выхода захвата/сравнения 4

— Бит 12 **CC4E**: Разрешение выхода захвата/сравнения 4

— Биты 11:10 Резерв, не трогать.

— Бит 9 **CC3P**: Полярность выхода захвата/сравнения 3

— Бит 8 **CC3E**:, Разрешение выхода захвата/сравнения 3

— Биты 7:6 Резерв, не трогать.

— Бит 5 **CC2P**: Полярность выхода захвата/сравнения 2

— Бит 4 **CC2E**: Разрешение выхода захвата/сравнения 2

— Биты 3:2 Резерв, не трогать.

— Бит 1 **CC1P**: Полярность выхода захвата/сравнения 1

1: Выдаёт событие захвата/сравнения:

В режиме вывода CC1:

0: OC1 активен высоким.

1: OC1 активен низким.

В режиме ввода CC1:

Выбирает для запуска или захвата прямой или инверсный IC1.

0: прямой: захват по переднему фронту IC1. Внешний запуск по IC1 не инвертируется.

1: инверсный: захват по заднему фронту IC1. Внешний запуск по IC1 инвертируется.

— Бит 0 **CC1E**: Разрешение выхода захвата/сравнения 1

В режиме вывода CC1:

0: Выкл. - OC1 не активен.

1: Вкл. - OC1 выводится на соответствующую ножку.

В режиме ввода CC1:

0: Захват выключен.

1: Захват включен.

Таблица 99. Управляющие биты стандартных OCx каналов

Бит CCxE	Состояние OCx
0	Выключен (OCx=0, OCx_EN=0)
1	OCx=OCxREF + Полярность, OCx_EN=1

NB: Состояние внешних ножек I/O, подключённых к стандартным OCx, зависит от состояния OCx и регистров GPIO и AFIO.

18.4.10.Счётчик TIM2 - TIM5 (TIMx_CNT)

Смещение адреса: 0x24

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:0 **CNT[15:0]**: Значение счётчика.

18.4.11.Предделитель TIM2 - TIM5 (TIMx_PSC)

Смещение адреса: 0x28

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:0 **PSC[15:0]**: Значение предделителя

Частота счёта (CK_CNT) равна f_{CK_PSC} / (PSC[15:0] + 1).

Значение PSC пишется в рабочий регистр предделителя по каждому событию обновления (включая очистку счётчика битом UG регистра TIMx_EGR или запуском в режиме Сброса).

18.4.12.Регистр предзагрузки TIM2 - TIM5 (TIMx_ARR)

Смещение адреса: 0x2C

По сбросу: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:0 **ARR[15:0]**: Значение перезагрузки.

Если значение нулевое, то счётчик блокируется.

18.4.13.Регистр 1 захвата/сравнения TIM2 - TIM5 (TIMx_CCR1)

Смещение адреса: 0x34

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

— Биты 15:0 **CCR1[15:0]**: Значение захвата/сравнения

В режиме вывода CC1:

Значение предзагрузки CCR1 надо писать в рабочий регистр захвата/сравнения. Если функция предзагрузки не включена (бит OC1PE в регистре TIMx_CCMR1), то он пишется постоянно, иначе только по событию обновления.

Результат сравнения рабочего регистра и TIMx_CNT подаётся на выход OC1.

В режиме ввода CC1:

CCR1 содержит значение счётчика по последнему событию захвата 1 (IC1). Здесь TIMx_CCR1 только читается.

18.4.14.Регистр 2 захвата/сравнения TIM2 - TIM5 (TIMx_CCR2)

Смещение адреса: 0x38

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

— Биты 15:0 **CCR2[15:0]**: Значение захвата/сравнения

В режиме вывода CC2:

Значение предзагрузки CCR2 надо писать в рабочий регистр захвата/сравнения. Если функция предзагрузки не включена (бит OC2PE в регистре TIMx_CCMR2), то он пишется постоянно, иначе только по событию обновления.

Результат сравнения рабочего регистра и TIMx_CNT подаётся на выход OC2.

В режиме ввода CC2:

CCR2 содержит значение счётчика по последнему событию захвата 2 (IC2). Здесь TIMx_CCR2 только читается.

18.4.15.Регистр 3 захвата/сравнения TIM2 - TIM5 (TIMx_CCR3)

Смещение адреса: 0x3C

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

— Биты 15:0 **CCR3[15:0]**: Значение захвата/сравнения

В режиме вывода CC3:

Значение предзагрузки CCR3 надо писать в рабочий регистр захвата/сравнения. Если функция предзагрузки не включена (бит OC3PE в регистре TIMx_CCMR3), то он пишется постоянно, иначе только по событию обновления.

Результат сравнения рабочего регистра и TIMx_CNT подаётся на выход OC3.

В режиме ввода CC3:

CCR3 содержит значение счётчика по последнему событию захвата 3 (IC3). Здесь TIMx_CCR3 только читается.

18.4.16.Регистр 4 захвата/сравнения TIM2 - TIM5 (TIMx_CCR4)

Смещение адреса: 0x40

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

— Биты 15:0 **CCR4[15:0]**: Значение захвата/сравнения

В режиме вывода CC4:

Значение предзагрузки CCR4 надо писать в рабочий регистр захвата/сравнения. Если функция предзагрузки не включена (бит OC4PE в регистре TIMx_CCMR4), то он пишется постоянно, иначе только по событию обновления.

Результат сравнения рабочего регистра и TIMx_CNT подаётся на выход OC4.

В режиме ввода CC4:

CCR4 содержит значение счётчика по последнему событию захвата 4 (IC4). Здесь TIMx_CCR4 только читается.

18.4.17.Регистр управления DMA (TIMx_DCR)

Смещение адреса: 0x48

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			DBL[4:0]					Reserved			DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

— Биты 15:13 Резерв, не трогать.

— Биты 12:8 **DBL[4:0]**: Длина пакета DMA

Это число передач DMA (таймер обнаруживает передачу пакета после выполнения чтения/записи по адресу регистра TIMx_DMAR).

Адрес TIMx_DMAR:

00000: 1 передача

00001: 2 передачи

00010: 3 передачи

…

10001: 18 передач

— Биты 7:5 Резерв, не трогать.

— Биты 4:0 **DBA[4:0]**: Базовый адрес DMA

Это базовый адрес передач DMA (при доступе через адрес TIMx_DMAR). DBA определяется как смещение от адреса регистра TIMx_CR1.

Пример:

00000: TIMx_CR1,

00001: TIMx_CR2,

00010: TIMx_SMCR,

...

Пример: Есть передача: DBL = 7 передач и DBA = TIMx_CR1. Тогда доступ будет к 7 регистрам, начиная с адреса TIMx_CR1.

18.4.18.Регистр адреса полного доступа DMA (TIMx_DMAR)

Смещение адреса: 0x4C

По сбросу: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAB[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:0 **DMAB[31:0]**: Регистр пакетного доступа DMA

Чтение и запись в регистр DMAR обращаются к регистру по адресу (адрес TIMx_CR1) + (DBA + индекс DMA) x 4, где (адрес TIMx_CR1) это адрес управляющего регистра 1, DBA это базовый адрес DMA в регистре TIMx_DCR, индекс DMA автоматически изменяется при DMA передачах от 0 до DBL из регистра TIMx_DCR.

Пример пакетного DMA

Здесь пакетом DMA полусловами пишутся регистры CCRx (x = 2, 3, 4).

Нужны следующие шаги:

1. Ставим нужный канал DMA:
   - Адрес периферии DMA пишем в регистр DMAR
   - Адрес памяти это адрес данных в RAM для записи в регистры CCRx.
   - Число передач = 3.
   - Выключаем кольцевой режим.
2. Пишем поля DBA и DBL регистра DCR: DBL = 3 передачи, DBA = 0xE.
3. Разрешаем запрос DMA обновления TIMx (ставим бит UDE в регистре DIER).
4. Включаем TIMx
5. Включаем канал DMA

Здесь каждый регистр CCRx обновляется единожды. Если регистры CCRx нужно обновить дважды, то число передач должно быть 6. Буфер в RAM содержит data1, data2, data3, data4, data5 и data6. В регистры CCRx они передаются так: по первому запросу DMA, data1 в CCR2, data2 в CCR3, data3 в CCR4 и по второму запросу обновления DMA, data4 в CCR2, data5 в CCR3 в data6 в CCR4.

18.4.19.Карта регистров TIM2 - TIM5

Таблица 100.

0x00	TIMx_CR1	Reserved																	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																		0	0	0	0	0	0	0	0	0	0
0x04	TIMx_CR2	Reserved																			TI1S	MMS [2:0]			CCDS	Reserved							
	Reset value																				0	0	0	0	0								
0x08	TIMx_SMCR	Reserved															ETP	ECE	ETPS [1:0]		ETF[3:0]				MSM	TS[2:0]			Reserved	SMS[2:0]			
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	
0x0C	TIMx_DIER	Reserved																TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Reserved	TIE	Reserved	CC4IE	CC3IE	CC2IE	CC1IE	UIE	
	Reset value																	0	0	0	0	0	0	0		0		0	0	0	0	0	
0x10	TIMx_SR	Reserved																			CC4OF	CC3OF	CC2OF	CC1OF	Reserved	TIF	Reserved	CC4IF	CC3IF	CC2IF	CC1IF	UIF	
	Reset value																				0	0	0	0		0		0	0	0	0	0	
0x14	TIMx_EGR	Reserved																								TG	Reserved	CC4G	CC3G	CC2G	CC1G	UG	
	Reset value																									0		0	0	0	0	0	
0x18	TIMx_CCMR1 Output Compare mode	Reserved															OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]		
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	TIMx_CCMR1 Input Capture mode	Reserved															IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]		
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x1C	TIMx_CCMR2 Output Compare mode	Reserved															O24CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]		
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	TIMx_CCMR2 Input Capture mode	Reserved															IC4F[3:0]				IC4 PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3S [1:0]		
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x20	TIMx_CCER	Reserved															Reserved				CC4P	CC4E	Reserved		CC3P	CC3E	Reserved		CC2P	CC2E	Reserved		CC1P	CC1E
	Reset value																				0	0			0	0			0	0			0	0

0x24	TIMx_CNT	Reserved	CNT[15:0]													
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIMx_PSC	Reserved	PSC[15:0]													
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TIMx_ARR	Reserved	ARR[15:0]													
	Reset value		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	Reserved															
0x34	TIMx_CCR1	Reserved	CCR1[15:0]													
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	TIMx_CCR2	Reserved	CCR2[15:0]													
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	TIMx_CCR3	Reserved	CCR3[15:0]													
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	TIMx_CCR4	Reserved	CCR4[15:0]													
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	Reserved															
0x48	TIMx_DCR	Reserved									DBL[4:0]					Reserved	DBA[4:0]					
	Reset value										0	0	0	0	0		0	0	0	0	0	
0x4C	TIMx_DMAR	Reserved	DMAB[15:0]													
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

19. Таймеры общего назначения (TIM9 - TIM14)

19.1. Введение в TIM9 - TIM14

Это 16-бит само-перегружаемые таймеры с программируемым предделителем.

Они могут использоваться для измерения длительности входных импульсов (захват входа), генерации сигналов (сравнение выхода и ШИМ).

Длина импульсов и сигналов может модулироваться от нескольких микросекунд до миллисекунд с помощью предделителей таймера и тактов RCC.

Улучшенные (TIM1 and TIM8) и обычные (TIMx) таймеры полностью независимы и не имеют общих ресурсов. Могут синхронизироваться между собой. См. *Секцию 15.3.15*.

19.2. Основные свойства TIM9 - TIM14

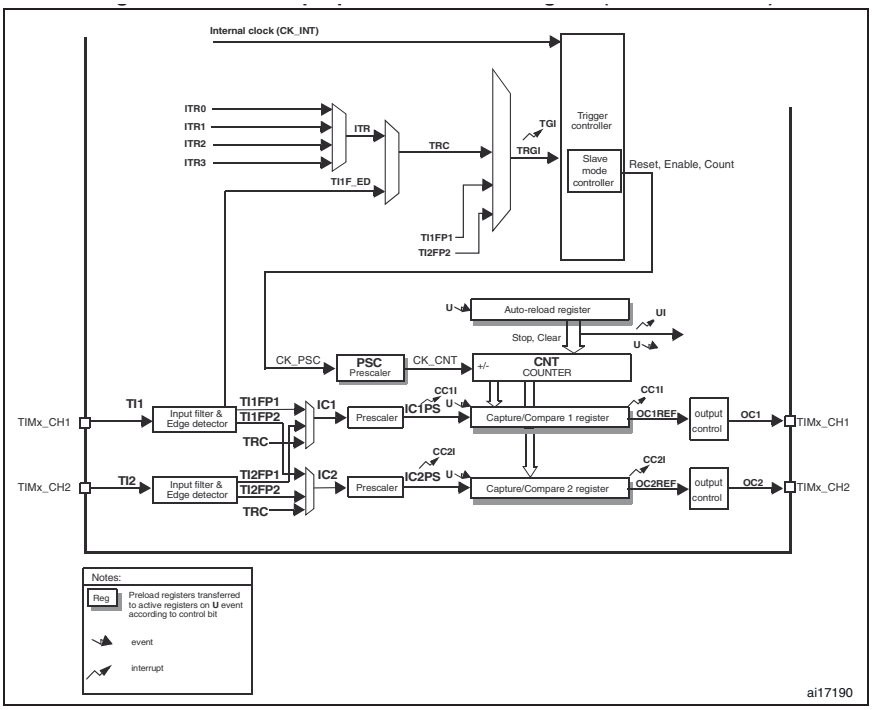
19.2.1. Основные свойства TIM9/TIM12

Это:

- 16-бит прямой счёт с само-перезагрузкой.
- 16-бит программируемый предделитель (можно “налету”) тактов на число от 1 до 65536.
- До 2 независимых каналов для:
  - Захвата входа
  - Сравнение выхода
  - ШИМ генерация (Фронт и Центр)
  - Одиночный импульс
- Схема синхронизации внешнего управления и объединения таймеров.

- Генерация прерываний по следующим событиям:
  - Обновление: переполнение, инициализация счётчика (программно или запуском)
  - Запуск (старт, стоп, инициализация или счёт по сигналу)
  - Захват входа
  - Сравнение выхода

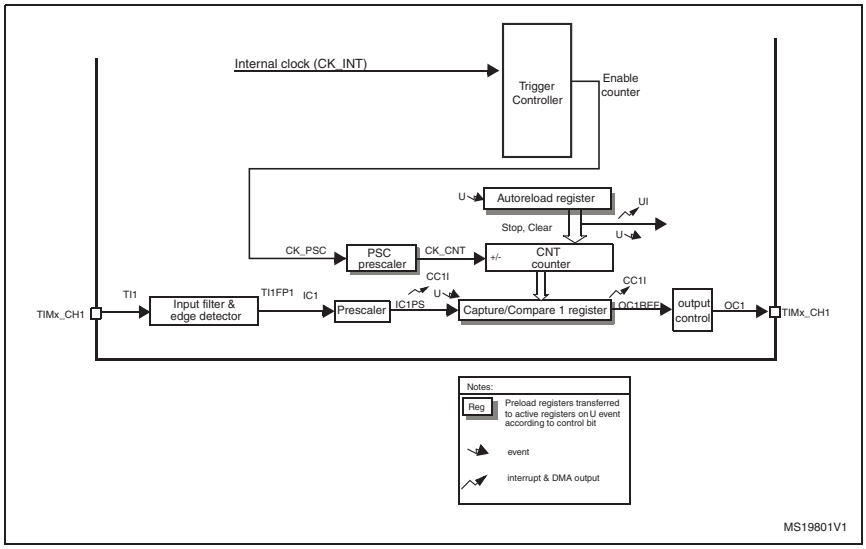
Рис. 180. Блок-схема TIM9/TIM12.



19.2.2. Основные свойства TIM10/TIM11 и TIM13/TIM14

- 16-бит прямой счёт с само-перезагрузкой.
- 16-бит программируемый предделитель (можно “налету”) тактов на число от 1 до 65536.
- Независимый канал для:
  - Захвата входа
  - Сравнение выхода
  - ШИМ генерация (Фронт и Центр)
  - Одиночный импульс
- Генерация прерываний по следующим событиям:
  - Обновление: переполнение, инициализация счётчика (программно)
  - Захват входа
  - Сравнение выхода

Рис. 181. Блок-схема TIM10/11/13/14.



19.3. Функциональное описание таймеров TIM9 - TIM14

19.3.1. Узел счёта

Это прямой 16-бит счётчик с регистром само-перезагрузки и предделителем.

Счётчик, регистр перезагрузки и предделитель доступны программно для чтения и записи даже во время счёта. Включает:

- Регистр счётчика (TIMx_CNT)
- Регистр предделителя (TIMx_PSC)
- Регистр перезагрузки (TIMx_ARR)

Регистр перезагрузки предзагружаемый (есть видимый и скрытый регистры). Содержимое видимого регистра пишется в скрытый постоянно или по каждому событию обновления (UEV), в зависимости от бита разрешения (ARPE) в регистре TIMx_CR1. Оно посылается программно или при переполнении счётчика при снятом бите UDIS в регистре TIMx_CR1.

Счётчик тактируется выходом предделителя CK_CNT. Разрешается битом CEN в регистре TIMx_CR1.

Счёт начинается через 1 такт после установки бита CEN в регистре TIMx_CR1.

Описание предделителя

Это 16-бит счётчик делителя входной частоты (от 1 до 65536) с видимым регистром TIMx_PSC. Его можно менять налету. Новое значение счётчика пишется по следующему событию обновления.

Рис. 182. Временная диаграмма с изменением предделителя с 1 на 2.

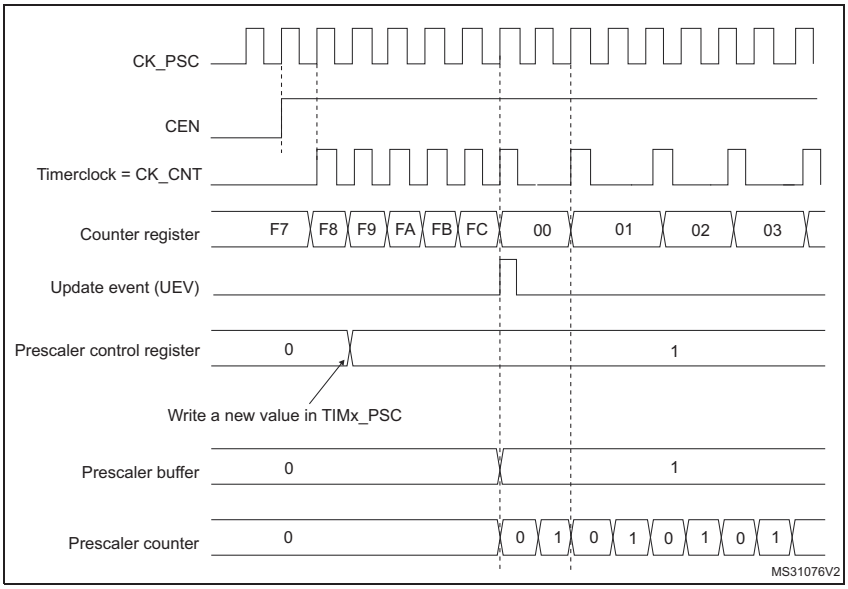
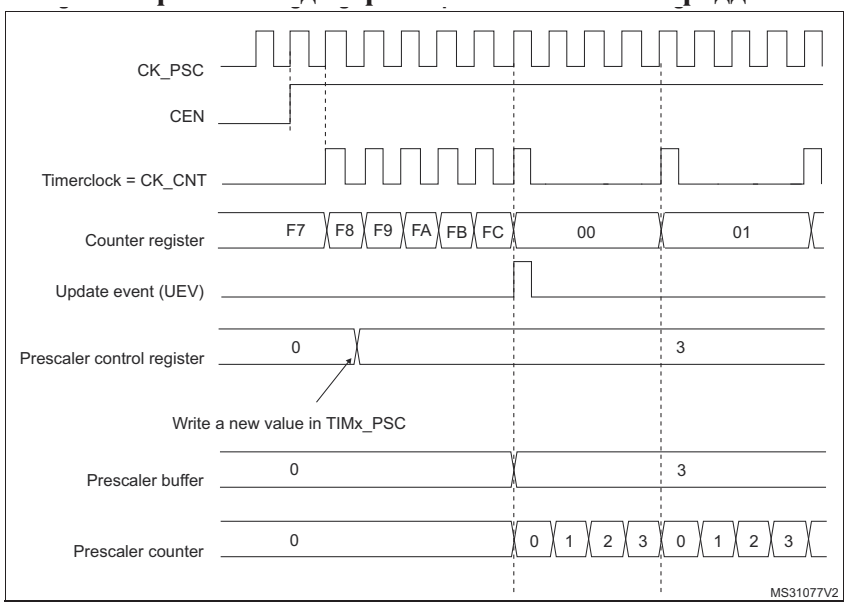


Рис. 183. Временная диаграмма с изменением предделителя с 1 на 4.



19.3.2. Режимы счёта

Прямой счёт

Счёт идёт от 0 до значения перезагрузки (регистр TIMx_ARR), сбрасывается в 0 и выдаёт событие переполнения счётчика.

Событие обновления UEV выдаётся при каждом переполнении счётчика или установкой бита UG в регистре TIMx_EGR (программно или контроллером режима ведомого).

Событие UEV выключается установкой бита UDIS в регистре TIMx_CR1. Этим предупреждается запись скрытого регистра во время изменения регистра предзагрузки. Также событие UEV не появляется при сброшенном бите UDIS 0. Однако, счётчик продолжает считать с 0, равно как и предделитель (не изменив своего значения). Кроме того, если стоит бит URS в регистре TIMx_CR1, то установка бита UG выдаёт событие UEV без установки флага UIF (без прерывания). Этим предупреждается одновременная выдача прерываний обновления и захвата при очистке счётчика по событию захвата.

По событию обновления обновляются все регистры и ставится флаг обновления (бит UIF в регистре TIMx_SR). Это зависит от бита URS:

- В скрытый регистр перезагрузки пишется содержимое TIMx_ARR,
- Буфер предделителя перегружается из регистра TIMx_PSC.

Примеры ниже используют TIMx_ARR=0x36.

Рис. 184. Временная диаграмма с делителем 1.

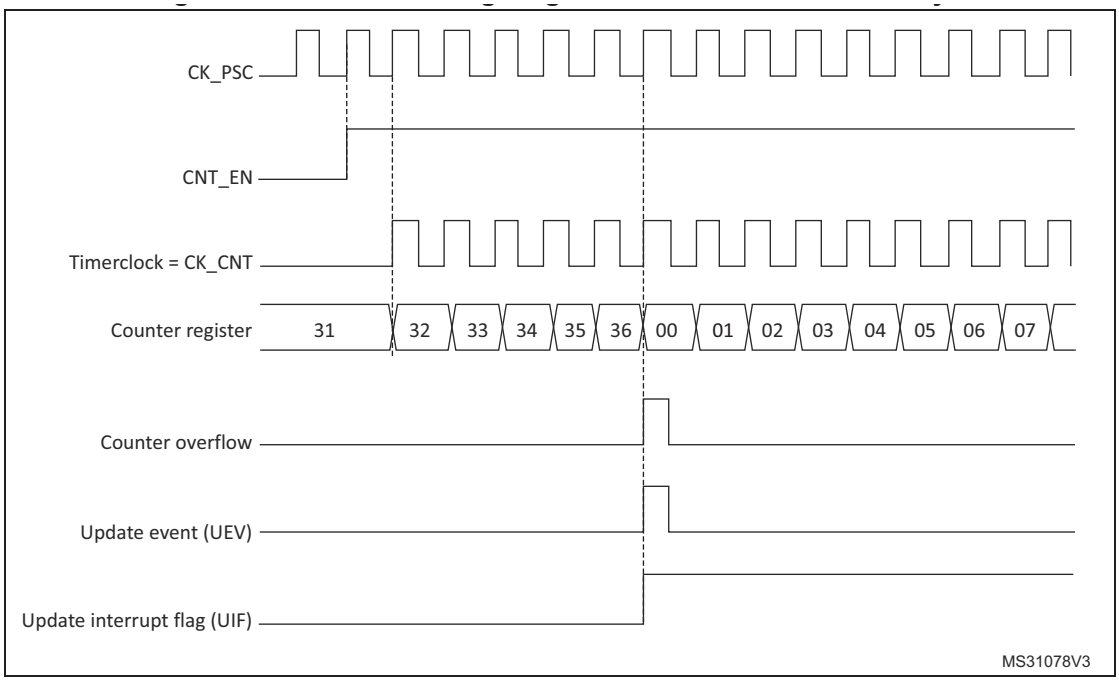


Рис. 185. Временная диаграмма с делителем 2.

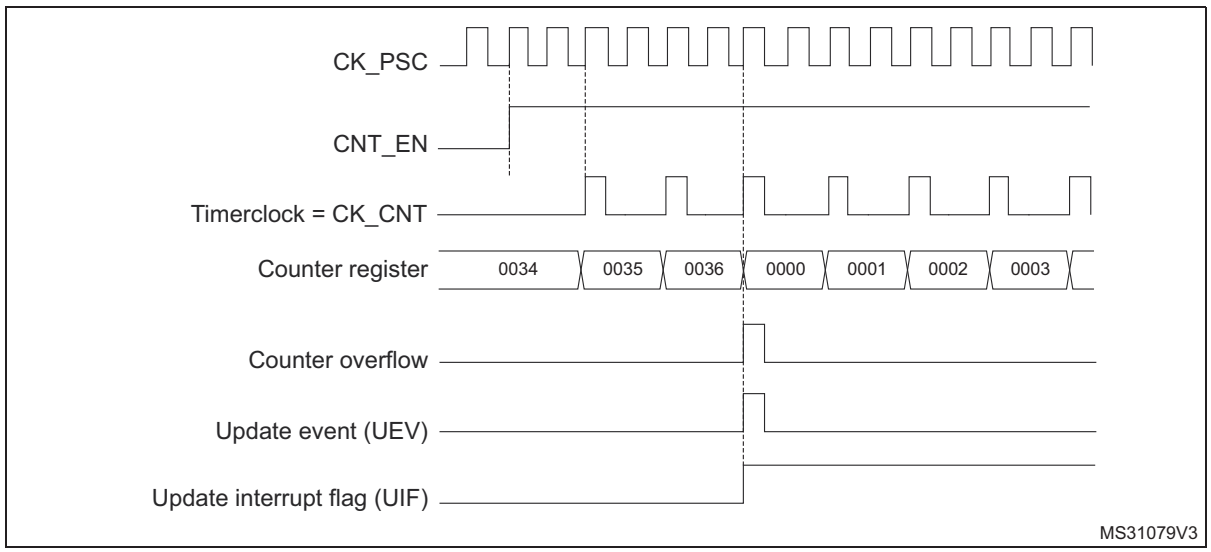


Рис. 186. Временная диаграмма с делителем 4.

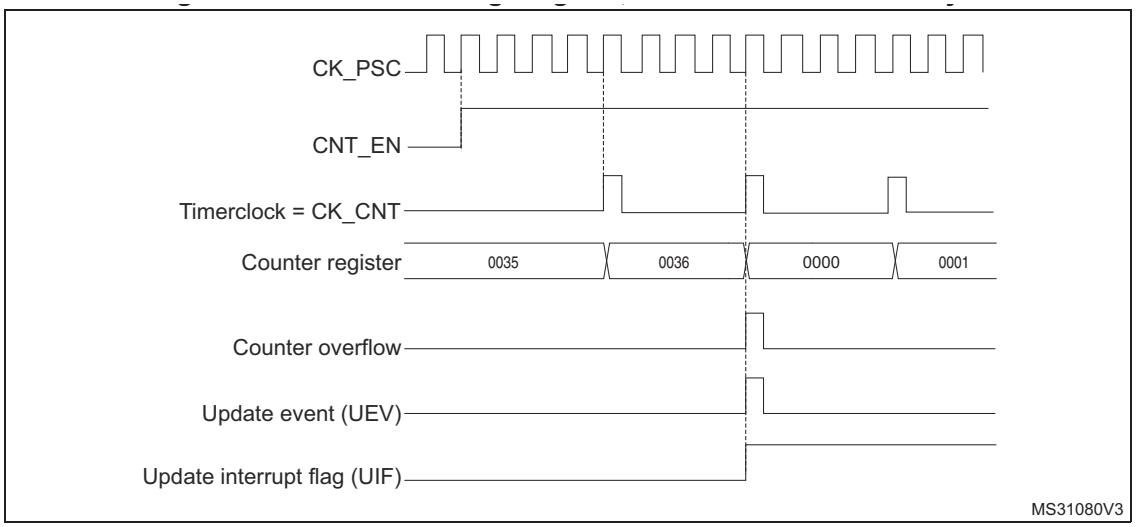


Рис. 187. Временная диаграмма с делителем N.

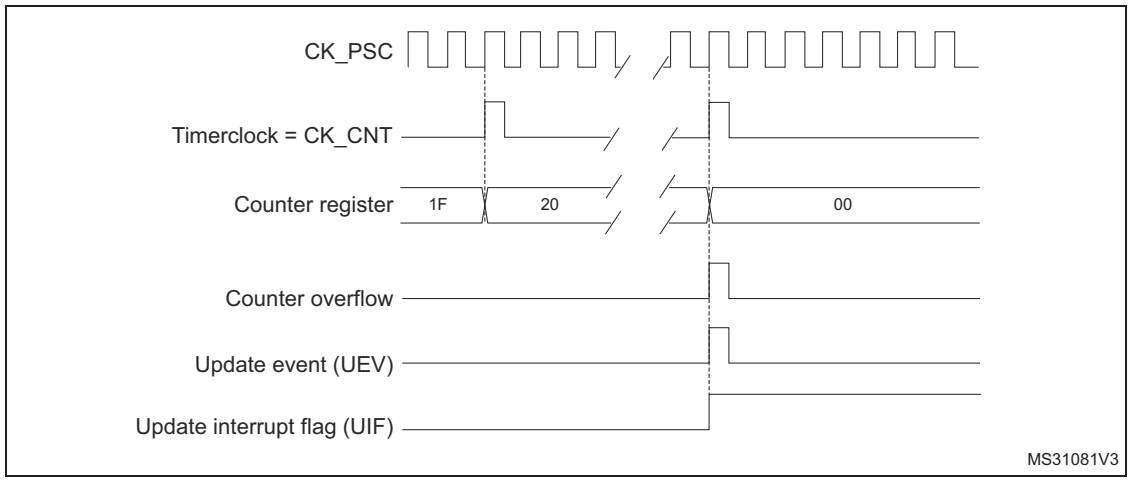


Рис. 188. Временная диаграмма события при ARPE=0 (TIMx_ARR не предзагружен).

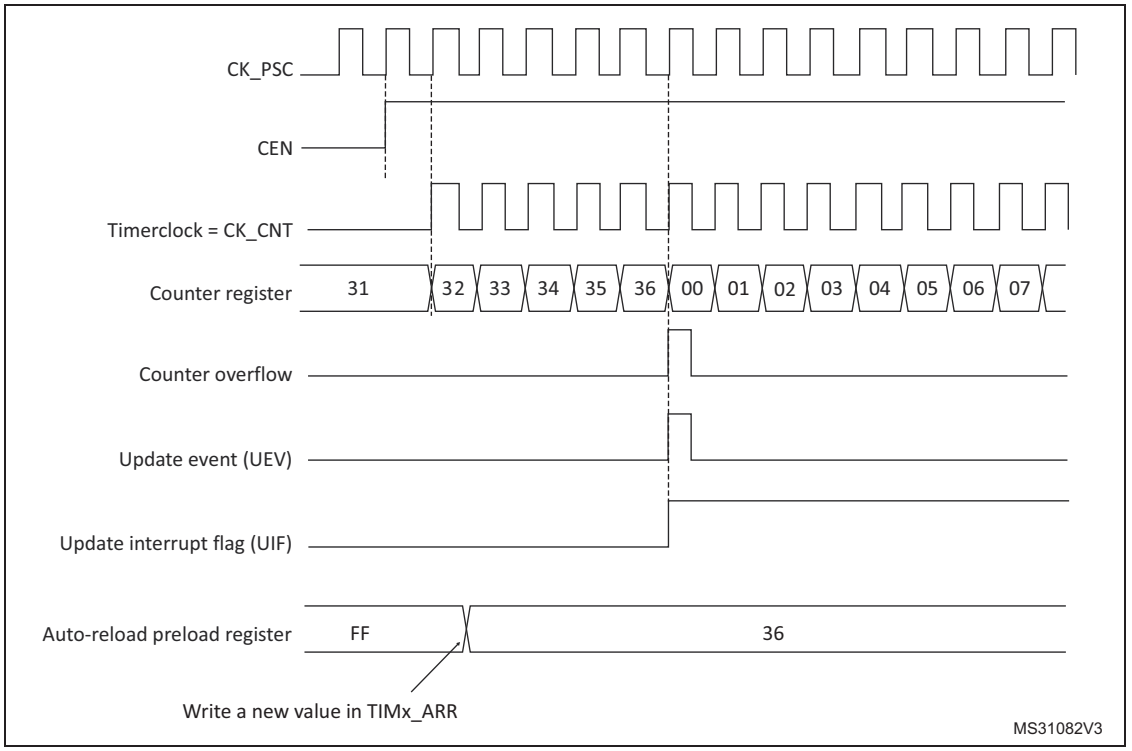
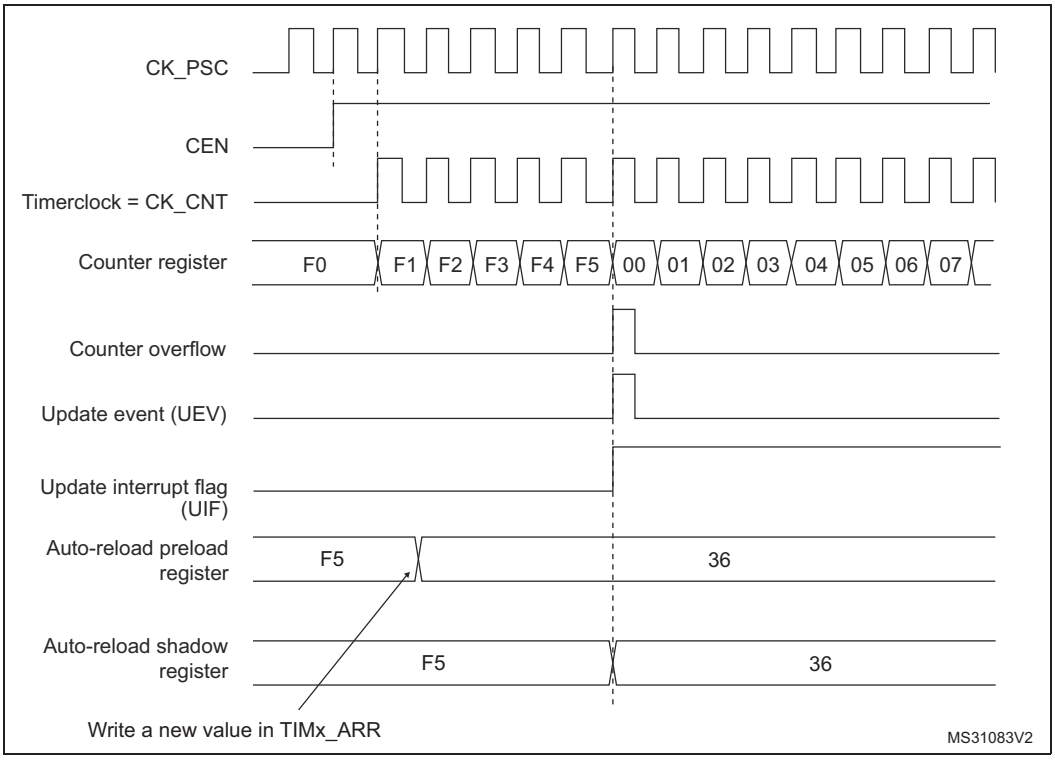


Рис. 189. Временная диаграмма события при ARPE=1 (TIMx_ARR предзагружен).



19.3.3. Выбор тактов

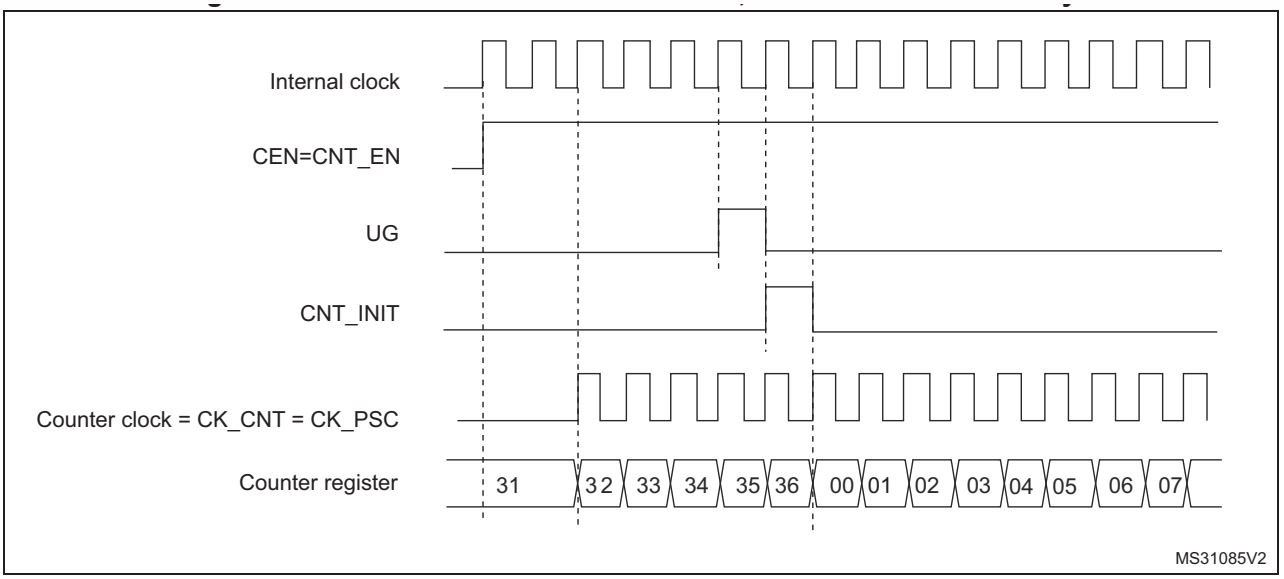
Счётчик может тактироваться из следующих источников:

- Внутренний (CK_INT)
- Внешний режим 1: внешняя ножка TIx
- Входы внутреннего сигнала (ITRx): использование одного таймера предделителем для другого.

Внутренний источник (CK_INT)

Если в ведомом режиме контроллер выключен (SMS=000), то всем управляют только биты CEN, DIR (в регистре TIMx_CR1) и UG (в регистре TIMx_EGR). Они изменяются только программно (кроме UG, снимающегося аппаратно). Сразу после установки бита CEN предделитель начинает получать внутренние такты CK_INT.

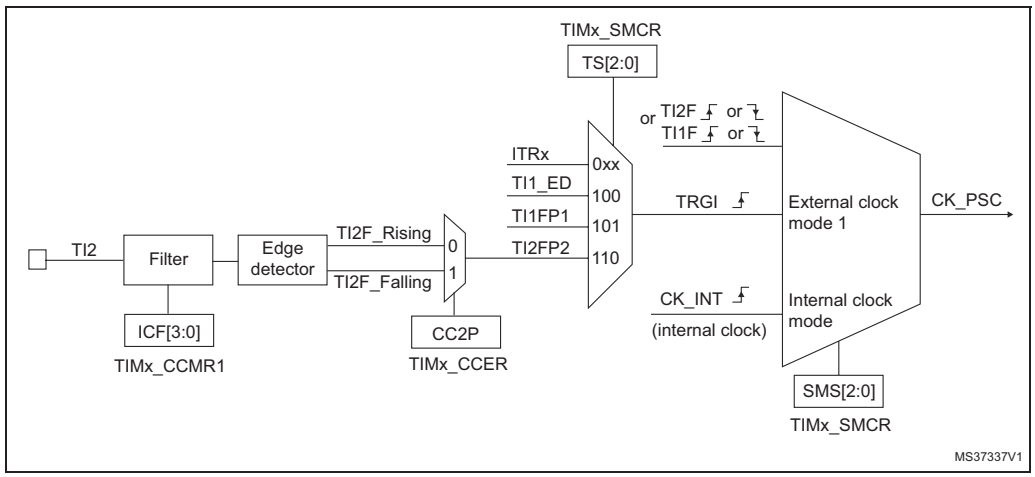
Рис. 190. Схема управления в нормальном режиме прямого счёта без предделителя.



Внешний источник режим 1 (TIM9 и TIM12).

Включается при установке SMS=111 в регистре TIMx_SMCR. Счётчик работает от переднего или заднего фронта выбранного входа.

Рис. 191. Внешний источник TI2.



Например, процедура включения режима прямого счёта по переднему фронту входа `TI2`:

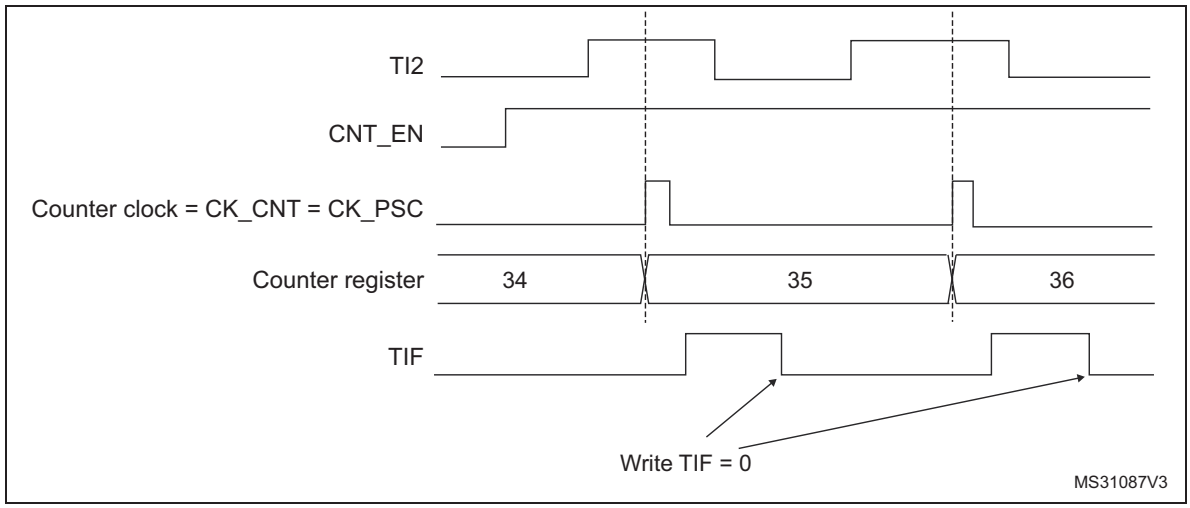
1. Включить определение переднего фронта `TI2` канала 2 записью `CC2S` = '`01`' в регистре `TIMx_CCMR1`.
2. Установить длительность входного фильтра битами `IC2F[3:0]` в регистре `TIMx_CCMR1` (если фильтр не нужен, то остаётся `IC2F`=`0000`).
3. Выбрать полярность переднего фронта записью `CC2P`=`CC2NP`=`0` в регистре `TIMx_CCER`.
4. Включить внешний режим 1 тактирования записью `SMS`=`111` в регистре `TIMx_SMCR`.
5. Выбрать `TI2` как источник сигнала записью `TS`=`110` в регистре `TIMx_SMCR`.
6. Включить счётчик записью `CEN`=1 в регистре `TIMx_CR1`.

Предделитель захвата для запуска не используется, ну и бог с ним.

По переднему фронту на `TI2` счётчик один раз тикает и ставит флаг `TIF`.

Задержка между передним фронтом на `TI2` и реальным тиком появляется из-за схемы ресинхронизации на входе `TI2`.

Рис. 192. Схема управления внешнего режима 1.

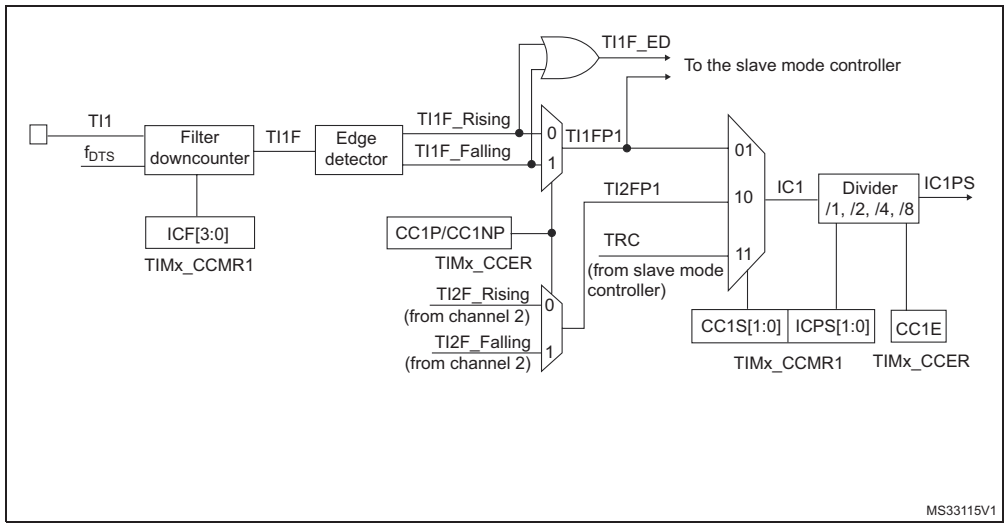


19.3.4. Каналы захвата/сравнения

Каждый канал захвата/сравнения построен вокруг парного регистра (включая невидимый), входного каскада захвата (с цифровым фильтром, мультиплексором и предделителем) и выходного каскада (с компаратором и управлением выходом).

Входной каскад считывает нужный вход `TIx` для выдачи фильтрованного сигнала `TIxF`. Далее, детектор фронта с выбором полярности выдаёт сигнал (`TIxFPx`), что может быть использован как вход запуска контроллером режима ведомого или как команда захвата. Предделитель стоит до регистра захвата (`ICxPS`).

Рис. 193. Входной каскад канала 1.



Выходной каскад выдаёт промежуточный сигнал, используемый как опорный: `OCxREF` (активный высокий). Полярность действует в конце цепочки.

Рис. 194. Основная схема канала 1.

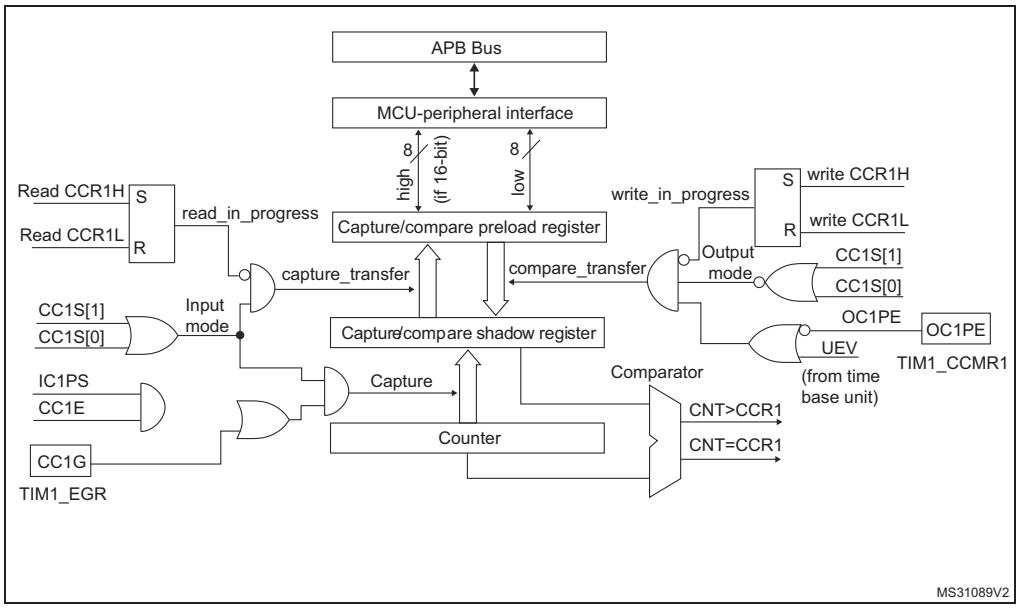
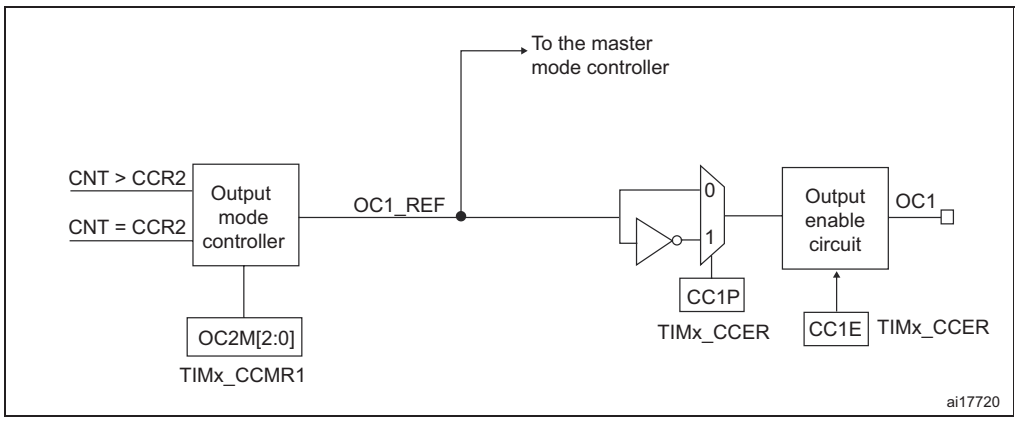


Рис. 195. Выходной каскад канала захвата/сравнения 1.



Блок захвата/сравнения состоит из двух регистров: предзагрузки и невидимого. Снаружи доступен только регистр предзагрузки и по чтению и по записи.

Захват выполняется внутренним регистром и результат пишется в регистр предзагрузки.

При сравнении регистр предзагрузки пишется в невидимый, где и сравнивается со счётчиком.

19.3.5. Режим захвата входа

Регистры `TIMx_CCRx` защёлкивают содержимое счётчика после передачи, обнаруженной соответствующим сигналом `ICx`. При этом ставится флаг `CCXIF` в регистре `TIMx_SR` и ставится

запрос прерывания. Если захват происходит при стоящем флаге CCxIF, то ставится флаг перезахвата CCxOF в регистре TIMx_SR. Флаг CCxIF снимается записью в него нуля или чтением захваченных данных из регистра TIMx_CCRx. CCxOF снимается записью '0'.

Далее захватываем содержимое TIMx_CCR1 по переднему фронту TI1. Делаем так:

- Выбираем активный вход: TIMx_CCR1 подключаем в TI1 и пишем "01" в биты CC1S регистра TIMx_CCMR1. Биты CC1S отличаются от 00, т. е. канал на вводе и регистр TIMx_CCR1 доступен только по чтению.
- Ставим длительность входного фильтра битами ICxF в регистре TIMx_CCMRx с учётом входного сигнала на TIx. Допустим, что при переключении входной сигнал нестабилен в течении пяти внутренних тактов. Значит длительность фильтра должна быть больше этих пяти тактов. Мы можем определить передачу на TI1 за 8 последовательных выборок нового уровня на частоте f_{DTS}. И мы пишем 0011 в биты IC1F регистра TIMx_CCMR1.
- Выбираем фронт сигнала на TI1 записью 0 в бит CC1P регистра TIMx_CCER (передний).
- Ставим предделитель. Здесь он нам не нужен (пишем 00 в биты IC1PS регистра TIMx_CCMR1).
- Разрешаем запись счётчика в регистр захвата установкой бита CC1E в TIMx_CCER.
- Если нужно разрешаем выдачу запросов прерывания битом CC1IE в TIMx_DIER.

Если захват входа произошёл, то:

- Регистр TIMx_CCR1 получает значение счётчика активной передачи.
- Ставится флаг прерывания CC1IF. Если он ещё стоял, то ставится и флаг CC1OF.
- В зависимости от бита CC1IE генерируется прерывание.

Чтобы обработать перезахват рекомендуется читать регистр данных до опроса флага перезахвата.

NB: Запросы прерывания IC можно выдавать программно установкой битов CCxG в регистре TIMx_EGR.

19.3.6. Режим ввода ШИМ (только TIM9/TIM12)

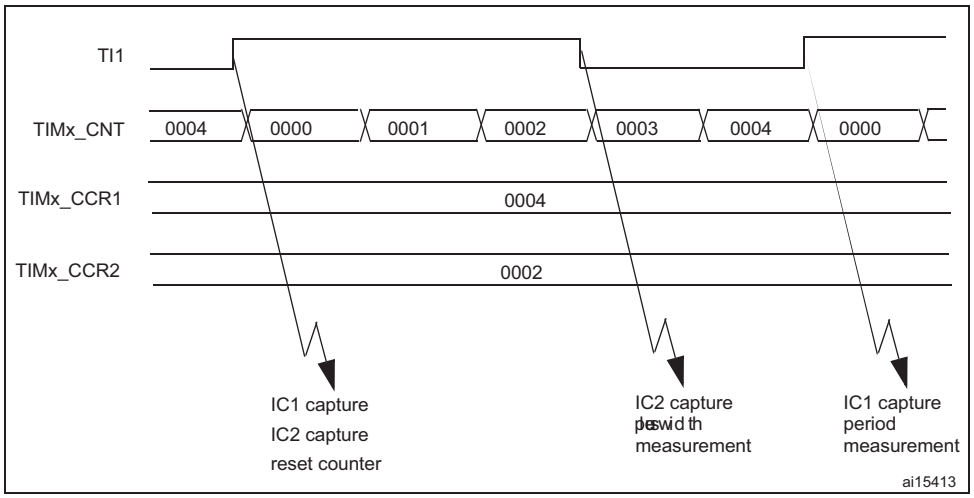
Это часть режима захвата входа. Процедура такая же, кроме:

- Два сигнала ICx направляются на на один вход TIx.
- Полярность фронтов этих входов ICx противоположная.
- Сигналом запуска берётся один из двух сигналов TIxFP, а контроллер ведомого режима выключается.

Например, можно измерить период (в регистре TIMx_CCR1) и длительность заполнения (в регистре TIMx_CCR2) ШИМ сигнала на TI1 следующей процедурой (зависит от частоты CK_INT и предделителя):

- Выбрать активный вход для TIMx_CCR1: Записать 01 в биты CC1S регистра TIMx_CCMR1 (TI1).
- Выбрать полярность для TI1FP1 (для захвата в TIMx_CCR1 и очистки счётчика): записать 0 в CC1P (передний фронт).
- Выбрать активный вход для TIMx_CCR2: записать 10 в биты CC2S регистра TIMx_CCMR1 (TI1).
- Выбрать полярность для TI1FP2 (для захвата в TIMx_CCR2): записать 1 в CC2P (задний фронт).
- Выбрать вход запуска: записать 101 в биты TS регистра TIMx_SMCR (TI1FP1).
- Поставить контроллер режима ведомого в сброс: записать 100 в биты SMS регистра TIMx_SMCR.
- Включить захват: поставить биты CC1E и CC2E в регистре TIMx_CCER.

Рис. 196. Времянка режима ввода ШИМ.



1. Режим ШИМ может использоваться только с сигналами TIMx_CH1/TIMx_CH2 так как к контроллеру режима ведомого подключены только TI1FP1 и TI2FP2.

19.3.7. Принудительный вывод

В режиме вывода (биты CCxS = 00 в TIMx_CCMRx), каждый выходной сигнал сравнения (OCxREF и затем OCx/OCxN) можно программно поставить в активное или неактивное состояние, независимо от результата сравнения выходного регистра сравнения и счётчика.

Активными выходные сигналы (OCxREF/OCx) ставятся записью 101 в биты OCxM нужного регистра TIMx_CCMRx. Активный сигнал OCxREF всегда высокий, а OCx противоположен биту полярности CCxP.

Пример: CCxP=0 (OCx активный высокий) => OCx ставится высоким.

Сигнал OCxREF снимается записью 100 в биты OCxM регистра TIMx_CCMRx.

Но сравнение внутреннего регистра TIMx_CCRx со счётчиком всё равно выполняется, биты ставятся, генерируются запросы прерывания и DMA. См. ниже.

19.3.8. Режим сравнения выхода

Используется для управления выходным сигналом и определения истечения периода времени.

При совпадении регистра захвата/сравнения со счётчиком эта схема:

- Ставит выходную ножку в заданное состояние (биты OCxM в регистре TIMx_CCMRx) и полярность (бит CCxP в регистре TIMx_CCER). Ножка может хранить состояние(OCxM=000), стать активной (OCxM=001), неактивной (OCxM=010) или переключиться (OCxM=011).
- Ставит флаг прерывания (бит CCxIF в регистре TIMx_SR).
- При установленной маске прерывания (бит CCxIE в регистре TIMx_DIER) выдаёт запрос.

В регистр TIMx_CCRx можно писать как через регистр предзагрузки, так и без него (см. бит OCxPE в регистре TIMx_CCMRx).

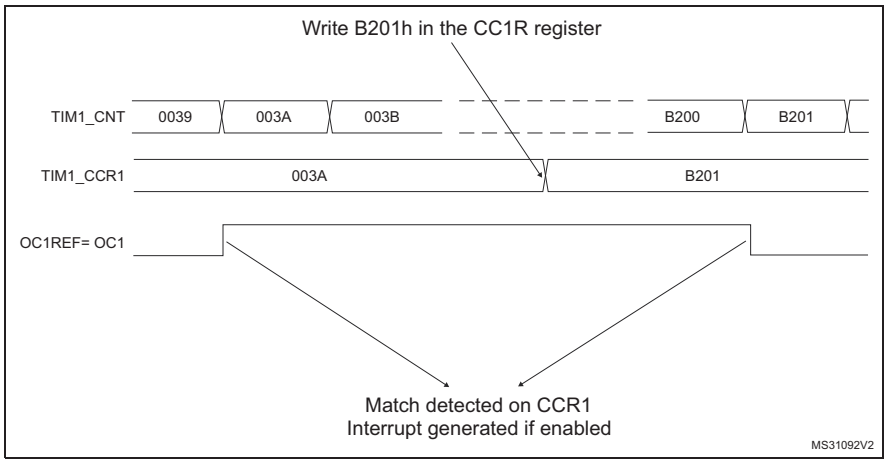
В режиме сравнения выхода бит события UEV на выходы OCxREF и OCx не влияет. Разрешение составляет один счёт счётчика. Этот режим можно использовать для выдачи одного импульса.

Процедура:

1. Выбрать такты счётчика (внутренние, внешние, предделитель).
2. Записать желаемое в регистры TIMx_ARR и TIMx_CCRx.
3. Если нужно прерывание, то установить бит CCxIE.
4. Выбрать режим выхода. Например:
   - Записать OCxM = 011 для переключения ножки OCx при совпадении CNT и CCRx
   - Записать OCxPE = 0 для отключения регистра предзагрузки
   - Записать CCxP = 0 ради высокого активного уровня
   - Записать CCxE = 1 для разрешения вывода
5. Включить счётчик установкой бита CEN в регистре TIMx_CR1.

При запрещённом регистре предзагрузки (OCxPE='0') регистр TIMx_CCRx можно обновлять в любое время, иначе он будет обновлён только по событию UEV.

Рис. 197. Режим сравнения выхода, переключение ножки OC1.



19.3.9. Режим ШИМ

Частота сигнала ШИМ определяется регистром TIMx_ARR, а коэффициент заполнения регистром TIMx_CCRx.

Режим ШИМ может включаться независимо по одному на каждый выход OCx записью ‘110’ (ШИМ1) или ‘111’ (ШИМ2) в биты OCxM регистра TIMx_CCMRx. Соответствующий регистр предзагрузки должно включать битом OCxPE в регистре TIMx_CCMRx и, соответсвенно, регистр авто-загрузки (в режимах прямого и реверсивного счёта) нужно включить битом ARPE в регистре TIMx_CR1.

Поскольку регистр предзагрузки пишется в теневой регистр только по событию обновления, то все регистры нужно заранее инициализировать установкой бита UG в регистре TIMx_EGR.

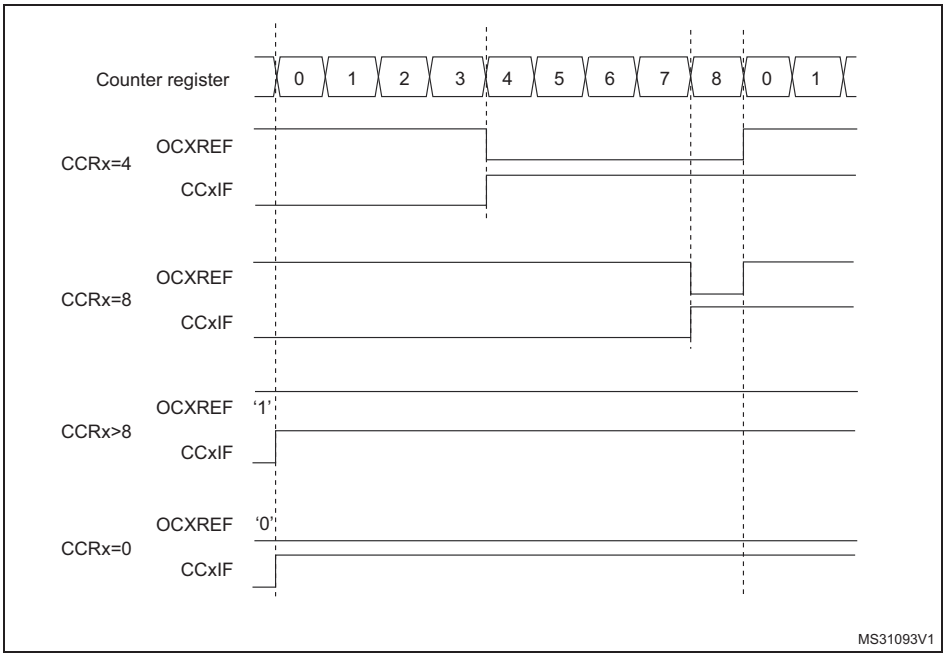
Полярность OCx (активный высокий или низкий) выбирается битом CCxP в регистре TIMx_CCER. Выходы OCx включаются битами CCxE, CCxNE, MOE, OSSI и OSSR в регистрах TIMx_CCER и TIMx_BDTR.

В режиме ШИМ (1 или 2), TIMx_CNT и TIMx_CCRx сравниваются на выполнение условия TIMx_CCRx ≤ TIMx_CNT (только при прямом счёте).

ШИМ по фронту

В примере ниже стоит режим ШИМ1. Опорный сигнал OCxREF остаётся высоким при TIMx_CNT < TIMx_CCRx. Если регистр TIMx_CCRx больше значения авто-загрузки (TIMx_ARR) то OCxREF равен ‘1’. При равенстве значений сравнения, OCxREF равен ‘0’. В примере ниже TIMx_ARR=8.

Рис. 198. ШИМ по фронту (TIMx_ARR=8).



19.3.10.Режим одного импульса (OPM)

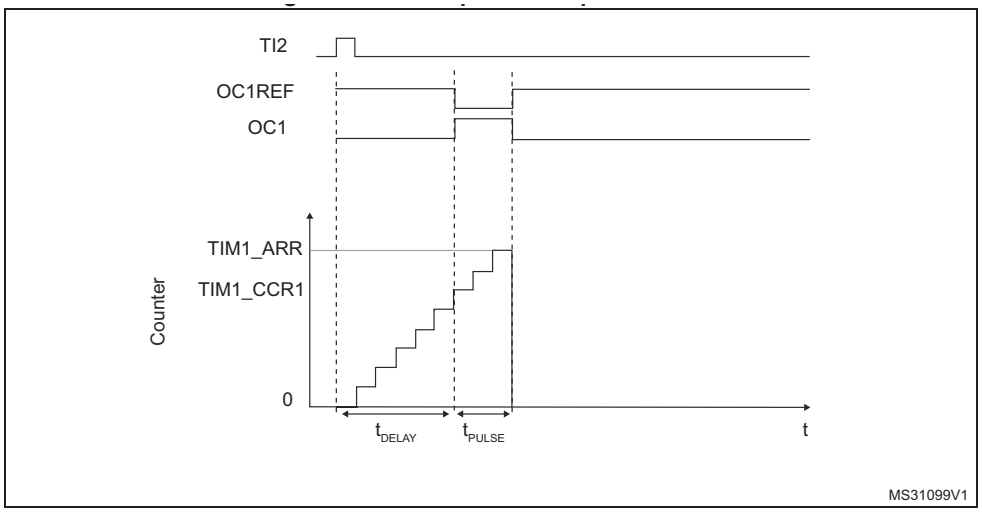
OPM это частный случай предыдущих режимов. Он позволяет счётчику запускаться по сигналу и генерировать импульс программируемой длины с программируемой задержкой.

Запускать счётчик можно из контроллера ведомого режима. Создавать сигнал можно в режиме сравнения выхода и ШИМ. Режим включается установкой бита OPM в регистре TIMx_CR1. Так счётчик автоматически останавливается по следующему событию UEV.

Импульс генерируется правильно только если значение сравнения отличается от начального значения счётчика. Перед стартом (таймер ждёт запуска) конфигурация должна быть:

CNT < CCRx ≤ ARR (в частности, 0 < CCRx)

Рис. 199. Режим одного импульса.



Например, выдадим положительный импульс на OC1 длиной t_{PULSE} после задержки t_{DELAY} по переднему фронту на ножке TI2.

Берём TI2FP2 как запуск 1:

- Поставим TI2FP2 на TI2 записью CC2S=’01’ в регистре TIMx_CCMR1.
- TI2FP2 ставим на передний фронт записью CC2P=’0’ в регистре TIMx_CCER.
- Выбираем TI2FP2 как запуск контроллером режима ведомого (TRGI) записью TS=’110’ в регистр TIMx_SMCR.
- TI2FP2 используем как старт записью SMS=‘110’ в TIMx_SMCR (режим запуска).

Сигнал OPM определяем записью регистров сравнения (учитываем частоту тактов и предделитель).

- Определяем t_{DELAY} записью в TIMx_CCR1.
- Ставим t_{PULSE} как разность значений перезагрузки и сравнения (TIMx_ARR - TIMx_CCR1+1).
- Сигнал создаётся переходом из ‘0’ в ‘1’ при сравнении и переходом из ‘1’ в ‘0’ при достижении счётчиком значения перезагрузки. Для этого, включаем режим ШИМ2 записью OC1M=111 в регистре TIMx_CCMR1. Регистры предзагрузки можно включить записью OC1PE=’1’ в регистре TIMx_CCMR1и ARPE в регистре TIMx_CR1, а можно не включать. В этом случае значение сравнения пишется в регистр TIMx_CCR1, значение перезагрузки в регистр TIMx_ARR, даём событие обновления битом UG и ждём внешнего запуска на TI2. В CC1P здесь пишется ‘0’.

Мы ждём только одного импульса (Однократный режим), так что для остановки счётчика по следующему событию обновления (счётчик переходит через значение перезагрузки и обнуляется) нужно поставить бит OPM регистра TIMx_CR1. Если бит OPM регистра TIMx_CR1 обнулён, то включается Повторяющийся режим.

Частный случай: быстрое включение OCx:

В этом режиме появление фронта на входе TIx ставит бит CEN, чем включает счётчик. Затем совпадение счётчика с значением сравнения переключает выходной сигнал. Но для этого нужны несколько тактов, ограничивая минимально возможное значение t_{DELAY}.

Если нужен сигнал с минимальной задержкой, то нужно ставить бит OCxFE в TIMx_CCMRx. Далее включаются OCxREF (и OCx) не глядя на сравнение. Его новый уровень равен уровню при сравнении. OCxFE работает только в режимах ШИМ1 и ШИМ2.

19.3.11.Синхронизация внешнего запуска TIM9/12

Есть три режима синхронизации TIMx: Сброс, Вентильный и Запуск.

Режим ведомого: Сброс

Счётчик и предделитель можно инициализировать по входу запуска. Кроме того, если бит URS в TIMx_CR1 сброшен, то генерируется событие UEV. Далее обновляются все предзагружаемые регистры (TIMx_ARR, TIMx_CCRx).

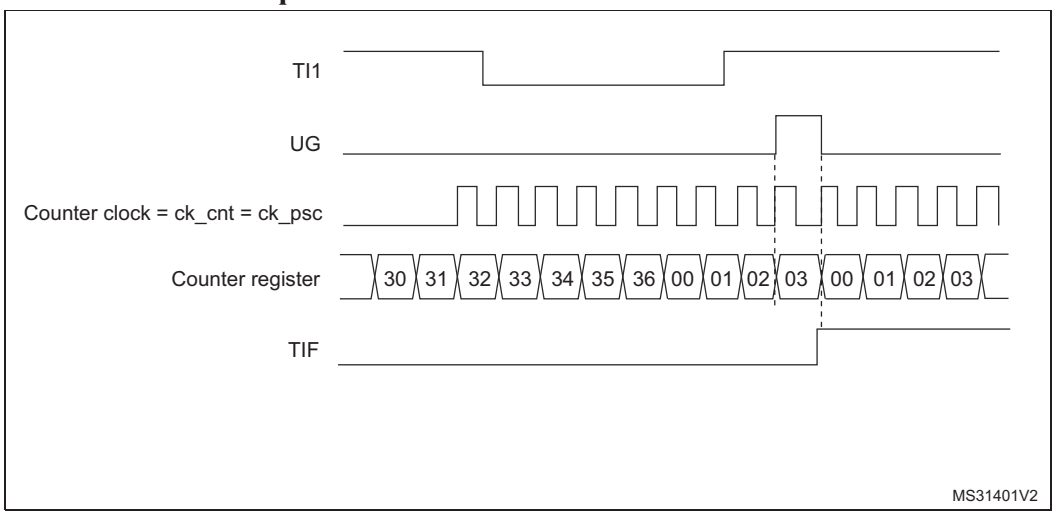
В этом примере счётчик прямого счёта сбрасывается по переднему фронту на входе TI1:

- Ставим канал 1 на передний фронт TI1. Входной фильтр отключён (IC1F=0000). Предделителя нет. Биты CC1S=01 в регистре TIMx_CCMR1 выбирают захват входа. Бит CC1P=0 в регистре TIMx_CCER определяет полярность и передний фронт.
- Ставим таймер в режим сброса записью SMS=100 в регистр TIMx_SMCR. Выбираем вход с TI1 записью TS=101 в регистре TIMx_SMCR.
- Запускаем счётчик записью CEN=1 в регистре TIMx_CR1.

Счётчик начинает считать внутренние такты вплоть переднего фронта TI1. Тогда счётчик сбрасывается с 0, ставится флаг запуска (бит TIF в регистре TIMx_SR) выдаются запросы прерывания и DMA (если разрешены битами TIE и TDE в регистре TIMx_DIER).

На рисунке ниже регистр перезагрузки TIMx_ARR=0x36. Задержка между фронтом TI1 и действительным сбросом появляется из-за схемы ресинхронизации на входе TI1.

Рис. 200. Режим Сброса.



Режим ведомого: Вентильный

Счётчик включается уровнем на выбранном входе.

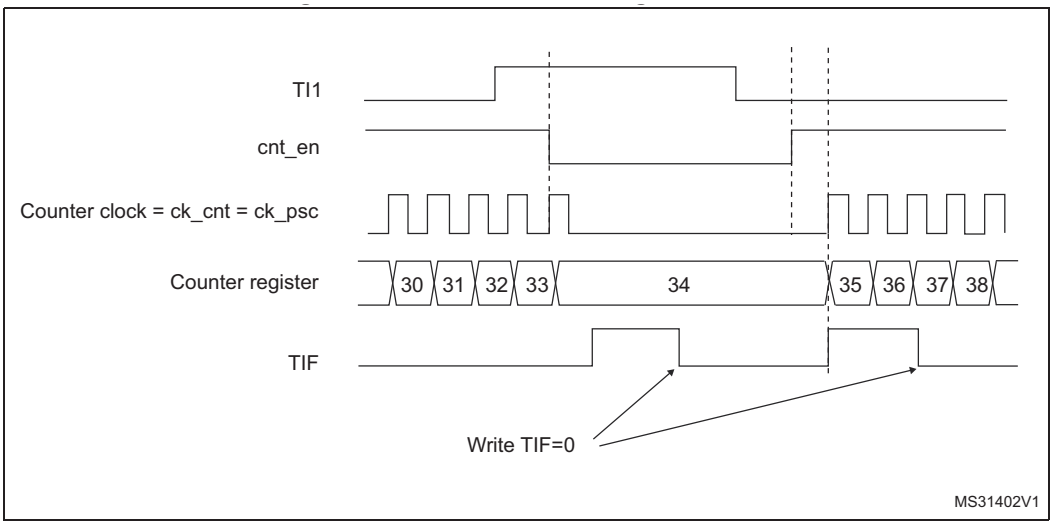
В этом примере счётчик прямого счёта работает только при низком уровне TI1:

- Ставим канал 1 на низкий уровень TI1. Входной фильтр отключён (IC1F=0000). Предделителя нет. Биты CC1S=01 в регистре TIMx_CCMR1 выбирают захват входа. Бит CC1P=1 в регистре TIMx_CCER определяет полярность и низкий уровень.
- Ставим таймер в вентильный режим записью SMS=101 в регистр TIMx_SMCR. Выбираем вход с TI1 записью TS=101 в регистре TIMx_SMCR.
- Запускаем счётчик записью CEN=1 в регистре TIMx_CR1. В этом режиме счётчик не работает при CEN=0, независимо от уровня на входе.

Счётчик начинает считать внутренние такты при низком уровне на TI1 и останавливается при высоком. Флаг запуска (бит TIF в регистре TIMx_SR) ставится при пуске и остановке таймера.

Задержка между фронтом TI1 и действительным сбросом появляется из-за схемы ресинхронизации на входе TI1.

Рис. 201. Вентильный режим.



Режим ведомого: Запуск

Счётчик включается событием на выбранном входе.

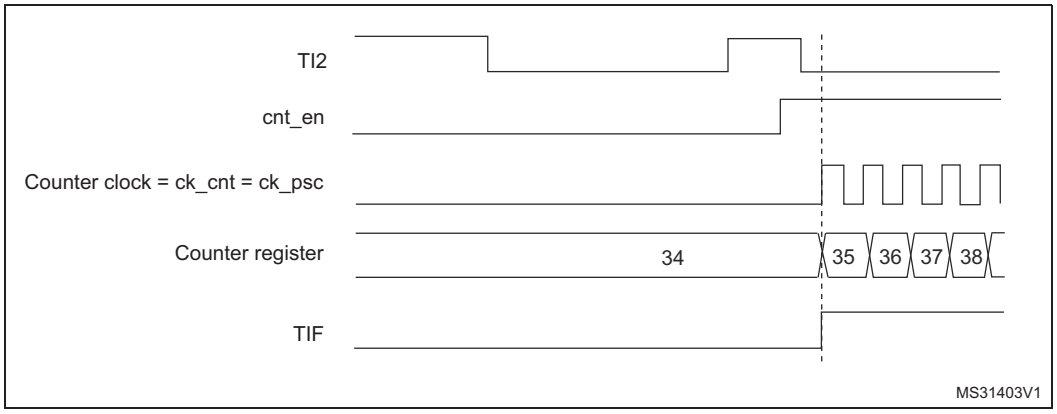
В этом примере счётчик прямого счёта работает по переднему фронту на TI2:

- Ставим канал 2 на передний фронт TI2. Входной фильтр отключён (IC1F=0000). Предделителя нет. Биты CC2S=01 в регистре TIMx_CCMR1 выбирают захват входа. Бит CC2P=1 в регистре TIMx_CCER определяет полярность и низкий уровень.
- Ставим таймер в режим запуска записью SMS=110 в регистр TIMx_SMCR. Выбираем вход с TI2 записью TS=110 в регистре TIMx_SMCR.

Счётчик начинает считать внутренние такты по переднему фронту на TI2 и ставит флаг запуска (бит TIF в регистре TIMx_SR).

Задержка между фронтом TI2 и действительным сбросом появляется из-за схемы ресинхронизации на входе TI2.

Рис. 202. Режим Запуска.



19.3.12.Синхронизация таймера TIM9/TIM12

Таймеры TIM связаны между собой для синхронизации или сцепления. См. *Секцию 15.3.15*.

NB: Такты ведомого таймера нужно включать до получения событий от ведущего и не должны меняться налету.

19.3.13.Режим отладки

В режиме отладки (ядро Cortex-M3 остановлено), счётчик TIMx либо работает, либо останавливается, в зависимости от бита DBG_TIMx_STOP в модуле DBG.

При стоящем счётчике (DBG_TIMx_STOP = 1 в регистре DBGMCU_APBx_FZ) выходы отключаются (как при сброшенном бите MOE). Выходы можно перевести в неактивное состояние (бит OSSI = 1), или возложить управление на контроллер GPIO (бит OSSI = 0) для перевода их в высокоимпедансное состояние (Hi-Z).

19.4. Регистры TIM9/12

32-бит регистры надо писать словами. Остальные регистры надо писать полусловами или словами. Читать можно байтами,полусловами или словами.

19.4.1. Регистр управления 1 TIM9/12 (TIMx_CR1)

Смещение адреса: 0x00

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKD[1:0]		ARPE	Reserved			OPM	URS	UDIS	CEN
						rw	rw	rw				rw	rw	rw	rw

— Биты 15:10 Резерв, не трогать.

— Биты 9:8 **CKD[1:0]:** Деление тактов.

Это отношение тактов таймера (CK_INT) и тактов задержки и выборки (t_{DTS}) в генераторах задержки и цифровых фильтрах (ETR, TIx),

00: $t_{DTS}=t_{CK\_INT}$

01: $t_{DTS}=2*t_{CK\_INT}$

10: $t_{DTS}=4*t_{CK\_INT}$

11: Резерв

— Бит 7 **ARPE**: Разрешение буфера перезагрузки TIMx_ARR

0: Нельзя

1: Нужно

— Бит 6:4 Резерв, не трогать.

— Бит 3 **OPM**: Режим одного импульса

0: Счётчик не останавливается

1: Счётчик останавливается по следующему событию обновления (снимается бит CEN)

— Бит 2 **URS**: Источник запроса по событию UEV.

Изменяется программно.

0: Разрешённые запросы прерывания выдаются по:

– Переполнение счётчика

– Установка бита UG

1: Разрешённые запросы прерывания выдаются только по Переполнению счётчика.

— Бит 1 **UDIS:** Выключение выдачи события обновления UEV.

Изменяется программно.

0: Событие UEV выдаётся по:

– Переполнение счётчика

– Установка бита UG

1: Событие UEV не выдаётся, скрытые регистры (ARR, PSC, CCRx) хранят значение. При стоящем бите UG счётчик и предделитель инициализируются.

— Бит 0 **CEN:** Включение счётчика.

0: Выключен

1: Включён

NB: В режиме Одного импульса бит CEN снимается аппаратно по событию обновления.

19.4.2. Регистр управления режима ведомого TIM9/12 (TIMx_SMCR)

Смещение адреса: 0x08

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									TS[2:0]			Res.	SMS[2:0]		
									rw	rw	rw		rw	rw	rw

— Биты 15:7 Резерв, не трогать.

— Биты 6:4 **TS[2:0]**: Выбор запуска

000: Внутренний 0 (ITR0)

001: Внутренний 1 (ITR1)

010: Внутренний 2 (ITR2)

011: Внутренний 3 (ITR3)

100: Детектор фронта TI1 (TI1F_ED)

101: Фильтрованный вход 1 (TI1FP1)

110: Фильтрованный вход 2 (TI2FP2)

111: Резерв

NB: Менять их можно только если не используются (например, при SMS=000) во избежание неверного определения фронта.

— Бит 3 Резерв, не трогать.

— Биты 2:0 **SMS[2:0]**: Выбор режима ведомого

При внешнем сигнале активный фронт TRGI подключается к внешнему входу с выбранной полярностью.

000: Режим ведомого выключен - если CEN = '1', то предделитель работает напрямую от внутренних тактов.

001: Резерв.

010: Резерв.

011: Резерв.

100: Режим сброса - Передний фронт TRGI инициализирует счётчик и запускает обновление регистров.

101: Вентильный режим - Такты счётчика разрешены при высоком TRGI. При низком TRGI счётчик останавливается, но не сбрасывается. Старт и стоп счётчика управляемые.

110: Режим запуска - Счётчик стартует по переднему фронту TRGI (но не сбрасывает). Управляемый только старт.

111: Внешние такты 1 - Счёт передних фронтов TRGI.

NB: Вентильный режим нельзя использовать если входом запуска выбран TI1F_ED (TS='100'). В действительности, TI1F_ED выдаёт 1 импульс на каждую передачу TI1F, тогда как вентильный режим проверяет уровень сигнала запуска.

Такты ведомого таймера нужно включать до приёма событий от ведущего таймера и не должны изменяться налету.

Таблица 101. Подключение внутреннего запуска TIMx.

Ведомый TIM	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
TIM9	TIM2_TRGO	TIM3_TRGO	TIM10_OC	TIM11_OC
TIM12	TIM4_TRGO	TIM5_TRGO	TIM13_OC	TIM14_OC

19.4.3. Регистр разрешения прерываний TIM9/TIM12 (TIMx_DIER)

Смещение адреса: 0x0C

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									TIE	Res			CC2IE	CC1IE	UIE
									rw				rw	rw	rw

Содержимое битов:

0: Нельзя

1: Можно

— Биты 15:7 Резерв, не трогать.

— Бит 6 **TIE**: Запрос прерывания запуска

— Биты 5:3 Резерв, не трогать.

— Бит 2 **CC2IE**: Запрос прерывания захвата/сравнения 2

— Бит 1 **CC1IE**: Запрос прерывания захвата/сравнения 1

— Бит 0 **UIE**: Запрос прерывания обновления

19.4.4. Регистр состояния TIM9/TIM12 (TIMx_SR)

Биты ставятся аппаратно, стираются записью 0.

Перезахват это появление события при стоящем его флаге.

Смещение адреса: 0x10

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					CC2OF	CC1OF	Reserved		TIF	Reserved			CC2IF	CC1IF	UIF
					rc_w0	rc_w0			rc_w0				rc_w0	rc_w0	rc_w0

Содержимое битов:

0: Не было

1: Было

— Биты 15:11 Резерв, не трогать.

— Бит 10 **CC2OF**: Флаг перезахвата захвата/сравнения 2

— Бит 9 **CC1OF**: Флаг перезахвата захвата/сравнения 1

— Бит 8:7 Резерв, не трогать.

— Бит 6 **TIF**: Флаг прерывания запуска

Ставится аппаратно по активному фронту на входе TRGI при работающем контроллере режима ведомого во всех режимах, кроме Вентильного, (тогда по обоим фронтам).

— Биты 5:3 Резерв, не трогать.

— Бит 2 **CC2IF**: Флаг прерывания захвата/сравнения 2

— Бит 1 **CC1IF**: Флаг прерывания захвата/сравнения 1

В режиме вывода CC1:

0: Не было

1: Счётчик TIMx_CNT сравнивается с TIMx_CCR1. При TIMx_CCR1 > TIMx_ARR, флаг CC1IF ставится по переполнению счётчика.

В режиме ввода CC1:

0: Не было

1: В TIMx_CCR1 лежит захваченное значение.

— Бит 0 **UIF**: Флаг прерывания обновления

Удержание прерывания обновления ставится при:

– Переполнении счётчика и UDIS=0 в регистре TIMx_CR1.

– При программной инициализации CNT битом UG в регистре TIMx_EGR, если URS=0 и UDIS=0 в регистре TIMx_CR1.

– При инициализации CNT событием запуска (см. регистр TIMx_SMCR), если URS=0 и UDIS=0 в регистре TIMx_CR1.

19.4.5. Регистр генерации событий (TIMx_EGR)

Событие генерируется программной установкой бита, снимается он аппаратно.

Смещение адреса: 0x14

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									TG	Reserved			CC2G	CC1G	UG
									w				w	w	w

— Биты 15:7 Резерв, не трогать.

— Бит 6 **TG**: Запуск.

1: Ставит флаг TIF в регистре TIMx_SR. Может возникнуть прерывание.

— Биты 5:3 Резерв, не трогать.

— Бит 2 **CC2G**: Захват/сравнение 2.

— Бит 1 **CC1G**: Захват/сравнение 1.

0: Ничего.

1: Выдаёт событие захвата/сравнения:

В режиме вывода CC1:

Ставит флаг CC1IF. Может возникнуть прерывание.

В режиме ввода CC1:

Текущее значение счётчика пишется в TIMx_CCR1. Ставит флаг CC1IF. Может возникнуть прерывание. При уже стоящем флаге CC1IF ставится и флаг CC1OF.

— Бит 0 **UG**: Обновление.

1: Инициализирует счётчик и обновление регистров. Текущий счётчик предделителя чистится не влияя на само значение. В режиме по центру или при DIR=0 счётчик обнуляется, иначе перегружается из регистра TIMx_ARR (при DIR=1).

19.4.6. Регистр режима захвата/сравнения 1 TIM9/TIM12 (TIMx_CCMR1)

Смещение адреса: 0x18

По сбросу: 0x0000

Каналы могут работать на ввод (захват) и вывод (сравнение), что определяется битами CCxS. У остальных битов для ввода и вывода значение отличается. OCxx обозначает функции вывода, ICxx функции ввода. При этом используются разные каскады.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				IC1F[3:0]				IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Сравнение выхода:

— Бит 15 **OC2CE:** Разрешение очистки сравнения 2.

— Биты 14:12 **OC2M[2:0]**: Режим сравнения выхода 2.

— Бит 11 **OC2PE**: Разрешение предзагрузки сравнения 2.

— Бит 10 **OC2FE**: Быстрое разрешение сравнения 2.

— Биты 9:8 **CC2S[1:0]**: Выбор Захвата/Сравнения 2.

Определяет направление канала (ввод/вывод) и используемый вход.

00: Канал CC2 выходной

01: Канал CC2 входной, IC2 на TI2

10: Канал CC2 входной, IC2 на TI1

11: Канал CC2 входной, IC2 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.

NB: Биты CC2S можно писать только при выключенном канале (CC2E = ‘0’ в TIMx_CCER).

— Бит 7 **OC1CE**: Разрешение очистки сравнения выхода 1.

0: ETRF на OC1REF не влияет

1: Высокий уровень на ETRF стирает OC1REF

— Биты 6:4 **OC1M[2:0]**: Режим сравнения выхода 1.

Определяют поведение выходного опорного сигнала OC1REF, от которого происходят OC1 и OC1N. OC1REF активен высоким, а OC1 и OC1N зависят от битов CC1P и CC1NP.

000: Заморозка - сравнение TIMx_CCR1 и TIMx_CNT на выходы не влияет (режим используется для генерации временной базы).

001: При совпадении TIMx_CNT и TIMx_CCR1 сигнал OC1REF ставится активным (высоким).

010: При совпадении TIMx_CNT и TIMx_CCR1 сигнал OC1REF ставится неактивным (низким).

011: Переключение - при TIMx_CNT=TIMx_CCR1 сигнал OC1REF перебрасывается.

100: Принудительный неактивный - OC1REF ставится низким.

101: Принудительный активный - OC1REF ставится высоким.

110: ШИМ 1 - при прямом счёте канал активен пока TIMx_CNT<TIMx_CCR1. При обратном счёте канал неактивен (OC1REF=‘0’) пока TIMx_CNT>TIMx_CCR1.

111: ШИМ 2 - при прямом счёте канал неактивен пока TIMx_CNT<TIMx_CCR1. При обратном счёте канал активен пока TIMx_CNT>TIMx_CCR1.

NB: В ШИМ 1 и 2 уровень OCREF изменяется только когда изменяется результат сравнения или при переключении из “Заморозки” в “ШИМ”.

— Бит 3 **OC1PE**: Разрешение регистра предзагрузки.

0: Предзагрузка TIMx_CCR1 выключена. TIMx_CCR1 можно писать в любое время, новое значение начинает действовать незамедлительно.

1: Предзагрузка TIMx_CCR1 включена. Доступ идёт к регистру предзагрузки. TIMx_CCR1 пишется в активный регистр по событию обновления.

NB: Режим ШИМ без установки регистра предзагрузки можно использовать только в режиме одного импульса (стоит бит OPM в регистре TIMx_CR1). Иначе поведение непредсказуемо.

— Бит 2 **OC1FE**: Разрешение быстрого сравнения.

Ускоряет действие события запуска по входу на выход CC.

0: CC1 работает как обычно (сравнение счётчика и CCR1) даже при включённом запуске. Минимальная задержка переключения выхода CC1 при фронте на входе составляет 5 тактов.

1: Активный фронт запуска по входу действует как сравнение на выходе CC1. Тогда OC ставится в уровень сравнения независимо от его результата. Задержка между импульсом на входе и выходом CC1 сокращается до 3 тактов. OCFE работает только в ШИМ 1 и ШИМ 2.

— Биты 1:0 **CC1S**: Выбор Захвата/Сравнения 1.

Определяет направление канала (ввод/вывод) и используемый вход.

00: Канал CC1 выходной

01: Канал CC1 входной, IC1 на TI1

10: Канал CC1 входной, IC1 на TI2

11: Канал CC1 входной, IC1 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.

NB: Биты CC1S можно писать только при выключенном канале (CC1E = ‘0’ в TIMx_CCER).

Захват входа.

— Биты 15:12 **IC2F**: Фильтр захвата входа 2.

— Биты 11:10 **IC2PSC[1:0]** : Предделитель захвата входа 2.

— Биты 9:8 **CC2S[1:0]**: Выбор Захвата/Сравнения 2.

Определяет направление канала (ввод/вывод) и используемый вход.

00: Канал CC2 выходной

01: Канал CC2 входной, IC2 на TI2

10: Канал CC2 входной, IC2 на TI1

11: Канал CC2 входной, IC2 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.

NB: Биты CC2S можно писать только при выключенном канале (CC2E = ‘0’ в TIMx_CCER).

— Биты 7:4 **IC1F[3:0]**: Фильтр захвата входа 1.

Определяет частоту выборки и длину цифрового фильтра сигнала TI1. Цифровой фильтр это счётчик, выдающий сигнал на выход после N последовательных событий на входе:

0000: Без фильтра, выборка на частоте f_{DTS}

0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2

0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4

0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8

0100: $f_{SAMPLING}=f_{DTS}/2$, N=6

0101: $f_{SAMPLING}=f_{DTS}/2$, N=8

0110: $f_{SAMPLING}=f_{DTS}/4$, N=6

0111: $f_{SAMPLING}=f_{DTS}/4$, N=8

1000: $f_{SAMPLING}=f_{DTS}/8$, N=6

1001: $f_{SAMPLING}=f_{DTS}/8$, N=8

1010: $f_{SAMPLING}=f_{DTS}/16$, N=5

1011: $f_{SAMPLING}=f_{DTS}/16$, N=6

1100: $f_{SAMPLING}=f_{DTS}/16$, N=8

1101: $f_{SAMPLING}=f_{DTS}/32$, N=5

1110: $f_{SAMPLING}=f_{DTS}/32$, N=6

1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

— Биты 3:2 **IC1PSC**: Предделитель захвата входа 1

Значение предделителя на входе CC1 (IC1).

По CC1E=’0’ (регистр TIMx_CCER) предделитель сбрасывается.

00: без предделителя, по каждому фронту на входе

01: каждые 2 события

10: каждые 4 события

11: каждые 8 события

— Биты 1:0 **CC1S**: Выбор Захвата/Сравнения 1.

Определяет направление канала (ввод/вывод) и используемый вход.

00: Канал CC1 выходной

01: Канал CC1 входной, IC1 на TI1

10: Канал CC1 входной, IC1 на TI2

11: Канал CC1 входной, IC1 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.

NB: Биты CC1S можно писать только при выключенном канале (CC1E = ‘0’ в TIMx_CCER).

19.4.7. Регистр разрешения захвата/сравнения (TIMx_CCER)

Смещение адреса: 0x20

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
								rw		rw	rw	rw		rw	rw

— Биты 15:8 Резерв, не трогать.
— Бит 7 **CC2NP**: Полярность инверсного выхода захвата/сравнения 2
— Бит 6 Резерв, не трогать.
— Бит 5 **CC2P**: Полярность выхода захвата/сравнения 2
— Бит 4 **CC2E**: Разрешение выхода захвата/сравнения 2
— Бит 3 **CC1NP**: Полярность инверсного выхода захвата/сравнения 1
Выходной CC1: CC1NP должен быть чистым
Входной CC1: CC1NP вместе с CC1P определяют полярность TI1FP1/TI2FP1 (см. CC1P).
— Бит 2 Резерв, не трогать.
— Бит 1 **CC1P**: Полярность выхода захвата/сравнения 1
1: Выдаёт событие захвата/сравнения:
В режиме вывода CC1:
0: OC1 активен высоким.
1: OC1 активен низким.
В режиме ввода CC1:
Выбирает для запуска или захвата прямой или инверсный IC1.
Биты CC1NP/CC1P определяют полярность TI1FP1 TI2FP1 запуска операции.
00: прямой/передний фронт и уровень
01: обратный/задний фронт и уровень
10: резерв.
11: неинверсные оба фронта и уровень
— Бит 0 **CC1E**: Разрешение выхода захвата/сравнения 1
В режиме вывода CC1:
0: Выкл. - OC1 не активен.
1: Вкл. - OC1 выводится на соответствующую ножку.
В режиме ввода CC1:
0: Захват выключен.
1: Захват включен.

Таблица 102. Управляющие биты стандартных OCx каналов

Бит CCxE	Состояние OCx
0	Выключен (OCx=0, OCx_EN=0)
1	OCx=OCxREF + Полярность, OCx_EN=1

NB: Состояние внешних ножек I/O, подключённых к стандартным OCx, зависит от состояния OCx и регистров GPIO и AFIO.

19.4.8. Счётчик TIM9/12 (TIMx_CNT)

Смещение адреса: 0x24
По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:0 **CNT[15:0]**: Значение счётчика.

19.4.9. Предделитель TIM9/12 (TIMx_PSC)

Смещение адреса: 0x28
По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:0 **PSC[15:0]**: Значение предделителя

Частота счёта (CK_CNT) равна f_{CK_PSC} / (PSC[15:0] + 1).

Значение PSC пишется в рабочий регистр предделителя по каждому событию обновления (включая очистку счётчика битом UG регистра TIMx_EGR или запуском в режиме Сброса).

19.4.10.Регистр предзагрузки TIM9/12 (TIMx_ARR)

Смещение адреса: 0x2C

По сбросу: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:0 **ARR[15:0]**: Значение перезагрузки.

Если значение нулевое, то счётчик блокируется.

19.4.11.Регистр 1 захвата/сравнения TIM9/12 (TIMx_CCR1)

Смещение адреса: 0x34

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

— Биты 15:0 **CCR1[15:0]**: Значение захвата/сравнения

В режиме вывода CC1:

Значение предзагрузки CCR1 надо писать в рабочий регистр захвата/сравнения. Если функция предзагрузки не включена (бит OC1PE в регистре TIMx_CCMR1), то он пишется постоянно, иначе только по событию обновления.

Результат сравнения рабочего регистра и TIMx_CNT подаётся на выход OC1.

В режиме ввода CC1:

CCR1 содержит значение счётчика по последнему событию захвата 1 (IC1). Здесь TIMx_CCR1 только читается.

19.4.12.Регистр 2 захвата/сравнения TIM9/12 (TIMx_CCR2)

Смещение адреса: 0x38

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

— Биты 15:0 **CCR2[15:0]**: Значение захвата/сравнения

В режиме вывода CC2:

Значение предзагрузки CCR2 надо писать в рабочий регистр захвата/сравнения. Если функция предзагрузки не включена (бит OC2PE в регистре TIMx_CCMR2), то он пишется постоянно, иначе только по событию обновления.

Результат сравнения рабочего регистра и TIMx_CNT подаётся на выход OC2.

В режиме ввода CC2:

CCR2 содержит значение счётчика по последнему событию захвата 2 (IC2). Здесь TIMx_CCR2 только читается.

19.4.13.Карта регистров TIM9/12

0x00	**TIMx_CR1**	Reserved							CKD [1:0]		ARPE	Reserved			OPM	URS	UDIS	CEN
	Reset value								0	0	0				0	0	0	0
0x08	**TIMx_SMCR**	Reserved										TS[2:0]			Reserved	SMS[2:0]		
	Reset value											0	0	0		0	0	0
0x0C	**TIMx_DIER**	Reserved										TIE	Reserved			CC2IE	CC1IE	UIE
	Reset value											0				0	0	0
0x10	**TIMx_SR**	Reserved						CC2OF	CC1OF	Reserved		TIF	Reserved			CC2IF	CC1IF	UIF
	Reset value							0	0			0				0	0	0
0x14	**TIMx_EGR**	Reserved										TG	Reserved			CC2G	CC1G	UG
	Reset value											0				0	0	0
0x18	**TIMx_CCMR1** Output Compare mode	Reserved		OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		Reserved	OC1M [2:0]			OC1PE	OC1FE	CC1 S [1:0]	
	Reset value			0	0	0	0	0	0	0		0	0	0	0	0	0	0
	TIMx_CCMR1 Input Capture mode	Reserved	IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1 S [1:0]	
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	Reserved																	
0x20	**TIMx_CCER**	Reserved									CC2NP	Reserved	CC2P	CC2E	CC1NP	Reserved	CC1P	CC1E
	Reset value										0		0	0	0		0	0
0x24	**TIMx_CNT**	Reserved	CNT[15:0]															
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIMx_PSC**	Reserved	PSC[15:0]															
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIMx_ARR**	Reserved	ARR[15:0]															
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	Reserved																	
0x34	**TIMx_CCR1**	Reserved	CCR1[15:0]															
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	**TIMx_CCR2**	Reserved	CCR2[15:0]															
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C to 0x4C	Reserved																	

19.5. Регистры TIM10/11/13/14

32-бит регистры надо писать словами. Остальные регистры надо писать полусловами или словами. Читать можно байтами,полусловами или словами.

19.5.1. Регистр управления 1 TIM10/11/13/14 (TIMx_CR1)

Смещение адреса: 0x00

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKD[1:0]		ARPE	Reserved			OPM	URS	UDIS	CEN
						rw	rw	rw				rw	rw	rw	rw

— Биты 15:10 Резерв, не трогать.

— Биты 9:8 **CKD[1:0]:** Деление тактов.

Это отношение тактов таймера (CK_INT) и тактов задержки и выборки (t_{DTS}) в генераторах задержки и цифровых фильтрах (ETR, TIx),

00: $t_{DTS}=t_{CK\_INT}$

01: $t_{DTS}=2*t_{CK\_INT}$

10: $t_{DTS}=4*t_{CK\_INT}$

11: Резерв

— Бит 7 **ARPE**: Разрешение буфера перезагрузки TIMx_ARR

0: Нельзя

1: Нужно

— Бит 6:4 Резерв, не трогать.

— Бит 3 **OPM**: Режим одного импульса

0: Счётчик не останавливается

1: Счётчик останавливается по следующему событию обновления (снимается бит CEN)

— Бит 2 **URS**: Источник запроса по событию UEV.

Изменяется программно.

0: Разрешённые запросы прерывания выдаются по:

– Переполнение счётчика

– Установка бита UG

1: Разрешённые запросы прерывания выдаются только по Переполнению счётчика.

— Бит 1 **UDIS:** Выключение выдачи события обновления UEV.

Изменяется программно.

0: Событие UEV выдаётся по:

– Переполнение счётчика

– Установка бита UG

1: Событие UEV не выдаётся, скрытые регистры (ARR, PSC, CCRx) хранят значение. При стоящем бите UG счётчик и предделитель инициализируются.

— Бит 0 **CEN:** Включение счётчика.

0: Выключен

1: Включён

NB: В режиме Одного импульса бит CEN снимается аппаратно по событию обновления.

19.5.2. Регистр разрешения прерываний TIM10/11/13/14 (TIMx_DIER)

Смещение адреса: 0x0C

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														CC1IE	UIE
														rw	rw

Содержимое битов:

0: Нельзя

1: Можно

— Биты 15:2 Резерв, не трогать.

— Бит 6 **TIE**: Запрос прерывания запуска

— Бит 1 **CC1IE**: Запрос прерывания захвата/сравнения 1

— Бит 0 **UIE**: Запрос прерывания обновления

19.5.3. Регистр состояния TIM10/11/13/14 (TIMx_SR)

Биты ставятся аппаратно, стираются записью 0.

Перезахват это появление события при стоящем его флаге.

Смещение адреса: 0x10

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CC1OF	Reserved							CC1IF	UIF
						rc_w0								rc_w0	rc_w0

Содержимое битов:

0: Не было

1: Было

— Биты 15:10 Резерв, не трогать.

— Бит 9 **CC1OF**: Флаг перезахвата захвата/сравнения 1

— Бит 8:3 Резерв, не трогать.

— Бит 2 **CC2IF**: Флаг прерывания захвата/сравнения 2

— Бит 1 **CC1IF**: Флаг прерывания захвата/сравнения 1

В режиме вывода CC1:

0: Не было

1: Счётчик TIMx_CNT сравнивается с TIMx_CCR1. При TIMx_CCR1 > TIMx_ARR, флаг CC1IF ставится по переполнению счётчика.

В режиме ввода CC1:

0: Не было

1: В TIMx_CCR1 лежит захваченное значение.

— Бит 0 **UIF**: Флаг прерывания обновления

Удержание прерывания обновления ставится при:

– Переполнении и UDIS=0 в регистре TIMx_CR1.

– При программной инициализации CNT битом UG в регистре TIMx_EGR, если URS=0 и UDIS=0 в регистре TIMx_CR1.

19.5.4. Регистр генерации событий TIM10/11/13/14 (TIMx_EGR)

Событие генерируется программной установкой бита, снимается он аппаратно.

Смещение адреса: 0x14

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														CC1G	UG
														w	w

— Биты 15:2 Резерв, не трогать.

— Бит 1 **CC1G**: Захват/сравнение 1.

0: Ничего.

1: Выдаёт событие захвата/сравнения:

В режиме вывода CC1:

Ставит флаг CC1IF. Может возникнуть прерывание.

В режиме ввода CC1:

Текущее значение счётчика пишется в TIMx_CCR1. Ставит флаг CC1IF. Может возникнуть прерывание. При уже стоящем флаге CC1IF ставится и флаг CC1OF.

— Бит 0 **UG**: Обновление.

1: Инициализирует счётчик и обновление регистров. Текущий счётчик предделителя чистится не влияя на само значение.

19.5.5. Регистр режима захвата/сравнения TIM10/11/13/14 (TIMx_CCMR1)

Смещение адреса: 0x18

По сбросу: 0x0000

Каналы могут работать на ввод (захват) и вывод (сравнение), что определяется битами `CCxS`. У остальных битов для ввода и вывода значение отличается. `OCxx` обозначает функции вывода, `ICxx` функции ввода. При этом используются разные каскады.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
Reserved								IC1F[3:0]				IC1PSC[1:0]			
								rw	rw	rw	rw	rw	rw	rw	rw

Сравнение выхода:

— Бит 15:7 Резерв, не трогать.

— Биты 6:4 **OC1M[2:0]**: Режим сравнения выхода 1.

Определяют поведение выходного опорного сигнала OC1REF, от которого происходят OC1 и OC1N. OC1REF активен высоким, а OC1 и OC1N зависят от битов CC1P и CC1NP.

000: Заморозка - сравнение TIMx_CCR1 и TIMx_CNT на выходы не влияет (режим используется для генерации временной базы).

001: При совпадении TIMx_CNT и TIMx_CCR1 сигнал OC1REF ставится активным (высоким).

010: При совпадении TIMx_CNT и TIMx_CCR1 сигнал OC1REF ставится неактивным (низким).

011: Переключение - при TIMx_CNT=TIMx_CCR1 сигнал OC1REF перебрасывается.

100: Принудительный неактивный - OC1REF ставится низким.

101: Принудительный активный - OC1REF ставится высоким.

110: ШИМ 1 - при прямом счёте канал активен пока TIMx_CNT<TIMx_CCR1. При обратном счёте канал неактивен (OC1REF='0') пока TIMx_CNT>TIMx_CCR1.

111: ШИМ 2 - при прямом счёте канал неактивен пока TIMx_CNT<TIMx_CCR1. При обратном счёте канал активен пока TIMx_CNT>TIMx_CCR1.

NB: В ШИМ 1 и 2 уровень OCREF изменяется только когда изменяется результат сравнения или при переключении из "Заморозки" в "ШИМ".

— Бит 3 **OC1PE**: Разрешение регистра предзагрузки.

0: Предзагрузка TIMx_CCR1 выключена. TIMx_CCR1 можно писать в любое время, новое значение начинает действовать незамедлительно.

1: Предзагрузка TIMx_CCR1 включена. Доступ идёт к регистру предзагрузки. TIMx_CCR1 пишется в активный регистр по событию обновления.

NB: Режим ШИМ без установки регистра предзагрузки можно использовать только в режиме одного импульса (стоит бит OPM в регистре TIMx_CR1). Иначе поведение непредсказуемо.

— Бит 2 **OC1FE**: Разрешение быстрого сравнения.

Ускоряет действие события запуска по входу на выход CC.

0: CC1 работает как обычно (сравнение счётчика и CCR1) даже при включённом запуске. Минимальная задержка переключения выхода CC1 при фронте на входе составляет 5 тактов.

1: Активный фронт запуска по входу действует как сравнение на выходе CC1. Тогда OC ставится в уровень сравнения независимо от его результата. Задержка между импульсом на входе и выходом CC1 сокращается до 3 тактов. OCFE работает только в ШИМ 1 и ШИМ 2.

— Биты 1:0 **CC1S**: Выбор Захвата/Сравнения 1.

Определяет направление канала (ввод/вывод) и используемый вход.

00: Канал CC1 выходной

01: Канал CC1 входной, IC1 на TI1

10: Канал CC1 входной, IC1 на TI2

11: Канал CC1 входной, IC1 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.

NB: Биты CC1S можно писать только при выключенном канале (CC1E = '0' в TIMx_CCER).

Захват входа.

— Биты 15:8 Резерв, не трогать.

— Биты 7:4 **IC1F[3:0]**: Фильтр захвата входа 1.

Определяет частоту выборки и длину цифрового фильтра сигнала TI1. Цифровой фильтр это счётчик, выдающий сигнал на выход после N последовательных событий на входе:

0000: Без фильтра, выборка на частоте f_{DTS}

0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2

0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4

0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8

0100: $f_{SAMPLING}=f_{DTS}/2$, N=6

0101: $f_{SAMPLING}=f_{DTS}/2$, N=8

0110: $f_{SAMPLING}=f_{DTS}/4$, N=6

0111: $f_{SAMPLING}=f_{DTS}/4$, N=8

1000: $f_{SAMPLING}=f_{DTS}/8$, N=6

1001: $f_{SAMPLING}=f_{DTS}/8$, N=8

1010: $f_{SAMPLING}=f_{DTS}/16$, N=5

1011: $f_{SAMPLING}=f_{DTS}/16$, N=6

1100: $f_{SAMPLING}=f_{DTS}/16$, N=8

1101: $f_{SAMPLING}=f_{DTS}/32$, N=5
1110: $f_{SAMPLING}=f_{DTS}/32$, N=6
1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

— Биты 3:2 **IC1PSC**: Предделитель захвата входа 1
Значение предделителя на входе CC1 (IC1).
По CC1E='0' (регистр TIMx_CCER) предделитель сбрасывается.
00: без предделителя, по каждому фронту на входе
01: каждые 2 события
10: каждые 4 события
11: каждые 8 событий

— Биты 1:0 **CC1S**: Выбор Захвата/Сравнения 1.
Определяет направление канала (ввод/вывод) и используемый вход.
00: Канал CC1 выходной
01: Канал CC1 входной, IC1 на TI1
10: Канал CC1 входной, IC1 на TI2
11: Канал CC1 входной, IC1 на TRC. Работает только если битом TS регистра TIMx_SMCR выбран внутренний запуск.
NB: Биты CC1S можно писать только при выключенном канале (CC1E = '0' в TIMx_CCER).

19.5.6. Разрешение захвата/сравнения TIM10/11/13/14 (TIMx_CCER)

Смещение адреса: 0x20
По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												CC1NP	Res.	CC1P	CC1E
												rw		rw	rw

— Биты 15:4 Резерв, не трогать.
— Бит 3 **CC1NP**: Полярность инверсного выхода захвата/сравнения 1
Выходной CC1: CC1NP должен быть чистым
Входной CC1: CC1NP вместе с CC1P определяют TI1FP1/TI2FP1 полярность (см. CC1P).
— Бит 2 Резерв, не трогать.
— Бит 1 **CC1P**: Полярность выхода захвата/сравнения 1
1: Выдаёт событие захвата/сравнения:
В режиме вывода CC1:
0: OC1 активен высоким.
1: OC1 активен низким.
В режиме ввода CC1:
Выбирает для запуска или захвата прямой или инверсный IC1.
Биты CC1NP/CC1P определяют полярность TI1FP1 TI2FP1 запуска операции.
00: прямой/передний фронт и уровень
01: обратный/задний фронт и уровень
10: резерв.
11: неинверсные оба фронта и уровень
— Бит 0 **CC1E**: Разрешение выхода захвата/сравнения 1
В режиме вывода CC1:
0: Выкл. - OC1 не активен.
1: Вкл. - OC1 выводится на соответствующую ножку.
В режиме ввода CC1:
0: Захват выключен.
1: Захват включен.

Таблица 103. Управляющие биты стандартных OCx каналов

Бит CCxE	Состояние OCx
0	Выключен (OCx=0, OCx_EN=0)
1	OCx=OCxREF + Полярность, OCx_EN=1

NB: Состояние внешних ножек I/O, подключённых к стандартным OCx, зависит от состояния OCx и регистров GPIO и AFIO.

19.5.7. Счётчик TIM10/11/13/14 (TIMx_CNT)

Смещение адреса: 0x24

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:0 **CNT[15:0]**: Значение счётчика.

19.5.8. Предделитель TIM10/11/13/14 (TIMx_PSC)

Смещение адреса: 0x28

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:0 **PSC[15:0]**: Значение предделителя

Частота счёта (CK_CNT) равна f_{CK_PSC} / (PSC[15:0] + 1).

Значение PSC пишется в рабочий регистр предделителя по каждому событию обновления (включая очистку счётчика битом UG регистра TIMx_EGR или запуском в режиме Сброса).

19.5.9. Регистр предзагрузки TIM10/11/13/14 (TIMx_ARR)

Смещение адреса: 0x2C

По сбросу: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:0 **ARR[15:0]**: Значение перезагрузки.

Если значение нулевое, то счётчик блокируется.

19.5.10.Регистр 1 захвата/сравнения TIM10/11/13/14 (TIMx_CCR1)

Смещение адреса: 0x34

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

— Биты 15:0 **CCR1[15:0]**: Значение захвата/сравнения

В режиме вывода CC1:

Значение предзагрузки CCR1 надо писать в рабочий регистр захвата/сравнения. Если функция предзагрузки не включена (бит OC1PE в регистре TIMx_CCMR1), то он пишется постоянно, иначе только по событию обновления.

Результат сравнения рабочего регистра и TIMx_CNT подаётся на выход OC1.

В режиме ввода CC1:

CCR1 содержит значение счётчика по последнему событию захвата 1 (IC1). Здесь TIMx_CCR1 только читается.

19.5.11.Карта регистров TIM10/11/13/14

0x00	**TIMx_CR1**	Reserved							CKD [1:0]		ARPE	Reserved			OPM	URS	UDIS	CEN
	Reset value								0	0	0				0	0	0	0
0x08	**TIMx_SMCR**	Reserved																
	Reset value																	
0x0C	**TIMx_DIER**	Reserved															CC1IE	UIE
	Reset value																0	0
0x10	**TIMx_SR**	Reserved							CC1OF	Reserved							CC1IF	UIF
	Reset value								0								0	0
0x14	**TIMx_EGR**	Reserved															CC1G	UG
	Reset value																0	0
0x18	**TIMx_CCMR1** Output compare mode	Reserved										OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value											0	0	0	0	0	0	0
	TIMx_CCMR1 Input capture mode	Reserved									IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value										0	0	0	0	0	0	0	0
0x1C	Reserved																	
0x20	**TIMx_CCER**	Reserved													CC1NP	Reserved	CC1P	CC1E
	Reset value														0		0	0
0x24	**TIMx_CNT**	Reserved	CNT[15:0]															
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIMx_PSC**	Reserved	PSC[15:0]															
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIMx_ARR**	Reserved	ARR[15:0]															
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	Reserved																	
0x34	**TIMx_CCR1**	Reserved	CCR1[15:0]															
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38 to 0x4C	Reserved																	

20. Базовые таймеры (TIM6 и TIM7)

20.1. Введение в TIM6 и TIM7

Это 16-бит само-перегружаемые таймеры с программируемым предделителем.

Они могут использоваться для создания временной базы и управления ЦАП через подключённые входы запуска.

Таймеры полностью независимы и не имеют общих ресурсов.

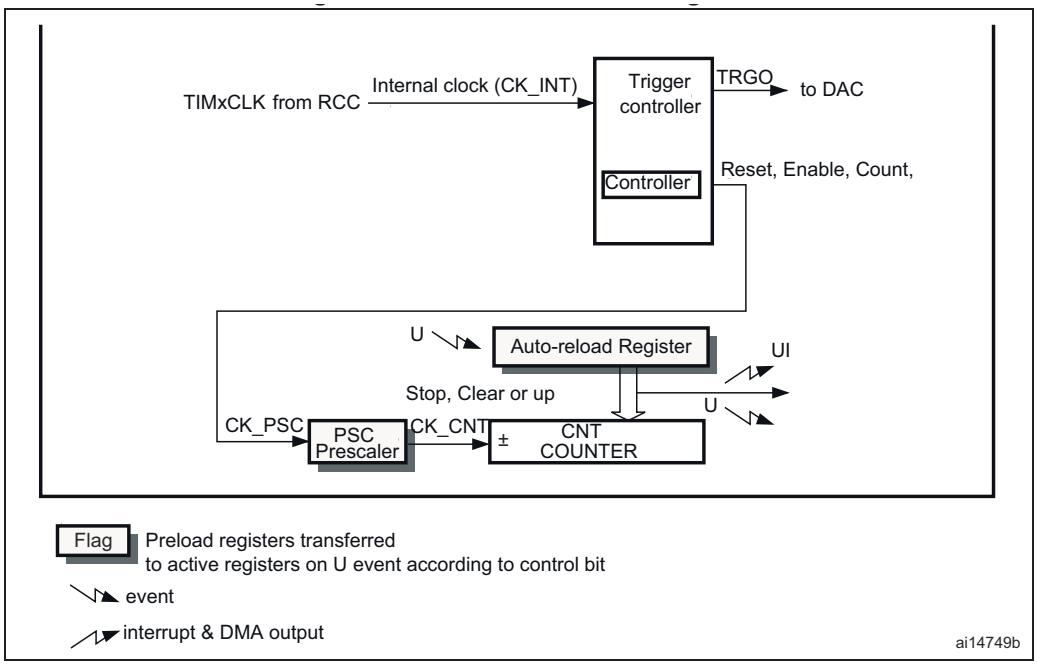
20.2. Основные свойства TIM6 и TIM7

Это:

- 16-бит прямой счётчик с перезагрузкой.
- 16-бит программируемый предделитель (можно “налету”) тактов на число от 1 до 65536.
- Схема синхронизации запуска ЦАП.

- Генерация Прерываний/DMA по переполнению счётчика.

Рис. 203. Блок-схема базового таймера.



20.3. Функциональное описание таймеров TIM6 и TIM7

20.3.1. Узел счёта

Это 16-бит счётчик с регистром перезагрузки. Тактирование может поступать через предделитель.

Счётчик, регистр перезагрузки и предделитель доступны программно для чтения и записи даже во время счёта.

Включает:

- Регистр счётчика (`TIMx_CNT`)
- Регистр предделителя (`TIMx_PSC`)
- Регистр перезагрузки (`TIMx_ARR`)

Регистр перезагрузки предзагружаемый (есть видимый и скрытый регистры). Содержимое видимого регистра пишется в скрытый постоянно или по каждому событию обновления (`UEV`), в зависимости от бита разрешения (`ARPE`) в регистре `TIMx_CR1`. Оно посылается программно или при переполнении/исчерпании счётчика при снятом бите `UDIS` в регистре `TIMx_CR1`.

Счётчик тактируется выходом предделителя `CK_CNT`. Разрешается битом `CEN` в регистре `TIMx_CR1`.

Счёт начинается через 1 такт после установки бита `CEN` в регистре `TIMx_CR1`.

Описание предделителя

Это 16-бит счётчик делителя входной частоты (от 1 до 65536) с видимым регистром `TIMx_PSC`. Его можно менять налету. Новое значение счётчика пишется по следующему событию обновления.

Рис. 204 Временная диаграмма с изменением предделителя с 1 на 2.

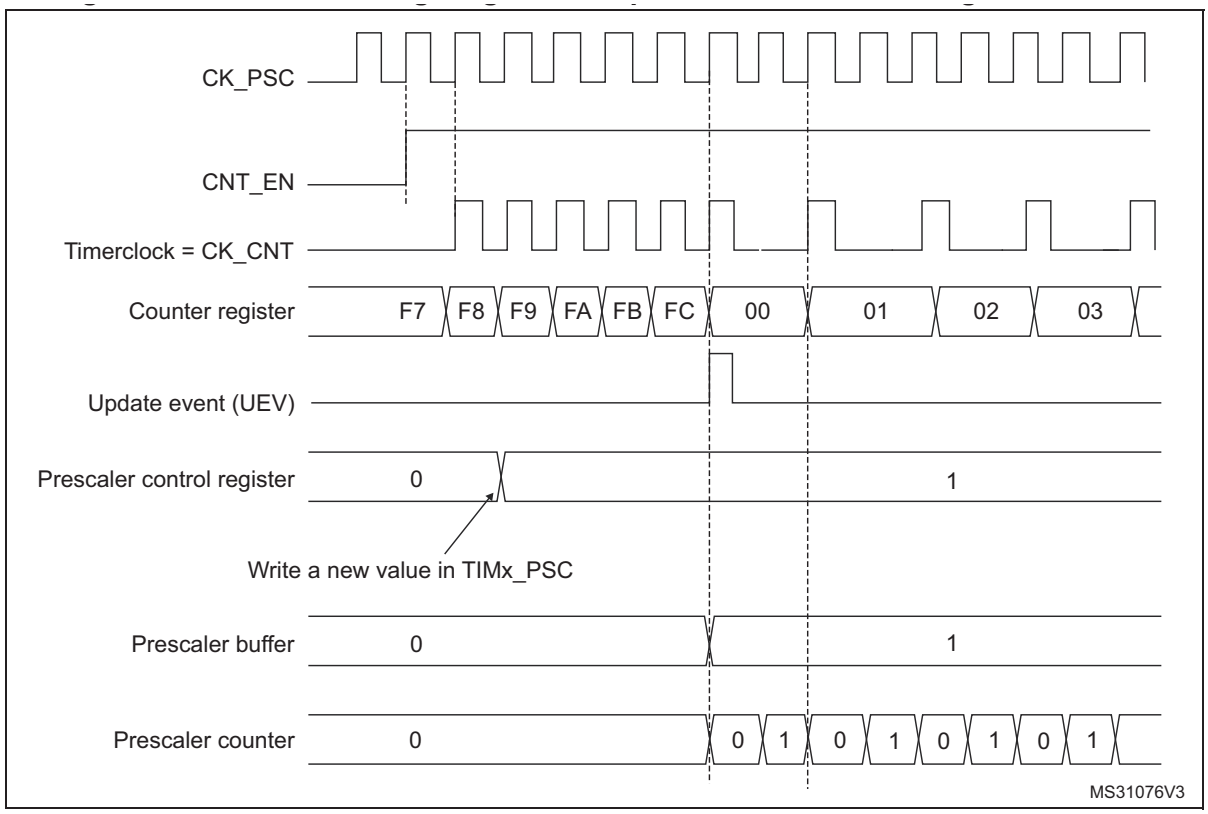
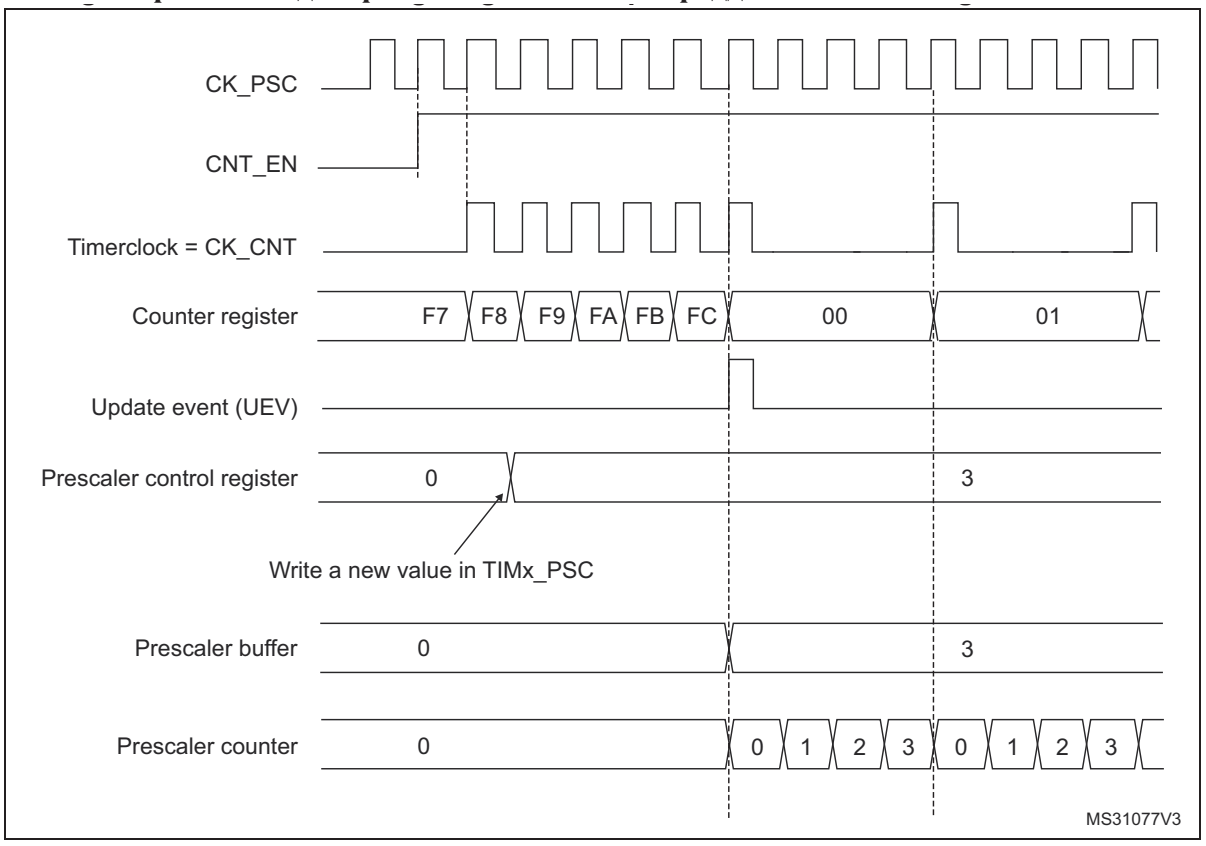


Рис. 205. Временная диаграмма с изменением предделителя с 1 на 4.



20.3.2. Режим счёта

Счёт идёт от 0 до значения перезагрузки (регистр TIMx_ARR), сбрасывается в 0 и выдаёт событие переполнения счётчика.

Событие обновления UEV выдаётся при каждом переполнении счётчика или установкой бита UG в регистре TIMx_EGR (программно или контроллером режима ведомого).

Событие UEV выключается установкой бита UDIS в регистре TIMx_CR1. Этим предупреждается запись скрытого регистра во время изменения регистра предзагрузки. Также событие UEV не

появляется при сброшенном бите UDIS 0. Однако, счётчик продолжает считать с 0, равно как и предделитель (не изменив своего значения). Кроме того, если стоит бит URS в регистре TIMx_CR1, то установка бита UG выдаёт событие UEV без установки флага UIF (без прерывания и запроса DMA). Этим предупреждается одновременная выдача прерываний обновления и захвата при очистке счётчика по событию захвата.

По событию обновления обновляются все регистры и ставится флаг обновления (бит UIF в регистре TIMx_SR). Это зависит от бита URS:

- В скрытый регистр перезагрузки пишется содержимое TIMx_ARR,
- Буфер предделителя перегружается из регистра TIMx_PSC.

Примеры ниже используют TIMx_ARR=0x36.

Рис. 206. Временная диаграмма с делителем 1.

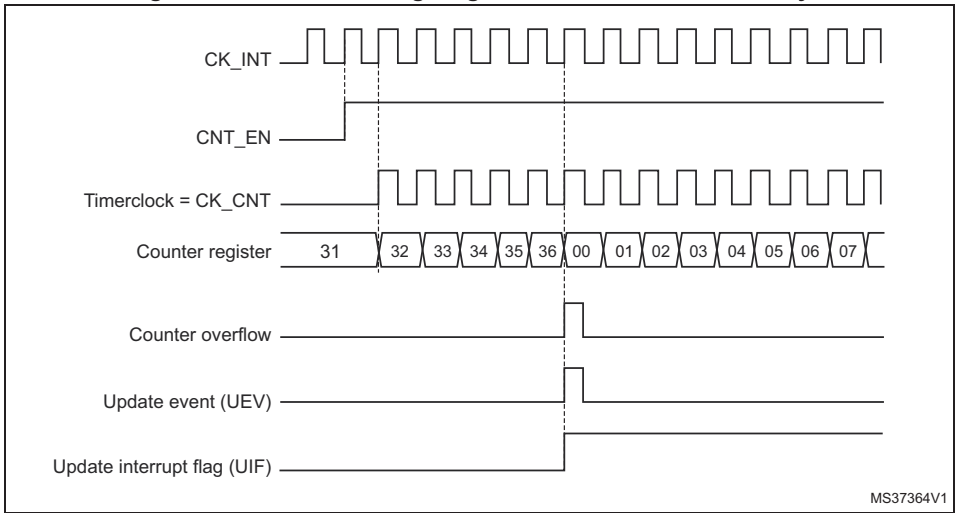


Рис. 207. Временная диаграмма с делителем 2.

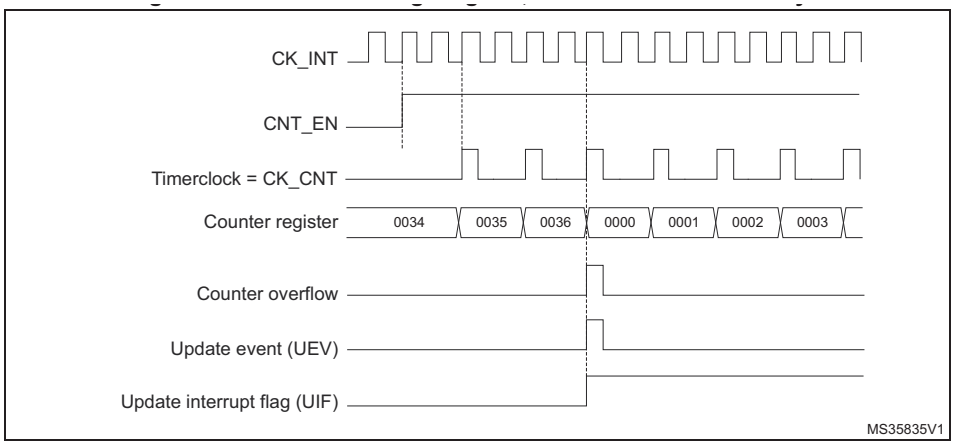


Рис. 208. Временная диаграмма с делителем 4.

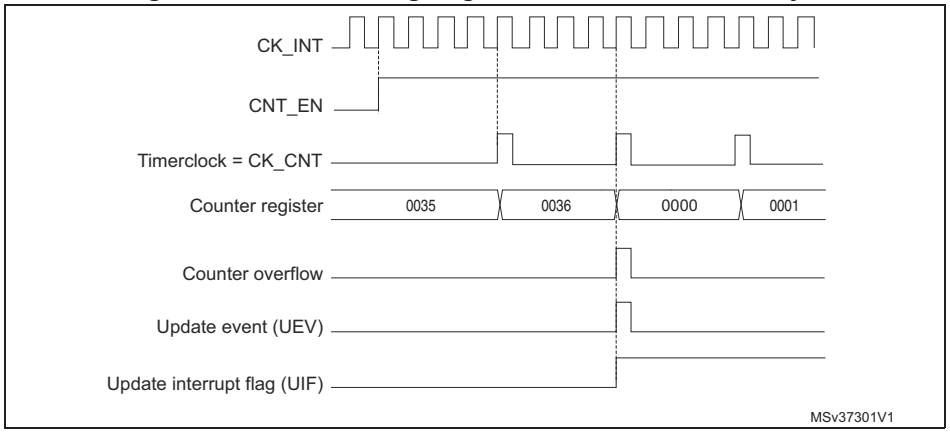


Рис. 209. Временная диаграмма с делителем N.

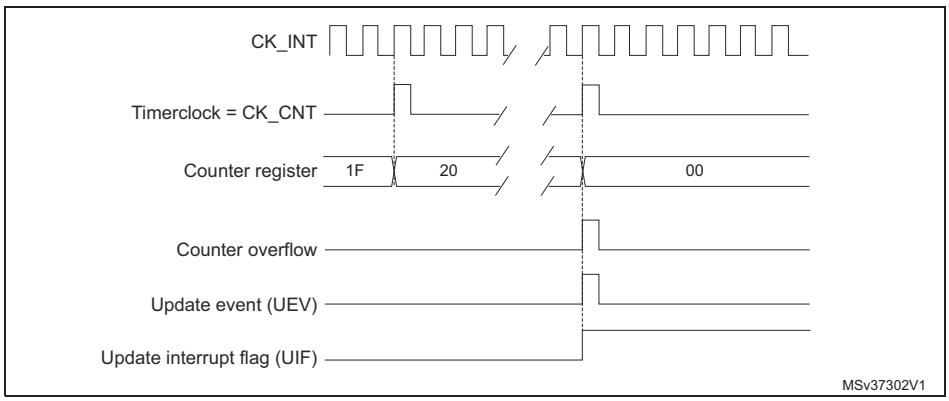


Рис. 210. Временная диаграмма события при ARPE=0 (TIMx_ARR не предзагружен).

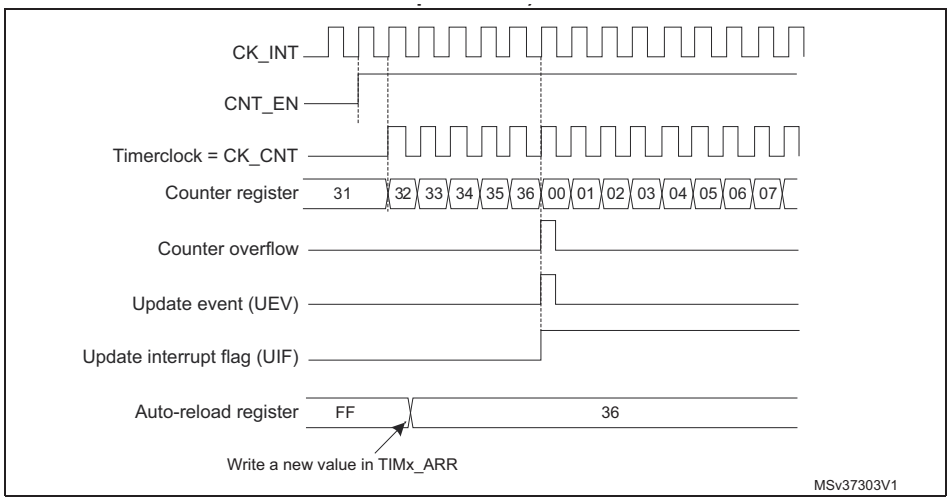
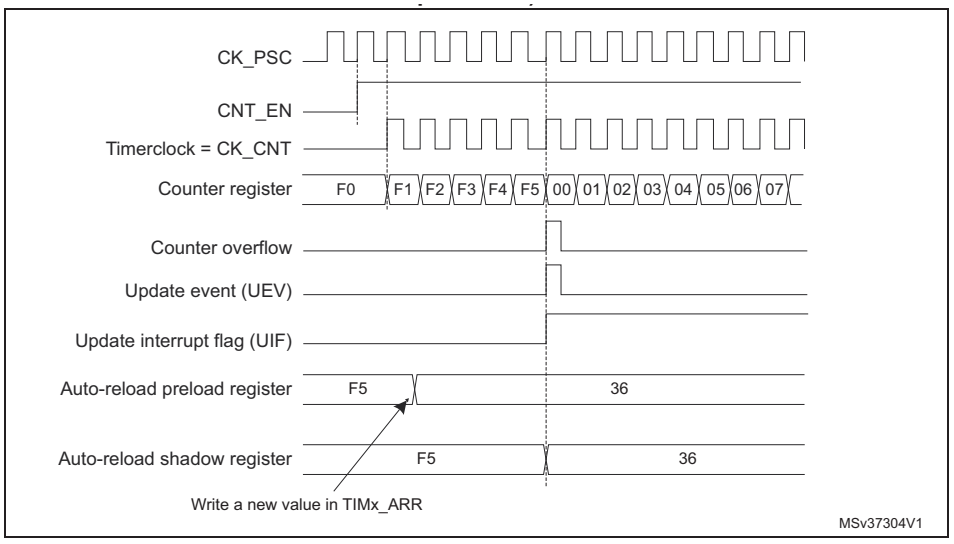


Рис. 211. Временная диаграмма события при ARPE=1 (TIMx_ARR предзагружен).

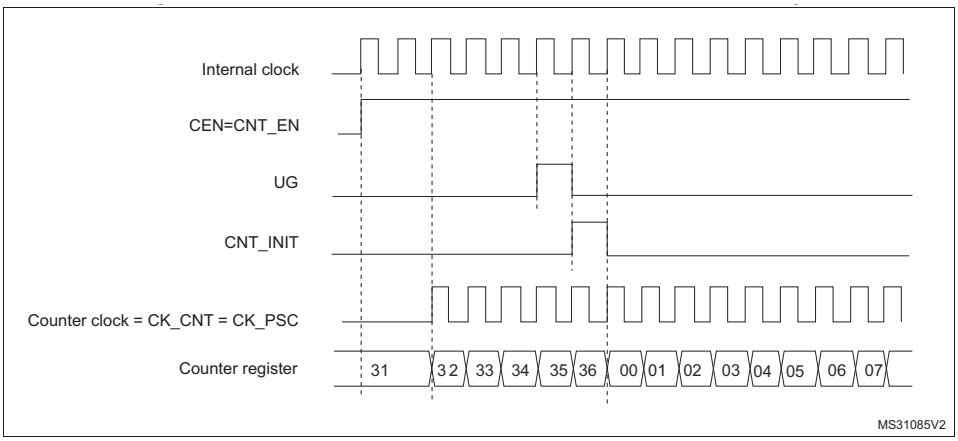


20.3.3. Выбор тактов

Счётчик может тактироваться внутреннего источника (CK_INT)

Всем управляют только биты CEN, DIR (в регистре TIMx_CR1) и UG (в регистре TIMx_EGR). Они изменяются только программно (кроме UG, снимающегося аппаратно). Сразу после установки бита CEN предделитель начинает получать внутренние такты CK_INT.

Рис. 212. Режим без предделителя.



20.3.4. Режим отладки

В режиме отладки (ядро Cortex-M3 остановлено), счётчик TIMx либо работает, либо останавливается, в зависимости от бита DBG_TIMx_STOP в модуле DBG.

20.4. Регистры TIM6 и TIM7

32-бит регистры надо писать словами. Остальные регистры надо писать полусловами или словами. Читать можно байтами, полусловами или словами.

20.4.1. Регистр управления 1 TIM6 и TIM7 (TIMx_CR1)

Смещение адреса: 0x00

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								ARPE	Reserved			OPM	URS	UDIS	CEN
								rw				rw	rw	rw	rw

— Биты 15:8 Резерв, не трогать.

— Бит 7 **ARPE**: Разрешение буфера перезагрузки TIMx_ARR

0: Нельзя

1: Нужно

— Бит 6:4 Резерв, не трогать.

— Бит 3 **OPM**: Режим одного импульса

0: Счётчик не останавливается

1: Счётчик останавливается по следующему событию обновления (снимается бит CEN)

— Бит 2 **URS**: Источник запроса по событию UEV.

Изменяется программно.

0: Разрешённые запросы прерывания/DMA выдаются по:

– Переполнение счётчика

– Установка бита UG

– Обновление от контроллера режима ведомого

1: Разрешённые запросы прерывания/DMA выдаются только по Переполнению счётчика.

— Бит 1 **UDIS:** Выключение выдачи события обновления UEV.

Изменяется программно.

0: Событие UEV выдаётся по:

– Переполнение счётчика

– Установка бита UG

– Обновление от контроллера режима ведомого

Регистры перегружаются значениями из регистров предзагрузки.

1: Событие UEV не выдаётся, скрытые регистры (ARR, PSC, CCRx) хранят значение. При стоящем бите UG или сигнале от контроллера режима ведомого счётчик и предделитель инициализируются.

— Бит 0 **CEN:** Включение счётчика.

0: Выключен

1: Включён

NB: Вентильный режим работает только при программно установленном бите CEN, режим запуска может ставить бит CEN аппаратно. В режиме Одного импульса бит CEN снимается аппаратно по событию обновления.

20.4.2. Регистр управления 2 (TIMx_CR2)

Смещение адреса: 0x04

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									MMS[2:0]			Reserved			
									rw	rw	rw				

— Биты 15:7 Резерв, не трогать.

— Биты 6:4 **MMS[2:0]:** Режим ведущего

Выбор информации для синхронизации ведомых таймеров (TRGO):

000: **Сброс** - Бит UG в регистре TIMx_EGR. Если на входе запуска идёт сигнал сброса (контроллер режима ведомого в сбросе), то сигнал на TRGO задерживается относительно реального сброса.

001: **Разрешение** - Сигнал разрешения счётчика (CNT_EN). Это полезно для одновременного запуска нескольких таймеров или управления окном ведомого таймера. Сигнал разрешения это объединение по OR бита CEN с входом запуска вентильного режима.

При подаче CNT_EN по входу запуска сигнал TRGO задерживается, кроме режима ведущий/ведомый (см. бит MSM в регистре TIMx_SMCR).

010: **Обновление** - На выход запуска (TRGO) направлено событие обновления. Например, ведущий таймер может быть предделителем ведомого.

NB: Тактирование ведомого таймера и ADC нужно включать до получения событий от ведущего таймера и не должно меняться налету.

— Биты 3:0 Резерв, не трогать.

20.4.3. Регистр разрешения прерываний TIM6 - TIM7 (TIMx_DIER)

Смещение адреса: 0x0C

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							UDE	Reserved							UIE
							rw								rw

Содержимое битов:

0: Нельзя

1: Можно

— Биты 15:9 Резерв, не трогать.

— Бит 8 **UDE**: Запрос обновления DMA

— Биты 7:1 Резерв, не трогать.

— Бит 0 **UIE**: Запрос прерывания обновления

20.4.4. Регистр состояния TIM6 - TIM7 (TIMx_SR)

Биты ставятся аппаратно, стираются записью 0.

Перезахват это появление события при стоящем его флаге.

Смещение адреса: 0x10

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															UIF
															rc_w0

Содержимое битов:

0: Не было

1: Было

— Бит 0 **UIF**: Флаг прерывания обновления

Удержание прерывания обновления ставится при:

– Переполнении счётчика и UDIS=0 в регистре TIMx_CR1.

– При программной инициализации CNT битом UG в регистре TIMx_EGR, если URS=0 и UDIS=0 в регистре TIMx_CR1.

20.4.5. Регистр генерации событий TIM6 - TIM7 (TIMx_EGR)

Событие генерируется программной установкой бита, снимается он аппаратно.

Смещение адреса: 0x14

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															UG
															w

— Биты 15:1 Резерв, не трогать.

— Бит 0 **UG**: Обновление.

1: Инициализирует счётчик и обновление регистров. Текущий счётчик предделителя чистится не влияя на само значение.

20.4.6. Счётчик TIM6 - TIM7 (TIMx_CNT)

Смещение адреса: 0x24

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:0 **CNT[15:0]**: Значение счётчика.

20.4.7. Предделитель TIM6 - TIM7 (TIMx_PSC)

Смещение адреса: 0x28

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:0 **PSC[15:0]**: Значение предделителя

Частота счёта (CK_CNT) равна f_{CK_PSC} / (PSC[15:0] + 1).

Значение PSC пишется в рабочий регистр предделителя по каждому событию обновления (включая очистку счётчика битом UG регистра TIMx_EGR или запуском в режиме Сброса).

20.4.8. Регистр предзагрузки TIM6 - TIM7 (TIMx_ARR)

Смещение адреса: 0x2C

По сбросу: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:0 **ARR[15:0]**: Значение перезагрузки.

Если значение нулевое, то счётчик блокируется.

20.4.9. Карта регистров TIM6 - TIM7

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIMx_CR1	Reserved																								ARPE	Reserved			OPM	URS	UDIS	CEN
	Reset value																									0				0	0	0	0
0x04	TIMx_CR2	Reserved																									MMS[2:0]			Reserved				
	Reset value																										0	0	0				
0x08	Reserved																																
0x0C	TIMx_DIER	Reserved																							UDE	Reserved							UIE
	Reset value																								0								0
0x10	TIMx_SR	Reserved																															UIF
	Reset value																																0
0x14	TIMx_EGR	Reserved																															UG
	Reset value																																0
0x18	Reserved																																
0x1C	Reserved																																
0x20	Reserved																																
0x24	TIMx_CNT	Reserved																CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIMx_PSC	Reserved																PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TIMx_ARR	Reserved																ARR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

21. Независимый сторожевой таймер (IWDG)

21.1. Введение в IWDG

MCU имеют два сторожевых таймера (Независимый и Оконный) для обнаружения программных отказов и запуска системного сброса и прерывания (только Оконный) при истечении заданного интервала времени.

Независимый сторож (IWDG) тактируется собственным выделенным низкочастотным генератором (LSI) и остаётся активным даже при сбое главных тактов. Оконный сторож (WWDG) тактируется предделителем тактов APB1 и может обнаруживать ненормально ранние и поздние события с поведении приложения.

IWDG подходит для полностью независимого контроля приложения, но с меньшей точностью по времени. WWDG пригоден для контроля поведения системы в рамках заданного временного окна.

21.2. Основные свойства IWDG

- Независимый обратный счётчик
- Тактируется независимым RC генератором (может работать в режимах Standby и Stop
- Выдаёт Сброс при достижении счётчиком 0x000

21.3. Функциональное описание IWDG

Если сторож запускается записью 0xCCCC в регистр ключа (IWDG_KR), то счёт начинается с числа 0xFFF. В конце счёта (0x000) выдаётся сигнал сброса IWDG.

Если в регистр IWDG_KR пишется ключ 0xAAAA, то в счётчик загружается значение регистра IWDG_RLR и отключается выдача сброса.

21.3.1. Аппаратный сторож

Если в битах конфигурации выбран “Аппаратный сторож”, то он автоматически запускается по включению питания и при завершении счёта выдаёт сброс независимо от какой-либо записи в регистр ключа.

21.3.2. Защита доступа

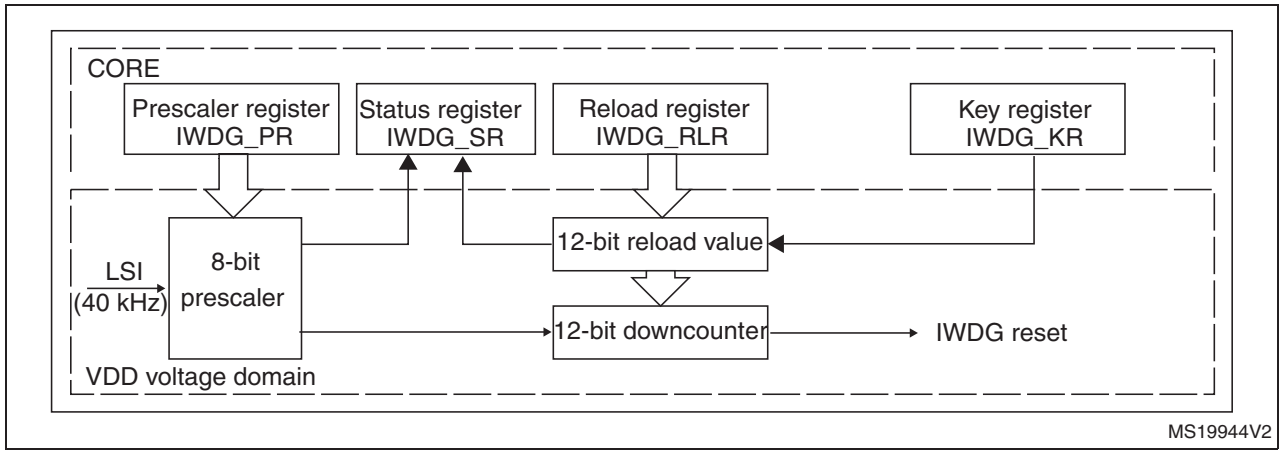
Регистры IWDG_PR и IWDG_RLR защищены от записи. Чтобы изменить их надо сначала записать 0x5555 в регистр IWDG_KR. Запись любого другого значения включает в этот регистр включает защиту записи. Предполагается, что это операций перезагрузки (запись 0xAAAA).

События обновления предделителя и перезагрузки счётчика показываются в регистре состояния.

21.3.3. Режим отладки

В режиме отладки (ядро Cortex-M3 остановлено) счётчик IWDG продолжает работать или останавливается в зависимости от бита конфигурации DBG_IWDG_STOP в модуле DBG.

Рис. 213. Блок-схема IWDG.



NB: Функции сторожа, реализованные в домене V_{DD}, продолжают работать в режимах Stop и Standby.

Таблица 107. Min/max таймауты IWDG (в ms) на 40 kHz (LSI)(1).

Предделитель	Биты PR[2:0]	Min таймаут RL[11:0]=0x000	Max таймаут RL[11:0]=0xFFF
/4	0	0.1	409.6
/8	1	0.2	819.2
/16	2	0.4	1638.4
/32	3	0.8	3276.8
/64	4	1.6	6553.6
/128	5	3.2	13107.2
/256	6 (или 7)	6.4	26214.4

1. Частоту RC генератора LSI можно менять.

21.4. Регистры IWDG

Регистры доступны полусловами или словами.

21.4.1. Регистр ключа (IWDG_KR)

Смещение адреса: 0x00

По сбросу: 0x0000 0000. Сбрасывается режимом Standby.

31–16	15–0
Reserved	KEY[15:0]
	w w w w w w w w w w w w w w w w

— Биты 31:16 Резерв, не трогать.

— Биты 15:0 **KEY[15:0]**: Значение ключа (только запись, чтение 0000h)

При записи AAAAh включается регулярный интервал, иначе при достижении 0 выдаётся сброс.

При записи 5555h разрешается доступ к регистрам IWDG_PR и IWDG_RLR.

При записи CCCCh запускает таймер (если не включён аппаратный сторож).

21.4.2. Регистр предделителя (IWDG_PR)

Смещение адреса: 0x04

По сбросу: 0x0000 0000.

31–3	2–0
Reserved	PR[2:0]
	rw rw rw

— Биты 31:3 Резерв, не трогать.

— Биты 2:0 **PR[2:0]:** Значение предделителя

Биты защищены от записи. При записи бит PVU в регистре IWDG_SR должен быть снят.

000: делитель /4

001: делитель /8

010: делитель /16

011: делитель /32

100: делитель /64

101: делитель /128

110: делитель /256

111: делитель /256

NB: Чтение этого регистра возвращает значение предделителя из домена V_{DD}. При незавершённой операции записи значение недостоверно. Читать надо при снятом бите PVU в регистре IWDG_SR.

21.4.3. Регистр перезагрузки (IWDG_RLR)

Смещение адреса: 0x08

По сбросу: 0x0000 0FFF. Сбрасывается режимом Standby.

31–12	11–0
Reserved	RL[11:0]
	rw rw rw rw rw rw rw rw rw rw rw rw

— Биты 31:12 Резерв, не трогать.

— Биты 11:0 **RL[11:0]:** Перегружаемое значение сторожа

Биты защищены от записи. Пишется в счётчик каждый раз после записи AAAAh в регистр IWDG_KR. Обратный отсчёт начинается с этого значения. Период таймаута это функция этого значения и предделителя тактов. При записи бит RVU в регистре IWDG_SR должен быть снят.

NB: Чтение этого регистра возвращает значение перезагрузки из домена V_{DD}. При незавершённой операции записи значение недостоверно. Читать надо при снятом бите PVU в регистре IWDG_SR.

21.4.4. Регистр состояния (IWDG_RLR)

Смещение адреса: 0x0C

По сбросу: 0x0000 0000. Режимом Standby не сбрасывается.

31–2	1	0
Reserved	RVU	PVU
	r	r

— Биты 31:2 Резерв, не трогать.

— Бит 1 **RVU:** Обновление значения перезагрузки

Ставится аппаратно во время операции записи регистра, снимается аппаратно после её завершения в домене V_{DD} (занимает до 5 циклов RC 40 kHz). Писать можно при снятом бите RVU.

— Бит 0 **PVU:** Обновление значения предделителя

Ставится аппаратно во время операции записи регистра, снимается аппаратно после её завершения в домене V_{DD} (занимает до 5 циклов RC 40 kHz). Писать можно при снятом бите PVU.

NB: При использовании нескольких значений перезагрузки или предделителя перед их записью нужно дождаться снятия битов RVU (перезагрузка) или PVU (предделитель). А завершения операции записи ждать не надо, она завершится без вас.

21.4.5. Карта регистров IWDG

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**IWDG_KR**	Reserved																KEY[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**IWDG_PR**	Reserved																													PR[2:0]		
	Reset value																														0	0	0
0x08	**IWDG_RLR**	Reserved																RL[11:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0C	**IWDG_SR**	Reserved																														RVU	PVU
	Reset value																															0	0

22. Оконный сторожевой таймер (WWDG)

22.1. Введение в WWDG

WWDG используется для обнаружения программных отказов, обычно вызываемых внешним влиянием или непредвиденными логическими условиями. По истечении заданного периода времени схема выдаёт сброс MCU, если только программа не обновляет обратный счётчик до очистки бита `T6`. Сброс MCU также выдаётся если 7-бит обратный счётчик (в регистре управления) обновляется до достижения счётчиком границы окна. То есть счётчик нужно обновлять в границах окна.

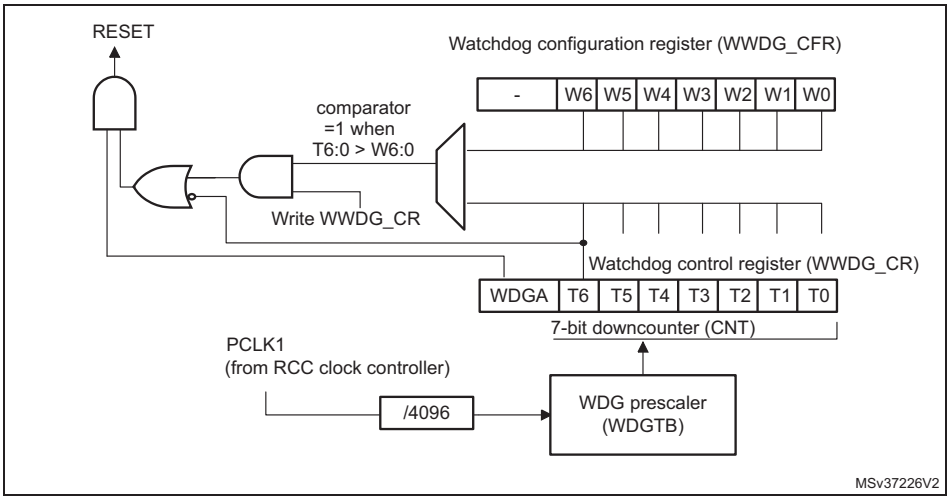
22.2. Основные свойства WWDG

- Программируемый независимый обратный счётчик
- Условный сброс (при активированном стороже)
  - Если значение счётчика становится меньше 0x40
  - Если счётчик перегружается вне окна
- Прерывание раннего пробуждения (EWI): выдаётся (если разрешено при активированном стороже) когда счётчик равен 0x40.

22.3. Функциональное описание WWDG

Сброс выдаётся активированным сторожем (стоит бит `WDGA` в регистре `WWDG_CR`) когда 7-бит счётчик (биты `T[6:0]`) переходят с 0x40 на 0x3F (`T6` очищается). Также он выдаётся при перезаписи счётчика когда он больше границы окна.

Рис. 214. Блок-схема сторожа.



Во избежание сброса MCU программа должна регулярно писать в работающий счётчик WWDG_CR число в пределах от 0xFF и 0xC0. Писать нужно только когда значение счётчика меньше границы окна.

Включение сторожа

По сбросу сторож всегда выключен. Включается он установкой бита WDGA в регистре WWDG_CR, выключается только сбросом.

Управление счётчиком

Счётчик работает даже при выключенном стороже. При включённом стороже бит T6 должен стоять во избежание немедленного сброса.

Биты T[5:0] содержат число шагов счётчика до выдачи сброса. Из-за неизвестного состояния предделителя на момент записи регистра WWDG_CR время лежит в диапазоне между минимальным и максимальным значениями. Регистр конфигурации (WWDG_CFR) содержит верхний предел окна: для предупреждения выдачи сброса перегружать счётчик надо когда его значение меньше границы окна и больше 0x3F.

NB: Для выдачи программного сброса можно использовать бит T6 (бит WDGA стоит, а T6 чист).

Прерывание сторожа

В случае надобности выполнения каких-либо действий перед реальным сбросом можно использовать Прерывание Ранней Побудки (EWI). Оно разрешается битом EWI в регистре WWDG_CFR. и вызывается при достижении счётчиком значения 0x40.

Прерывание EWI можно использовать для выполнения проверки и постепенного выключения системы без выдачи сброса WWDG. В этом случае обработчик должен перегрузить счётчик WWDG и запустить нужные действия.

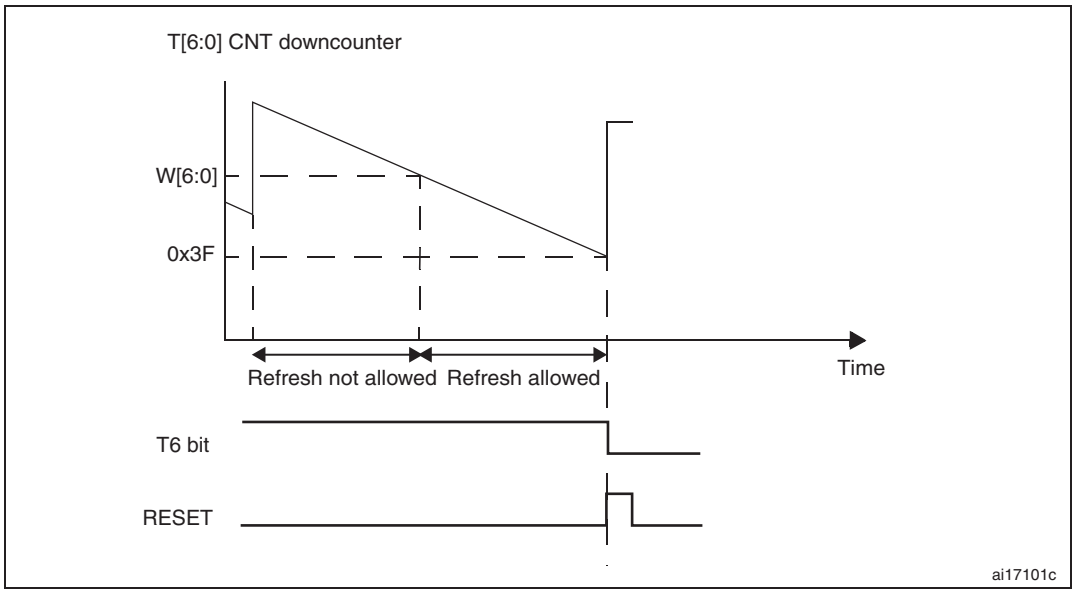
Прерывание EWI очищается снятием бита EWIF в регистре WWDG_SR.

NB: Если прерывание EWI не может быть по каким-либо причинам обслужено, то сброс WWDG таки произойдёт.

22.4. Вычисление таймаута WWDG

Внимание: При записи регистра WWDG_CR всегда ставьте бит T6.

Рис. 215. Временная диаграмма Оконного сторожа.



Формула вычисления таймаута:

$$t_{WWDG} = t_{PCLK1} \times 4096 \times 2^{WDGTB[1:0]} \times (T[5:0] + 1) \qquad (ms)$$

где:

t_{WWDG}: таймаут WWDG

t_{PCLK1}: период тактов APB1 в ms

4096: значение внутреннего делителя

Например, частота APB1 равна 24 MHz, WDGTB[1:0] равны 3 и T[5:0] равны 63:

$$t_{WWDG} = 1/\ 24000 \times 4096 \times 2^{3} \times (63+1) = 21.85ms$$

Таблица 109. Min и Max значения t_{WWDG} при f_{PCLK1}=36MHz.

Предделитель	WDGTB	Min	Max
1	0	113 µs	7.28 ms
2	1	227 µs	14.56 ms
4	2	455 µs	29.12 ms
8	3	910 µs	58.25 ms

22.5. Режим отладки

В режиме отладки (ядро Cortex-M3 остановлено) счётчик WWDG продолжает работать или останавливается в зависимости от бита конфигурации DBG_WWDG_STOP в модуле DBG.

22.6. Регистры WWDG

22.6.1. Регистр ключа (WWDG_CR)

Смещение адреса: 0x00

По сбросу: 0x0000 007F.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								WDGA	T[6:0]						
								rs	rw						

— Биты 31:8 Резерв, не трогать.

— Бит 7 **WDGA:** Бит активации

Ставится программно, снимается аппаратно по сбросу. При WDGA = 1 может выдаваться сброс.

0: Выключен

1: Включён

— Биты 6:0 **T[6:0]:** 7-бит счётчик (MSB to LSB)

Значение счётчика. Декрементируется каждые (4096 x $2^{WDGTB[1:0]}$) тактов PCLK1. Сброс выдаётся при переходе от 0x40 к 0x3F (T6 очищается).

22.6.2. Регистр конфигурации (WWDG_CFR)

Смещение адреса: 0x04

По сбросу: 0x0000 007F.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						EWI	WDGTB[1:0]		W[6:0]						
						rs	rw		rw						

— Биты 31:10 Резерв, не трогать.

— Бит 9 **EWI:** Прерывание Ранней Побудки

Если установлен, то при достижении счётчиком 0x40 выдаётся прерывание. Чистится аппаратно после сброса.

— Биты 8:7 **T[6:0]:** База таймера

Изменение базы предделителя:

00: CK Counter Clock (PCLK1 / 4096) / 1

01: CK Counter Clock (PCLK1 / 4096) / 2

10: CK Counter Clock (PCLK1 / 4096) / 4

11: CK Counter Clock (PCLK1 / 4096) / 8 .

— Биты 6:0 **W[6:0]:** 7-бит значение окна чтобы сравнивать со счётчиком.

22.6.3. Регистр состояния (WWDG_SR)

Смещение адреса: 0x08

По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															EWIF
															rc_w0

— Биты 31:10 Резерв, не трогать.

— Бит 0 **EWIF:** Флаг Прерывания Ранней Побудки

Ставится аппаратно при достижении счётчиком 0x40. Чистится программно записью ‘0’. Запись ‘1’ бестолкова. Ставится также при не разрешённом прерывании.

22.6.4. Карта регистров WWDG

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	WWDG_CR	Reserved																								WDGA	T[6:0]						
	Reset value																									0	1	1	1	1	1	1	1
0x04	WWDG_CFR	Reserved																						EWI	WDGTB1	WDGTB0	W[6:0]						
	Reset value																							0	0	0	1	1	1	1	1	1	1
0x08	WWDG_SR	Reserved																															EWIF
	Reset value																																0

23. Криптографический процессор (CRYP)

23.1. Введение в CRYP

Криптографический процессор может шифровать и дешифровать данные по алгоритмам DES, TDES и AES (128, 192, или 256). Он полностью совместим с промышленными стандартами.

Алгоритмы DES и TDES могут работать в режиме *Простой замены* (ECB) или *Блок-чейн* (CBC).

Периферия CRYP это 32-bit AHB2 устройство. Оно поддерживает передачи DMA входных и выходных данных и два FIFO (по 8 слов).

23.2. Основные свойства CRYP

- Годится для операций AES, DES и TDES
- AES
  - Поддерживает алгориты ECB, CBC, CTR, CCM и GCM (CCM и GCM только на STM32F42xxx и STM32F43xxx)
  - Поддерживает 128-, 192- и 256-бит ключи
  - 4 × 32-бит векторы (IV) в режимах CBC, CTR, CCM и GCM

Table 111. Число циклов обработки 128-бит блока (STM32F415/417xx)

Алгоритм / Размер ключа	ECB	CBC	CTR
128b	14	14	14
192b	16	16	16
256b	18	18	18

Table 112. Число циклов обработки 128-бит блока (STM32F43xxx)

Algorithm / Key size	ECB	CBC	CTR	GCM				CCM			
				Init	Header	Payload	Tag	Init	Header	Payload	Tag
128b	14	14	14	24	10	14	14	12	14	25	14
192b	16	16	16	28	10	16	16	14	16	29	16
256b	18	18	18	32	10	18	18	16	18	33	18

- DES/TDES
  - Прямая имплементация простого алгоритма DES (используется один ключ, K1)
  - Поддерживает цепочные алгоритмы ECB и CBC
  - Поддерживает 64-, 128- и 192-бит ключи (включая чётность)
  - 2 × 32-bit векторы инициализации (IV) в режиме CBC
  - 16 тактов HCLK обработки одного 64-бит блока DES
  - 48 тактов HCLK обработки одного 64-бит блока TDES
- Общее для DES/TDES и AES
  - IN и OUT FIFO (по 8 слов, 32-бит ширины, для 4 блоков DES или 2 блоков AES)
  - 2 канала DMA (входной и выходной)
  - Логика обмена данных для поддержки 1-, 8-, 16- и 32-бит данных

23.3. Функциональное описание CRYP

Есть два криптографических ядра, TDES с поддержкой DES и AES. Алгоритмы TDES и AES используют блоковую шифрацию, незавершённые блоки надо программно расширять и удалять этот хвост при дешифрации. Аппаратура этого не делает.

Рис. 216. Блок-схема (STM32F415/417xx)

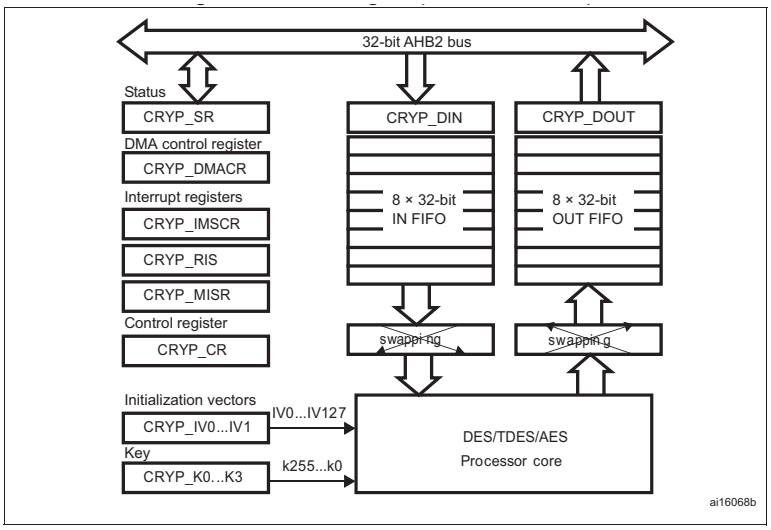
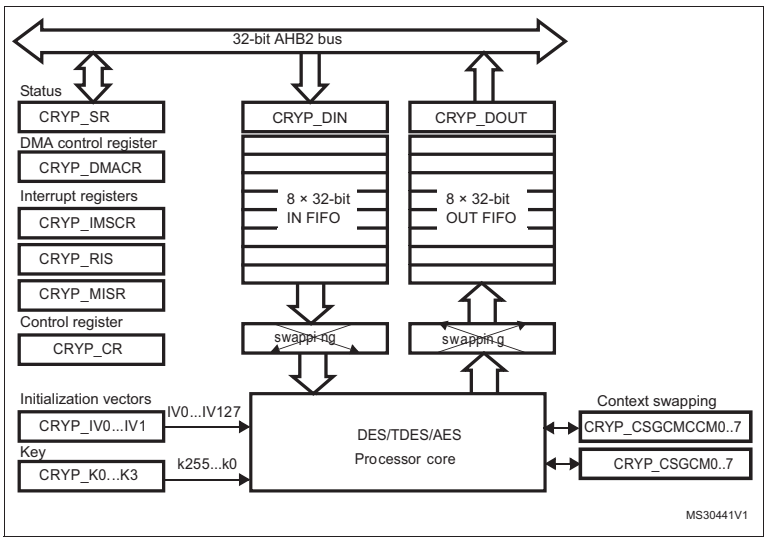


Рис. 217. Блок-схема (STM32F43xxx)



23.3.1. Криптографическое ядро DES/TDES

Состоит из трёх компонентов:

- Алгоритм DES — (DEA)
- Множественные ключи (1 для алгоритма DES, от 1 до 3 для алгоритма TDES)
- Вектор инициалиации (в режиме CBC)

Основная работа TDES: входной блок читается в DEA и кодируется первым ключом, K1 (K0 не используется в TDES). Результат декодируется вторым ключом, K2, и снова кодируется третьим ключом, K3. Ключ зависит от алгоритма:

- DES: Ключ = [K1]
- TDES: Ключ = [K3 K2 K1]

где Kx=[KxR KxL], R = правый, L = левый

Результирующий блок используется для вычисления шифрованного текста.

Результат промежуточных стадий DEA за границы криптографии не выходит.

TDES имеет три опции ключей:

- Три независимых ключа. K1, K2 и K3.
- Два независимых ключа. K1 и K2 независимые, K3 = K1.
- Три равных ключа (это одинарный DES).

Режим прямой замены DES и TDES (DES/TDES-ECB)

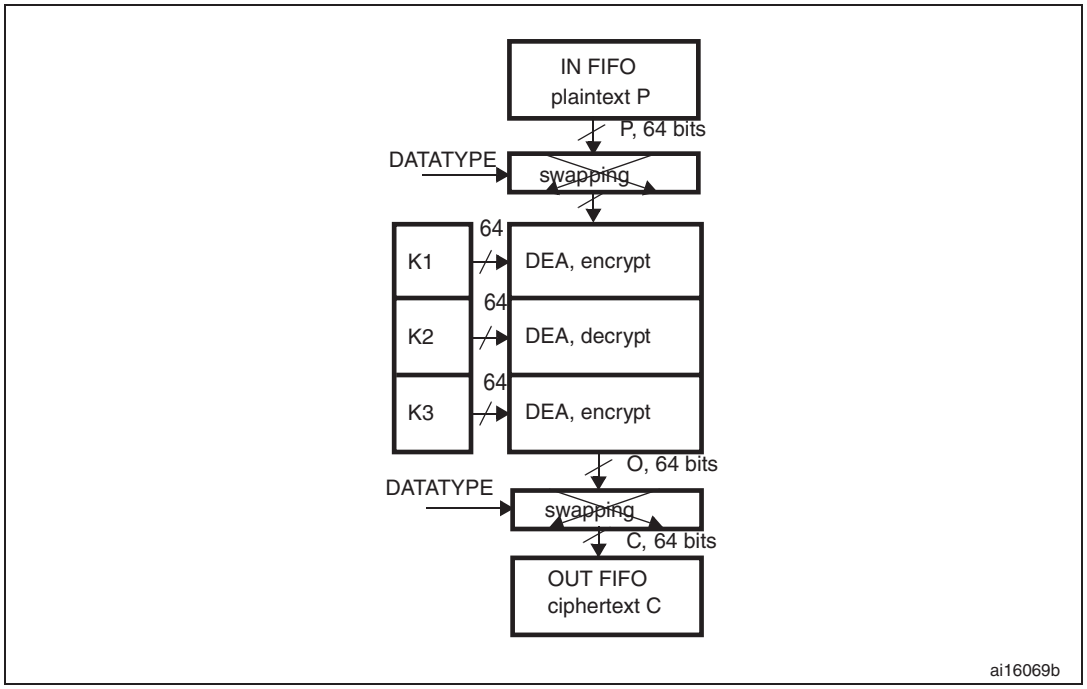
- DES/TDES-ECB шифрация

  64-бит блок простого текста (P) после обмена бит/байт/полуслово (см. секцию 23.3.3) становится входным блоком (I). Он через DEA кодируется ключом K1. Результат подаётся на вход DEA, где декодируется ключом K2. Результат подаётся на вход DEA, где кодируется ключом K3. Полученный 64- бит выходной блок (O) после обмена бит/байт/полуслово пихается в OUT FIFO.

- DES/TDES-ECB дешифрация

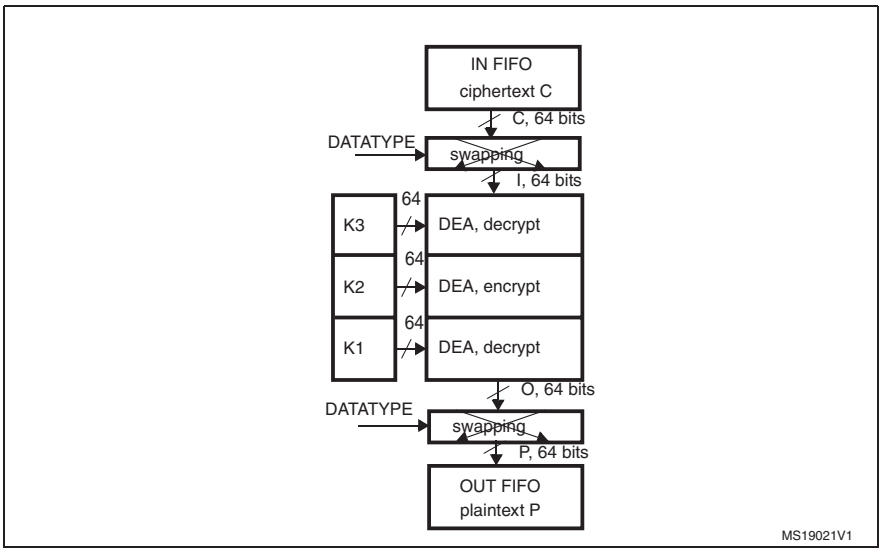
  64-бит блок шифрованного текста (C) после обмена бит/байт/полуслово становится входным блоком (I). Он через DEA декодируется ключом K3. Результат подаётся на вход DEA, где кодируется ключом K2. Результат подаётся на вход DEA, где декодируется ключом K1. Полученный 64- бит выходной блок (O) после обмена бит/байт/полуслово становится простым текстом (P).

Рис. 218. Кодирование DES/TDES-ECB



1. K: key; C: cipher text; I: input block; O: output block; P: plain text.

Рис. 219. Декодирование DES/TDES-ECB



1. K: key; C: cipher text; I: input block; O: output block; P: plain text.

Режим блок-чейн DES и TDES (DES/TDES-CBC)

- DES/TDES-CBC шифрация

  Сначала простой текст делится на 64-бит блоки данных. Первый блок простого текста (P1) после обмена бит/байт/полуслово (см. секцию 23.3.3) и XOR с вектором инициализации IV становится входным блоком (I1) (I1 = IV ⊕ P1). Он через DEA кодируется ключом K1. Результат подаётся на вход DEA, где декодируется ключом K2. Результат подаётся на вход DEA, где кодируется ключом K3. Полученный 64-бит выходной блок (O1) становится первым шифрованным (C1). После его XOR с вторым блоком простого текста (P2) получается второй входной блок (I2) = (C1 ⊕ P2). Он через TDEA становится вторым шифрованным (C2). Цепочка повторяется вплоть до обработки последнего блока простого текста. Если весь простой текст не состоит из целого числа блоков, то последнюю часть надо шифровать определённым для приложения способом.

- DES/TDES-CBC дешифрация

  Первый шифрованный блок (C1) напрямую становится входным (I1). Он декодируется через K3, результат кодируется через K2 и полученное декодируется через K1. Итог (O1) после XOR с IV выдаёт первый блок простого текста (P1 = O1 ⊕ IV). Второй шифрованный блок (C2), в качестве (I2), обрабатывается через TDEA. Результат (O2) после XOR с первым шифрованным блоком даёт второй блок простого текста (P2 = O2 ⊕ C1). Цепочка продолжается до конца шифрованного текста. Последний неполный шифрованный блок обрабатывается определённым для приложения способом.

Рис. 220. Кодирование DES/TDES-CBC

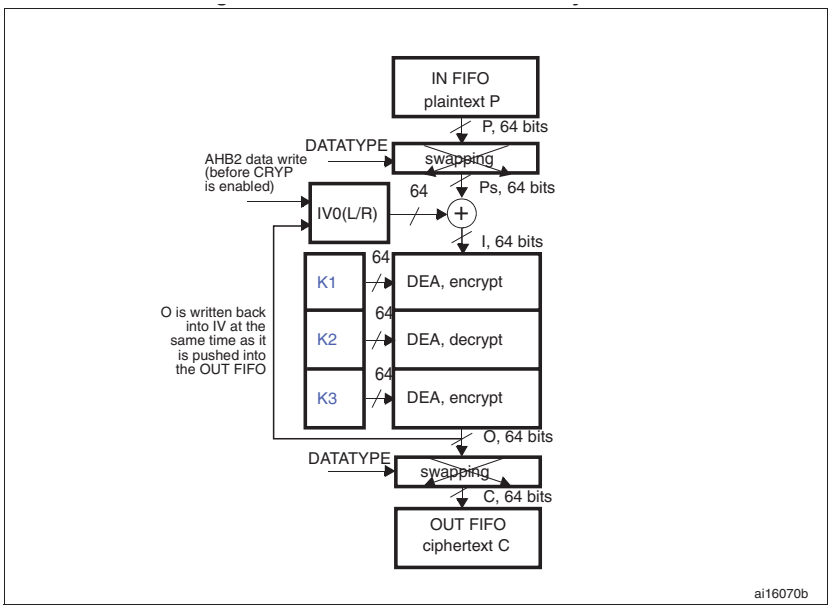
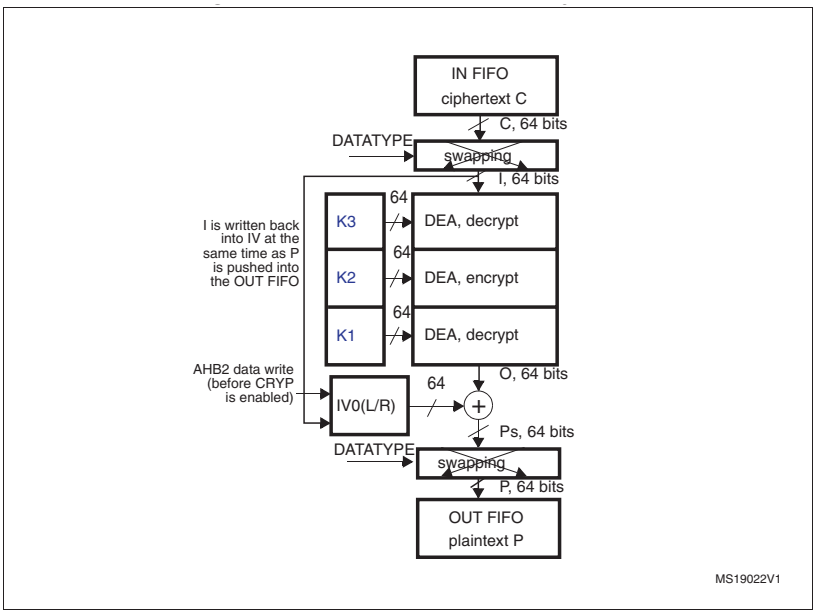


Рис. 220. Декодирование DES/TDES-CBC



23.3.2. Криптографическое ядро AES

Состоит из трёх компонентов:

- Алгоритм AES (AEA)
- Множественных ключей
- Вектора/(.ов) инициализации или "Анонса"

Есть три длины ключей AES: 128, 192 и 256 бит и, в зависимости от режима, ноль или один 128-бит вектор инициализации (IV).

Основная работа AES: входной блок читается из FIFO в AEA и кодируется ключом (K0...3).

Формат ключа зависит от его длины:

- При Keysize=128: Key=[K3K2]
- При Keysize=192: Key=[K3K2K1]
- При Keysize=256: Key=[K3K2K1K0]
  где Kx=[KxR KxL], R = правый, L = левый

Режим прямой замены AES (AES-ECB)

- AES-ECB шифрация
  128-бит блок простого текста (P) после обмена бит/байт/полуслово (см. секцию 23.3.3) становится входным блоком (I). Он через AEA кодируется 128, 192 или 256-бит ключом. Полученный 128- бит выходной блок (O) после обмена бит/байт/полуслово пихается в OUT FIFO как шифрованный текст (C).
- AES-ECB дешифрация
  Сначала надо подготовить секретный ключ собрав кольцевой ключ, который будет использован первым при дешифрации. Это делается ядром AES. См. Секцию 23.3.6.
  128-бит блок шифрованного текста (C) после обмена бит/байт/полуслово становится входным блоком (I). Процедура дешифрации обратна шифрации. Полученный 128- бит выходной блок (O) после обмена бит/байт/полуслово становится простым текстом (P).

Рис. 222. Кодирование AES-ECB

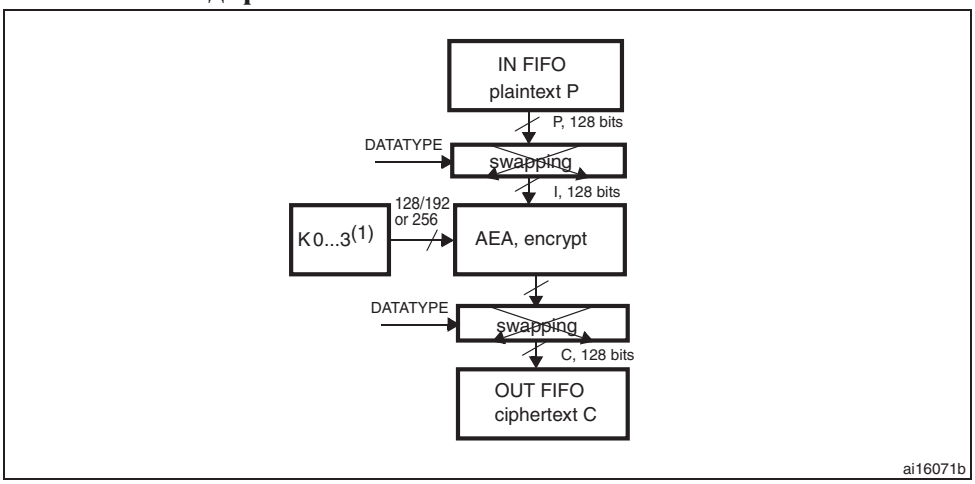
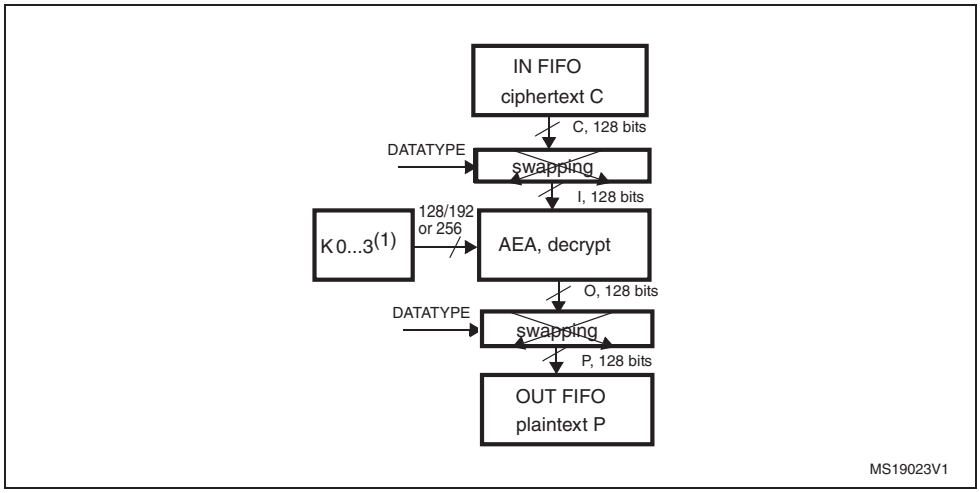


Рис. 223. Декодирование AES-ECB



Режим AES блок-чейн (AES-CBC)

- AES-CBC шифрация

  Первый входной блок (I1) после обмена бит/байт/полуслово (см. Секцию 23.3.3) получают с помощью XOR первого блока простого текста (P1) с 128-бит вектором инициализации IV (I1 = IV ⊕ P1). Он через AEA кодируется 128-, 192- или 256-бит ключом (K0...K3). Полученный 128-бит выходной блок (O1) и есть кодированный (C1), то есть, C1 = O1. Его XOR с вторым простым блоком даёт второй входной блок, (I2) = (C1 ⊕ P2). Второй входной блок после обработки AEA даёт второй кодированный. И цепочка продолжается до последнего блока. Последний неполный блок обрабатывается определённым для приложения способом. Создание секретного ключа описано в Секции 23.3.6.

- AES-CBC дешифрация

  Первый кодированный блок (C1) и есть входной (I1). Он через AEA декодируется 128-, 192- или 256-бит ключом (K0...K3). Полученный 128-бит выходной блок (O1) после XOR с 128-бит вектором инициализации IV (тем же самым, что и при кодировании) даёт первый блок простого текста (P1 = O1 ⊕ IV). Второй кодированный блок обрабатывается через AEA. Полученный 128-бит выходной блок (O2) после XOR с первым кодированным даёт второй блок простого текста (P2 = O2 ⊕ C1). И цепочка продолжается до последнего блока. Последний неполный шифрованный блок обрабатывается определённым для приложения способом.

Рис. 224. Кодирование AES-CBC

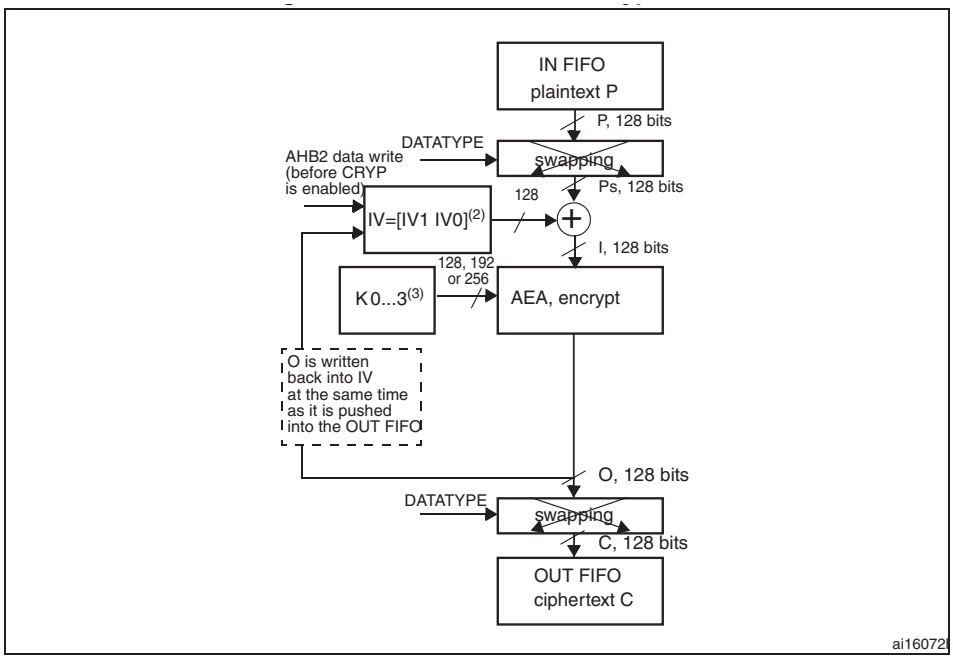
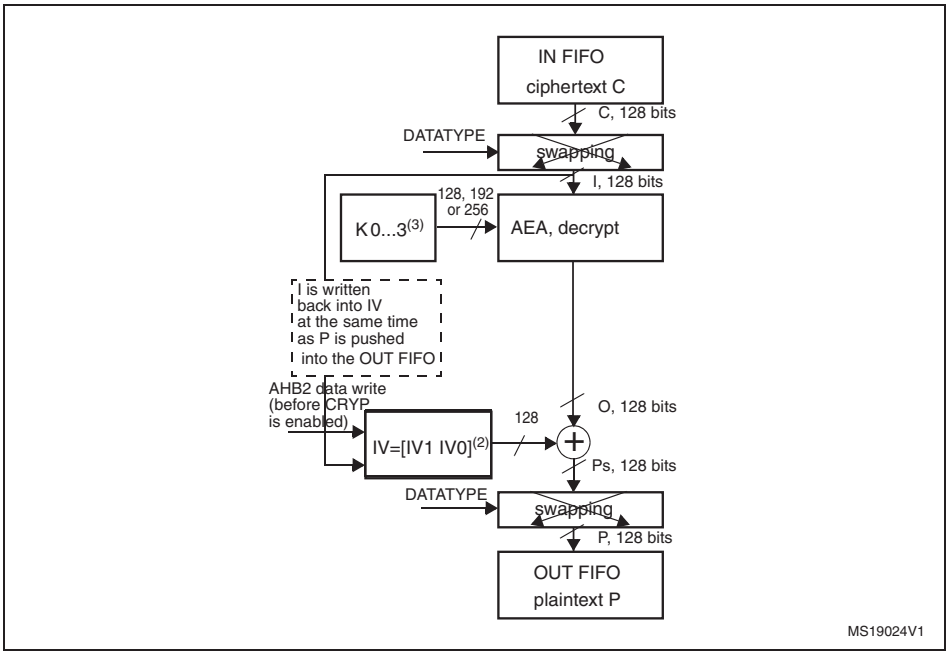


Рис. 225. Декодирование AES-CBC



1. K: ключ; C: кодированный текст; I: входной блок; O: выходной блок; Ps: простой текст перед обменом при декодировании или после обмена после при кодировании; P: простой текст; IV: векторы инициализации.
2. IVx=[IVxR IVxL], R=правый, L=левый.
3. Если Key size=128=>Key=[K3K2].
   Если Key size = 192 => Key = [K3 K2 K1]
   Если Key size = 256 => Key = [K3 K2 K1 K0].

Режим счётчика AES (AES-CTR)

Блок AES используется как генератор потока ключа. Этот поток через XOR с простым текстом даёт шифрованный текст. Операция симметричная и кодирование равно декодированию.

Дано:

- Простой текст: P[0], P[1], ..., P[n] (128 бит каждый)
- Ключ K (размер совсем безразличен)
- Начальный блок счётчика (так сказать, ICB, хоть и работает также как и IV при CBC)

Кодированный текст вычисляется так:

```
C[i] = enck(iv[i]) ⊕ P[i],
```

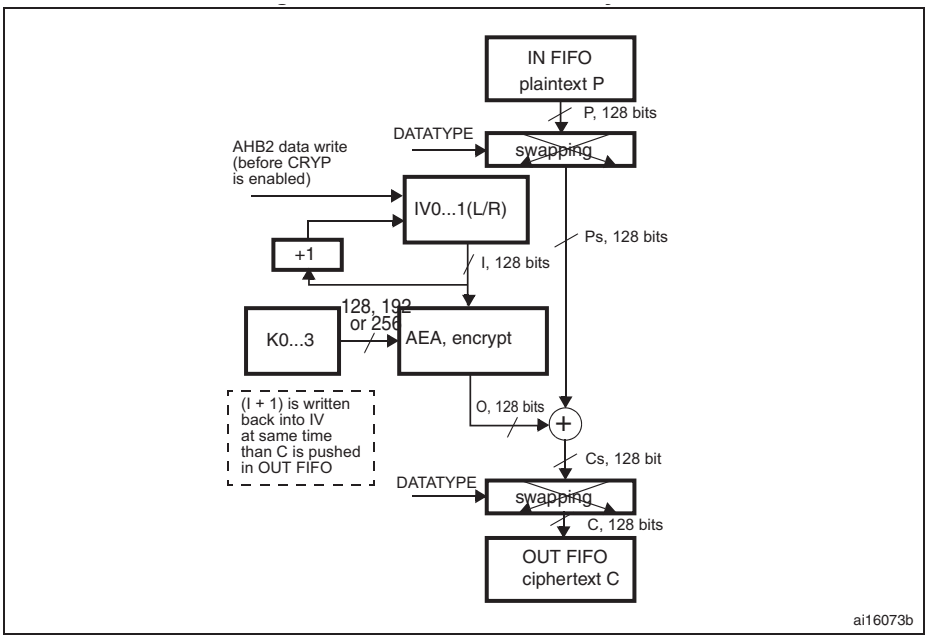
где:

`iv[0]` = ICB и `iv[i+1] = func(iv[i])`,

где `func` это функция обновления для предыдущего блока `iv`; `func` это в основном инкремент одного из полей блока `iv`.

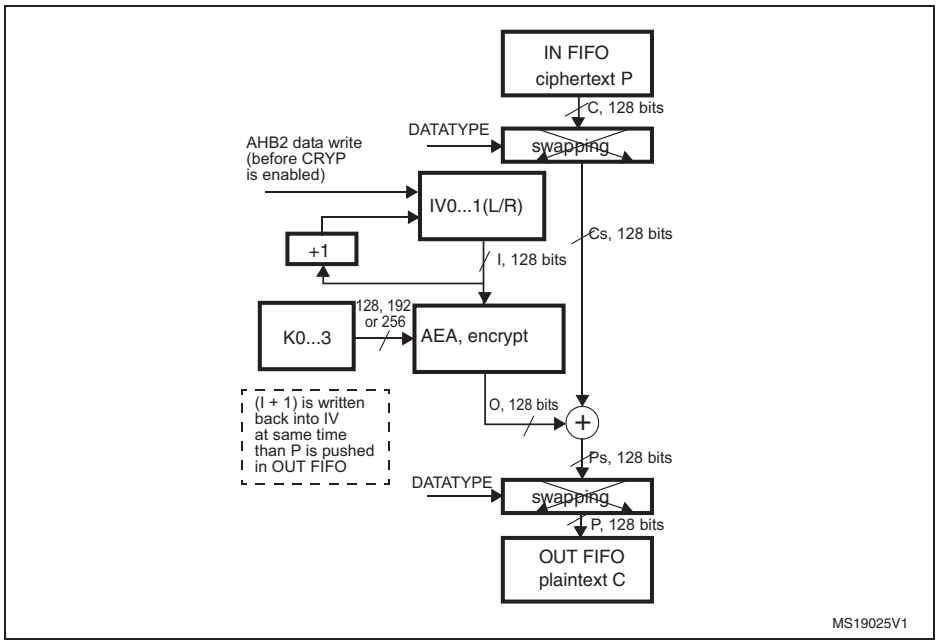
При одинаковом ICB для кодирования и декодирования всё становится одинаковым.

Рис. 226. Кодирование AES-CTR



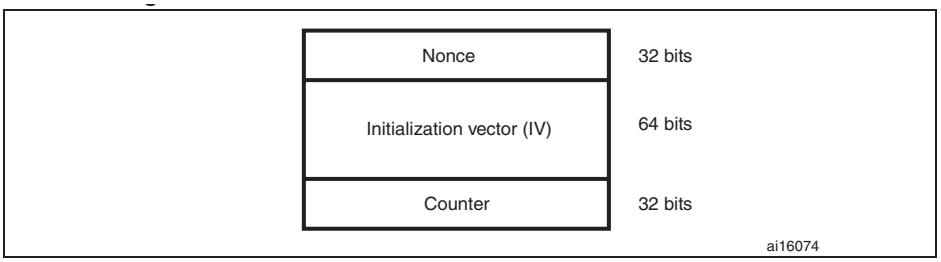
1. K: ключ; C: шифрованный текст; I: входной блок; o: выходной блок; Ps: простой текст перед обменом (при декодировании) или после него (при кодировании); Cs: шифрованный после обмена (при декодировании) или перед ним ((при кодировании)); P: простой текст; IV: Векторы инициализации.

Рис. 227. Декодирование AES-CTR



1. K: ключ; C: шифрованный текст; I: входной блок; o: выходной блок; Ps: простой текст перед обменом (при декодировании) или после него (при кодировании); Cs: шифрованный после обмена (при декодировании) или перед ним ((при кодировании)); P: простой текст; IV: Векторы инициализации.

Рис. 228. Структура начального блока счётчика AES-CTR



- **Nonce** ("Анонс") это 32-бит одноразовое число. Для каждой связи надо назначать новое.
- Вектор инициализации (IV) это 64-бит число. Стандарт требует, чтобы оно использовалось в данном ключе только один раз
- 32-бит счётчик в формате *big-endian*, с инкрементом по каждому блоку, начиная с '1'.

Блок инкрементирует только младшие 32 бита, остальные 96 бит не меняются.

Режим Галуа/счётчика AES (GCM)

Выполняет кодирование и аутентификацию простого текста и выдаёт кодированный текст и тег (код аутентификации или проверки целостности сообщения). Основан на режиме счётчика AES. Для генерации тега используется множитель для фиксированного ограниченного поля. В начале алгоритма нужен вектор инициализации.

Обрабатываемое сообщение делится на 2 части:

- Заголовок (дополнительные данные аутентификации): незащищённые данные с аутентификацией (вроде маршрутизации пакета)
- Нагрузка (простой или кодированный текст): само кодированное сообщение с аутентификацией.

Заголовок идёт первым и не смешивается с текстом.

Стандарт GCM требует передачи в конце сообщения особого 128-бит блока с размерами заголовка (64 бита) и нагрузки (64 бита). При вычислениях блоки заголовков надо отличать от блоков нагрузки.

В режиме GCM для кодирования/декодирования нужны четыре шага:

1. Фаза инициализации GCM
   Здесь вычисляется и сохраняется ключ HASH для обработки всех блоков.
   Последовательность:
   a) Выключаем криптопроцессор снятием бита CRYPEN в регистре CRYP_CR.
   b) Выбираем цепочный режим GCM записью '01000' в биты ALGOMODE регистра CRYP_CR.
   c) Пускаем фазу GCM Init записью '00'в биты GCM_CCMPH регистра CRYP_CR.
   d) Инициализируем регистры ключа (128,192 и 256 бит) в CRYP_KEYRx, равно как и вектор инициализации (IV).
   e) Установкой бита CRYPEN в '1' пускаем вычисление ключа HASH.
   f) Ждём снятия бита CRYPEN.
   g) Ставим бит CRYPEN в '1'.
2. Фаза заголовка GCM
   Последовательность:
   h) Пускаем фазу заголовка записью '01' в биты GCM_CCMPH регистра CRYP_CR.
   i) Пишем данные заголовка. Есть три метода:
      – Блоками по 32 бита пишем данные в регистр CRYP_DIN, флаг IFNF показывает готовность входного FIFO принять данные. Размер заголовка должен быть кратен 128 бит (4 слова).
      – Блоками по 8 слов пишем данные в регистр CRYP_DIN, флаг IFEM показывает готовность входного FIFO принять данные (IFEM='1'). Размер заголовка должен быть кратен 128 бит (4 слова).
      – Используем DMA.
   j) После передачи всех данных заголовка ждём снятия бита BUSY в регистре CRYP_SR.
3. Фаза нагрузки GCM (кодирование/декодирование)
   Последовательность:
   k) Пишем '10' в биты GCM_CCMPH регистра CRYP_CR.
   l) Битом ALGODIR в CRYP_SR выбираем направление (кодирование или декодирование).
   m) Данные нагрузки пишем в регистр CRYP_DIN, флаг IFNF показывает готовность входного FIFO принять данные. В регистр CRYP_DIN писать блоками по 8 слов, флаг IFEM показывает готовность входного FIFO принять данные (IFEM='1'). Одновременно можно отслеживать флаг OFNE/OFFU регистра CRYP_DOUT по теме пустоты выходного FIFO.
   n) Повторяем предыдущий шаг до конца обработки всех блоков нагрузки. А можно использовать и DMA.

4. Финальная фаза GCM
   Последовательность:
   o) Пишем ’11’ в биты GCM_CCMPH[1:0] регистра CRYP_CR.
   p) В регистр CRYP_DIN 4 раза пишем объединённое число битов в заголовке (64 бита) и число битов нагрузки (64 бита).
   q) Ждём установки флага OFNE (FIFO не пуст) в регистре CRYP_SR.
   r) Читаем регистр CRYP_DOUT четыре раза: это тег аутентификации.
   s) Выключаем криптопроцессор (бит CRYPEN в CRYP_CR = ‘0’)

NB: По концу декодирования ключ заново вычислять не надо, сгенерированный тег должен совпадать с переданным в самом сообщении. Кроме того, бит ALGODIR (направление) должен стоять.

При переходе из фазы заголовка в фазу тега криптопроцессор включать/выключать не надо.

Код аутентификации сообщения AES Галуа (GMAC)

Это поддерживается для аутентификации простого текста. Тег генерируется алгоритмом GCM и множителем с фиксированным ограниченным полем.

В начале алгоритма нужен вектор инициализации.

Алгоритм GMAC это тот же GCM , но только с заголовком, фаза нагрузки не нужна.

Комбинированная шифромашина AES (CCM)

CCM это кодирование и аутентификация простого текста с выдачей кода и тега (код аутентификации или целостности сообщения). Основано на режиме счётчика AES. Для генерации 128-бит тега используется режим AES CBC.

Стандарт CCM (RFC 3610 Счётчик с CBC-MAC (CCM) от Сентября 2003) определяет некоторые правила кодирования для первого блока аутентификации (блок B0). В частности, первый блок включает флаги, анонс и длину нагрузки в байтах. Для кодирования/декодирования стандарт CCM определяет другой формат, именуемый A или счётчик. В фазе нагрузки счётчик инкрементируется и при генерации тега его младшие 32 бита заполняются ‘1’ (пакет A0 в стандарте CCM).

NB: Пакет B0 аппаратно не формируется, это дело программы.

Как и в алгоритме GCM, сообщение разбивается на 2 части:

- Заголовок (дополнительная аутентификация): незащищённые аутентифицированные данные (вроде маршрута пакета)
- Нагрузка (простой или шифрованный текст): защищённое аутентифицированное сообщение.

NB: Заголовок идёт первым и части не смешиваются.

При кодировании/декодировании в режиме CCM нужны 4 шага:

1. Фаза инициализации CCM
   Пакет B0 сообщения CCM (1й пакет) пишем в регистр CRYP_DIN, регистр CRYP_DOUT не нужен.

Последовательность:

   a) Выключаем криптопроцессор очисткой бита CRYPEN в регистре CRYP_CR.
   b) Выбираем режим CCM записью ‘01001’ в биты ALGOMODE регистра CRYP_CR.
   c) Пускаем фазу инициализации записью ‘00’ в биты GCM_CCMPH регистра CRYP_CR.
   d) Пишем регистры ключа (128,192 и 256 бит) в CRYP_KEYRx и вектор инициализации (IV).
   e) Ставим бит CRYPEN регистра CRYP_CR.
   f) Пишем пакет B0 во входной регистр данных.
   g) Ждём очистки бита CRYPEN.
   h) Ставим бит CRYPEN в ‘1’.

2. Фаза заголовка CCM
   Для кодирования и декодирования последовательность идентична. Регистр CRYP_DOUT не нужен. Если дополнительной аутентификации нет, то фазу пропускаю.

Последовательность:

   i) Пускаем фазу заголовка записью ‘01’ в биты GCM_CCMPH регистра CRYP_CR.

j) Есть три метода записи заголовка:

- Блоками по 32 бита в регистр CRYP_DIN, флаг IFNF показывает готовность FIFO принять данные. Размер заголовка должен быть кратен 128 битам (4 слова).
- Блоками по 8 слов в регистр CRYP_DIN, флаг IFEM показывает готовность FIFO принять данные (IFEM ='1'). Размер заголовка должен быть кратен 128 битам (4 слова)
- Использовать DMA.

NB: Первый блок B1 программно форматируют по длине заголовка.

k) По завершению записи заголовка ждём очистки флага BUSY.

3. Фаза нагрузки CCM (кодирование/декодирование)
   Полученные данные появляются в регистре CRYP_DOUT.

Последовательность:

l) Пишем '10' в биты GCM_CCMPH регистра CRYP_CR.

m) Битом ALGODIR в CRYP_CR выбираем направление (кодирование/декодирование).

n) Пишем сообщение в регистр CRYP_DIN, флаг IFNF показывает готовность FIFO принять данные. По другому можно писать блоками по 8 слов в регистр CRYP_DIN, флаг IFEM показывает готовность FIFO принять данные (IFEM ='1'). Параллельно, флаг OFNE/OFFU регистра CRYP_DOUT показывает пустоту выходного FIFO.

o) Повторяем предыдущий шаг до завершения всей нагрузки. А можно использовать DMA

4. Финальная фаза CCM
   Создаётся тег аутентификации и пишется в регистр CRYP_DOUT.

p) Пишем '11' в биты GCM_CCMPH[1:0] регистра CRYP_CR.

q) Пишем начальный счётчик A0 и пишем 128-бит значение A0 по 32 бита в CRYP_DIN.

r) Ждём установки флага OFNE (FIFO не пуст) в регистре CRYP_SR.

s) За четыре чтения регистра CRYP_DOUT получаем тег аутентификации.

t) Выключаем криптопроцессор (бит CRYPEN в CRYP_CR = '0')

NB: Форматирование исходных пакетов B0 и B1 и сравнение тега выполняются программно.
Между фазами заголовка и тега криптопроцессор включать/выключать не нужно.

Код аутентификации кодированного сообщения AES (CMAC)

Алгоритм CMAC идентичен CCM, но фаза нагрузки пропускается.

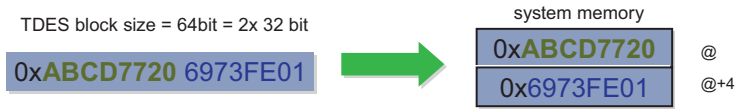
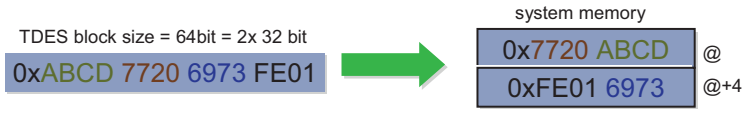
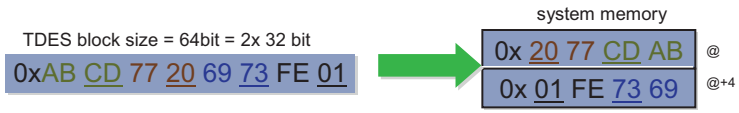
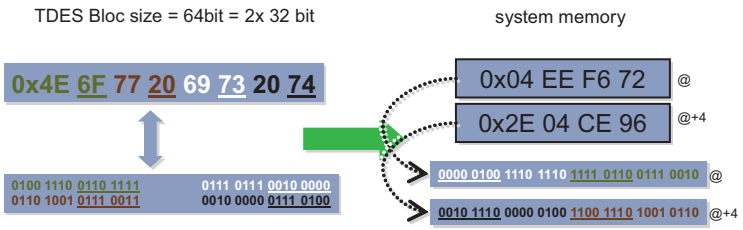
23.3.3. Типы данных

Данные в процессор CRYP подаются по 32 бита (словами) через регистр CRYP_DIN. В алгоритме DES поток обрабатывается блоками по 64 бит, нумерованными слева направо, от M1 до M64, где M1 это самый левый бит, а M64 самый правый бит блока. Алгоритм AES имеет 128-бит блоки.

Системная память имеет *little-endian* организацию: независимо от типа данных (бит, байт, 16-бит полуслово, 32-бит слово) менее значащие данные занимают меньшие адреса. Стало быть при передаче их из IN FIFO в процессор CRYP их нужно менять местами. Точно также это нужно при записи данных из CRYP процессора в OUT FIFO.

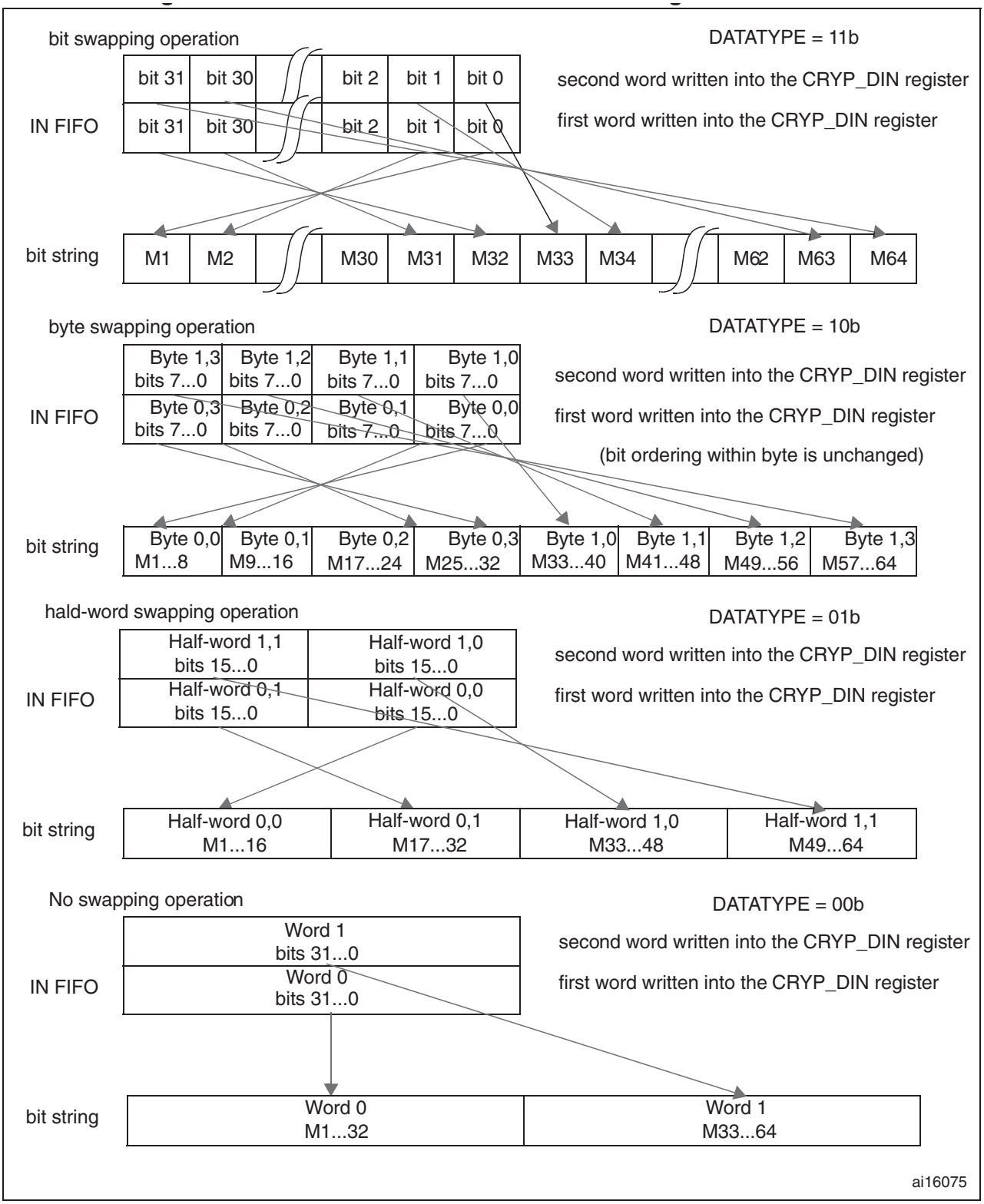
Тип обрабатываемых данных задаётся в поле DATATYPE регистра CRYP_CR.

Таблица 113. Типы данных

DATATYPE in CRYP_CR	Swapping performed	System memory data (plaintext or cypher)
00b	No swapping	Example: TDES block value **0xABCD77206973FE01** is represented in system memory as: 
01b	Half-word (16-bit) swapping	Example: TDES block value **0xABCD77206973FE01** is represented in system memory as: 
10b	Byte (8-bit) swapping	Example: TDES block value **0xABCD77206973FE01** is represented in system memory as: 
11b	Bit swapping	TDES block value **0x4E6F772069732074** is represented in system memory as: 

(По-моему, всё понятно и так)

Рис. 229. Построение 64-бит блока в соответствии с DATATYPE



NB: Операция обмена идентична для передач входного и выходного FIFO.

23.3.4. Векторы инициализации - CRYP_IV0...1(L/R)

Это два данных по 64 бита. Их формат и представление в системной памяти отличается от простого и шифрованного текста и на них не влияет значение DATATYPE.

Они представлены как два последовательных 32-бит слова, CRYP_IVL (левое, биты IV1...32) и CRYP_IVR (правое, биты IV33...64).

При кодировании DES или TDES CBC, биты CRYP_IV0(L/R) по XOR складываются с 64-бит данными из IN FIFO (после обмена в соответствии с DATATYPE), то есть, с битами M1...64 блока

данных. Результат из блока DEA3 копируется назад в вектор CRYP_IV0(L/R), который используется для XOR со следующим блоком из IN FIFO, и т. д.

При декодировании DES или TDES CBC, биты CRYP_IV0(L/R) по XOR складываются с 64-бит блоком данных (биты M1…64) из TDEA1 (перед обменом в соответствии с DATATYPE), обмениваются и пихаются в OUT FIFO. После этого значение CRYP_IV0(L/R) заменяется на выход из IN FIFO, IN FIFO извлекается. Можно обрабатывать новый 64-бит блок.

При кодировании AES CBC биты CRYP_IV0...1(L/R) по XOR складываются с 128-бит блоком из IN FIFO (после обмена в соответствии с DATATYPE). Результат из ядра AES копируется в вектор CRYP_IV0…1(L/R), который используется для XOR со следующим блоком из IN FIFO, и т. д.

При декодировании AES CBC биты CRYP_IV0...1(L/R) по XOR складываются с 128-бит блоком из ядра AES (перед обменом в соответствии с DATATYPE), обмениваются и пихаются в OUT FIFO. После этого значение CRYP_IV0...1(L/R) заменяется на выход из IN FIFO, IN FIFO извлекается. Можно обрабатывать новый 128-бит блок.

При кодировании и декодировании AES CTR биты CRYP_IV0…1(L/R) кодируются ядром AES. Результат кодирования складывается по XOR с 128-бит блоком из IN FIFO (после обмена в соответствии с DATATYPE). Результат XOR обменивается и пихается в OUT FIFO, часть счётчика CRYP_IV0...1(L/R) (32 LSB) инкрементируются.

Писать в регистры CRYP_IV0…1(L/R) можно только при сброшенном бите BUSY в регистре CRYP_SR.

См. Рис. 230 ниже.

23.3.5. CRYP занят

Криптопроцессор сам запускает кодирование или декодирование (в зависимости от состояния бита ALGODIR в регистре CRYP_CR) и стоящем бите CRYPEN = 1 регистре CRYP_CR если во входном FIFO есть достаточно данных (не меньше 2 слов для алгоритмов DES и TDES, 4 слова для AES) и хватает места в выходном FIFO (не меньше 2 (DES/TDES) или 4 (AES) слов).

Этот процесс занимает 48 тактов AHB2 для алгоритма Triple-DES, 16 тактов AHB2 для простого DES, и 14, 16 или 18 тактов AHB2 для AES с ключами 128, 192 или 256 бит, соответственно. В это время бит BUSY в CRYP_SR стоит. По концу процесса два (DES/TDES) или четыре (AES) слова пишутся в выходной FIFO и бит BUSY снимается. В режимах CBC и CTR также обновляются векторы инициализации CRYP_IVx(L/R)R (x = 0..3).

У занятого криптопроцессора (бит BUSY в CRYP_SR стоит) запись в регистры ключей (CRYP_Kx(L/R)R, x = 0..3), регистры инициализации (CRYP_IVx(L/R)R, x = 0..3) и в биты [9:2] регистра CRYP_CR игнорируется. Но во время работы можно снять бит CRYPEN, тогда работающие DES, TDES или AES завершают работу, пишут два или четыре слова результат в выходной FIFO и бит BUSY снимается.

NB: В режимах DES и TDES при переполнении выходного FIFO,и наличии хоть одного блока во входном FIFO, он оттуда извлекается, но бит BUSY останется стоять пока в выходном FIFO не появится место для нового блока.

23.3.6. Процедура кодирования или декодирования

Инициализация

1. Инициализируем устройство (порядок операций не важен, кроме подготовки ключа для AES-ECB или AES-CBC декодирования. Размер и значение ключа надо вводить до подготовки ключа и потом конфигурировать алгоритм):
   a) В биты KEYSIZE регистра CRYP_CR пишем размер ключа (128-, 192- или 256-бит для AES)
   b) В регистры CRYP_KxL/R пишем симметричный ключ (от 2 до 8 регистров по алгоритму)
   c) В биты DATATYPE регистра CRYP_CR пишем тип данных (1-, 8-, 16- или 32-bit)
   d) При декодировании в AES-ECB или AES-CBC готовим ключ: битами ALGOMODE = ‘111’ в CRYP_CR ставим режим. Пишем ‘1’ в CRYPEN: бит BUSY встаёт. Ждём снятия BUSY (CRYPEN снимется сам): ключ для декодирования готов

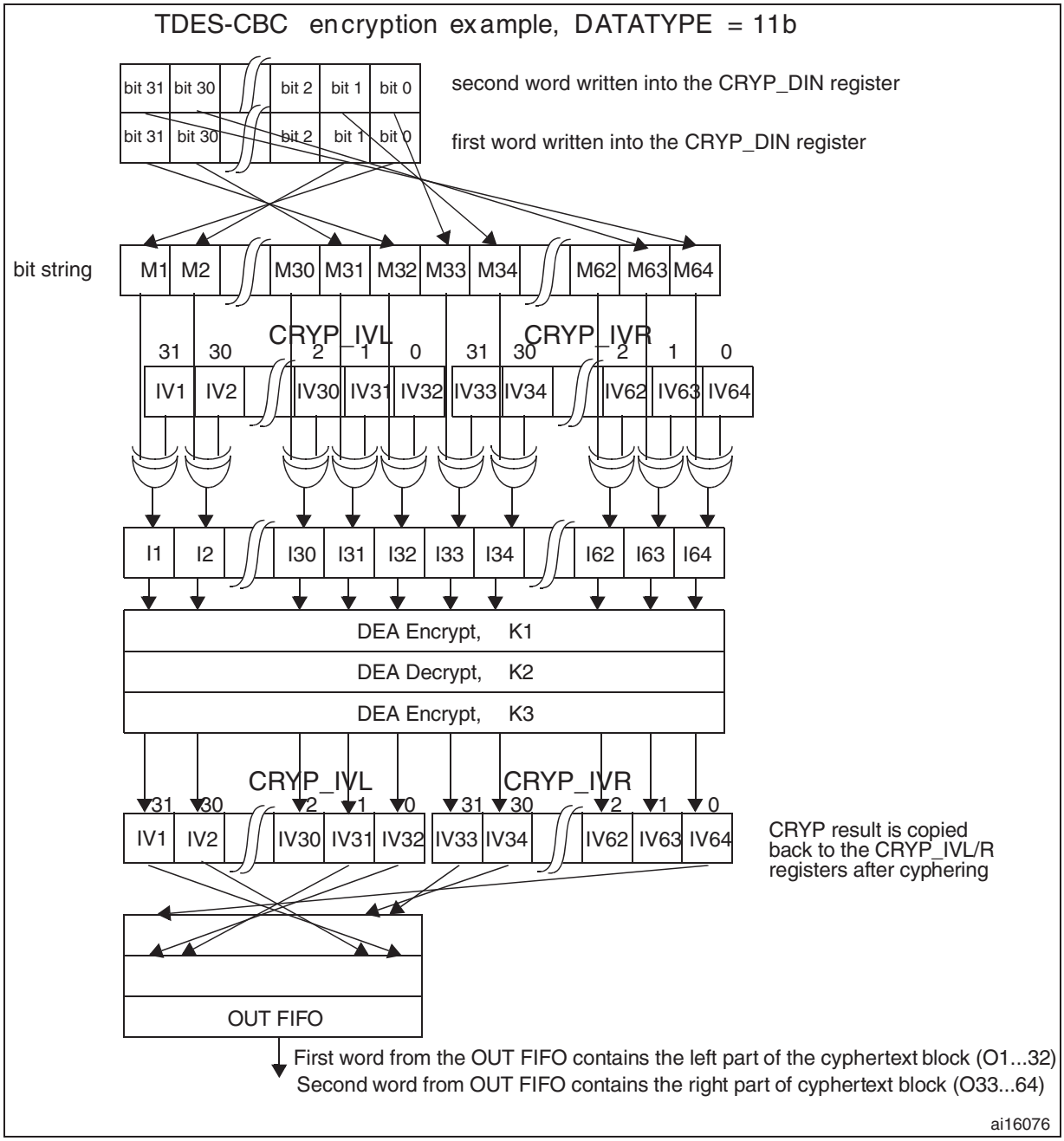
e) Битами ALGOMODE в регистре CRYP_CR задаём алгоритм и цепочки (DES/TDES в ECB/CBC, или AES в ECB/CBC/CTR/GCM/CCM)

f) Битом ALGODIR в CRYP_CR задаём направление (кодирование/декодирование)

g) В регистр CRYP_IVxL/R пишем вектор инициализации (только CBC или CTR)

2. Записью 1 в бит FFLUSH регистра CRYP_CR сливаем IN и OUT FIFO

Рис. 230. Векторы инициализации при TDES-CBC кодировании



Обработка с использованием DMA

1. Ставим контроллер DMA на передачу входных данных из памяти. Длина передачи равна длине сообщения. Устройства ради, длина сообщения должна быть целым числом блоков. Данные передаются в пакетном режиме. Длина пакета равна 4 словам для AES и 2 или 4 словам для DES/TDES. DMA должен выдавать прерывания по концу передачи выходных данных.
2. Включаем криптопроцессор установкой бита CRYPEN. Разрешаем запросы DMA установкой битов DIEN и DOEN в регистре CRYP_DMACR.
3. Всё работает само. Прерывание DMA показывает конец обработки. Оба FIFO обычно пусты и BUSY = 0.

Обработка с использованием прерываний

1. Установкой битов INIM и OUTIM в CRYP_IMSCR разрешаем прерывания.
2. Включаем криптопроцессор установкой бита CRYPEN в регистре CRYP_CR.
3. При управлении входными данными: пишем входное сообщение в IN FIFO. За раз можно писать 2 или 4 слова, или до заполнения FIFO. После записи последнего слова сообщения выключаем прерывание очисткой бита INIM.
4. При управлении выходными данными: читаем сообщение из OUT FIFO. За раз можно читать 1 блок (2 или 4 слова) или до опустения FIFO. По прочтению последнего слова INIM=0, BUSY=0 и оба FIFO пусты (IFEM=1 и OFNE=0). Выключают прерывания очисткой бита OUTIM и периферией, снятием бита CRYPEN.

Обработка без DMA и прерываний

1. Включаем криптопроцессор установкой бита CRYPEN в регистре CRYP_CR.
2. Пишем первые блоки во входной FIFO (от 2 до 8 слов).
3. Теперь до самого конца сообщения повторяем:
   a) Ждём OFNE=1, читаем OUT FIFO (1 блок или до опустения FIFO)
   b) Ждём IFNF=1, пишем в IN FIFO (1 блок или до заполнения FIFO)
4. По концу обработки BUSY=0 и оба FIFO пусты (IFEM=1 и OFNE=0). Периферия выключается очисткой бита CRYPEN.

23.3.7. Смена контекстов

Это пример переключения контекстов для разных задач ОС с использованием DMA:

Для AES и DES

1. Сохранение контекста
   a) Стопорим передачи DMA с IN FIFO снятием бита DIEN в регистре CRYP_DMACR.
   b) Ждём освобождения обоих FIFO (IFEM=1 и OFNE=0 в CRYP_SR) и бит BUSY чист.
   c) Стоп передачам DMA с OUT FIFO снятием бита DOEN в CRYP_DMACR и чистим CRYPEN.
   d) Сохраняем текущую конфигурацию (биты [9:2] и бит 19 из CRYP_CR) и, не в режиме ECB, векторы инициализации. Значение ключа уже должно быть доступно в памяти. Если надо, сохраняем статус DMA (указатели для IN и OUT сообщений, остаток байтов и пр.).
   e) Для алгоритмов GCM/GMAC и CCM/CMAC ещё надо сохранить:
      – биты [17:16] регистра CRYP_CR
      – регистры смены контекстов:
      CRYP_CSGCMCCM0..7 для GCM/GMAC и CCM/CMAC
      CRYP_CSGCM0..7 для GCM/GMAC.
2. Конфигурируем и выполняем другую обработку.
3. Восстановление контекста
   a) Конфигурируем процессор как в секции 23.3.6 с сохранённой конфигурацией. Для декодирования AES-ECB и AES-CBC ключ надо подготовить снова.
   b) Если надо, конфигурируем контроллер DMA на передачу остатка сообщения.
   c) Включаем процессор установкой бита CRYPEN и DMA битами DIEN и DOEN.

Для TDES

Делается также, как и для AES, но во входном FIFO может содержаться до 4 необработанных блоков, и иногда может быть быстрее прервать их обработку, чем ждать её завершения.

1. Сохранение контекста
   a) Стопорим передачи DMA с IN FIFO очисткой бита DIEN в регистре CRYP_DMACR.
   b) Выключаем процессор снятием бита CRYPEN (текущий блок будет завершён).
   c) Ждём пустого OUT FIFO (OFNE=0 в регистре CRYP_SR) и бит BUSY чист.

d) Стопорим передачи DMA с OUT FIFO чисткой бита DOEN в регистре CRYP_DMACR.

e) Сохраняем текущую конфигурацию (биты [9:2] и бит 19 из CRYP_CR) и, не в режиме ECB, векторы инициализации. Значение ключа уже должно быть доступно в памяти. Если надо, сохраняем статус DMA (указатели для IN и OUT сообщений, остаток байтов и пр.). Вычитываем из IN FIFO необработанные данные и сохраняем их в памяти.

В режимах GCM/GMAC и CCM/CMAC ещё надо сохранить биты [17:16] регистра CRYP_CR.

2. Конфигурируем и выполняем другую обработку.
3. Восстановление контекста

a) Конфигурируем процессор как в секции 23.3.6 с сохранённой конфигурацией. Для декодирования AES-ECB и AES-CBC ключ надо подготовить снова.

b) Пишем ранее сохранённые данные обратно в IN FIFO.

c) Если надо, конфигурируем контроллер DMA на передачу остатка сообщения.

d) Включаем процессор установкой бита CRYPEN и DMA битами DIEN и DOEN.

23.4. Прерывания CRYP

CRYP подаёт на NVIC один запрос прерываний, объединяющий по OR два маскируемых источника. Разрешение каждого из них ставится в регистре CRYP_IMSCR.

В регистре CRYP_RISR выдаётся статус исходных, а в регистре CRYP_MISR маскированных прерываний.

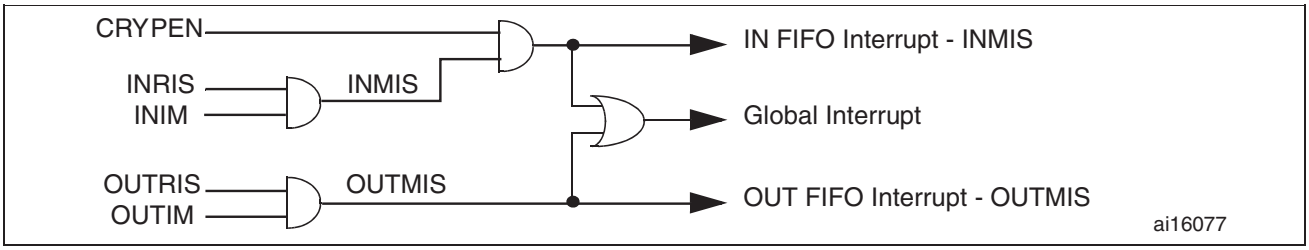
Прерывание выходного FIFO - OUTMIS

Выставляется при наличии хоть одного 32-бит слова в выходном FIFO (отражает состояние флага OFNE). Бит разрешения CRYP на прерывание OUTMIS не влияет, прочитать надо весь FIFO.

Прерывание входного FIFO - INMIS

Выставляется, если во входном FIFO меньше четырёх слов (есть свободное место). Снимается заполнением FIFO. Бит разрешения CRYP разрешает и прерывание INMIS.

Рис. 231. Схема прерываний CRYP



23.5. Интерфейс CRYP DMA

Связью с контроллером DMA управляют через регистр CRYP_DMACR.

Запросы пакетных и одинарных передач могут выставляться одновременно. Например, если в OUT FIFO есть 6 слов, то выставляются оба запроса. Сначала обрабатывается пакетная передача для 4 слов, а затем одинарные для оставшихся 2 слов.

Снимается запрос соответствующим сигналом очистки от DMA. После снятия сигнала очистки, запрос может встать снова, если есть нужное условие. При выключении CRYP или при снятии битов разрешения (DIEN для IN FIFO и DOEN для OUT FIFO в регистре CRYP_DMACR) запросы снимаются.

NB: Контроллер DMA должен передавать пакеты по 4 слова или меньше. Иначе можно потерять некоторые данные, что отвратительно. Канал OUTDMA должен быть приоритетнее, чем INDMA.

23.6. Регистры CRYP

Всего наличествуют несколько регистров управления и состояния, восемь регистров ключа и четыре регистра векторов инициализации.

23.6.1. Регистр управления (CRYP_CR) для STM32F415/417xx

Смещение адреса: 0x00

По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRYPEN	FFLUSH	Reserved				KEYSIZE		DATATYPE		ALGOMODE[2:0]			ALGODIR	Res.	Res.
rw	w					rw	rw	rw	rw	rw	rw	rw	rw		

— Биты 31:16 Резерв, не трогать.

— Бит 15 **CRYPEN**: Включение процессора

0: CRYP выкл.

1: CRYP вкл.

NB: Бит CRYPEN автоматически снимается по концу подготовки ключа (ALGOMODE=111b) или фазы инициализации GCM_CCM

— Бит 14 **FFLUSH**: Слив FIFO

При CRYPEN = 0, запись 1 сливает IN и OUT FIFO (сбрасываются указатели чтения и записи FIFO.). Запись 0 бессмысленна.

При CRYPEN = 1, любая запись бестолкова.
Чтение всегда возвращает 0.

— Биты 13:10 Резерв, не трогать.

— Биты 9:8 **KEYSIZE[1:0]**: Размер ключа (только для AES)

00: 128 бит

01: 192 бит

10: 256 бит

11: Резерв, не писать

— Биты 7:6 **DATATYPE[1:0]**: Тип данных в регистре CRYP_DIN.

00: 32-бит. Слова не меняются. Первое слово, передаваемое из/в FIFO это биты 1...32 блока данных, второе слово - биты 33…64.

01: 16-бит, полуслово. Каждое слово, передаваемое из/в FIFO это 2 полуслова, меняются местами.

10: 8-бит, байт. Каждое слово, передаваемое из/в FIFO это 4 байта, меняются местами.

11: биты, строка. Каждое слово, передаваемое из/в FIFO это 32 бита (1й бит в позиции 0), меняются местами друг с другом.

— Биты 5:3 **ALGOMODE[2:0]**: Режим алгоритма

000: TDES-ECB (triple-DES): Без связи между блоками. Векторы инициализации (CRYP_IV0(L/R)) не используются, работают три ключа (K1, K2, и K3), K0 не используется.

001: TDES-CBC (triple-DES, цепочка блоков): Перед подачей в алгоритм входной блок XOR-ится с выходным. Векторы инициализации (CRYP_IV0(L/R)) используются, работают три ключа (K1, K2, и K3), K0 не используется.

010: DES-ECB (простой DES): Без связи между блоками. Векторы инициализации (CRYP_IV0(L/R)) используются, работает ключ K1, остальные (K0, K2, K3) не используются.

011: DES-CBC (простой DES, цепочка блоков): Перед подачей в алгоритм входной блок XOR-ится с выходным. Векторы инициализации (CRYP_IV0(L/R)) используются, работает ключ K1, остальные (K0, K2, K3) не используются.

100: AES-ECB (простой AES): Без связи между блоками. Векторы инициализации (CRYP_IV0L/R...1L/R) не используются, работают все ключи (K0...K3).

101: AES-CBC (AES, цепочка блоков): Перед подачей в алгоритм входной блок XOR-ится с выходным. Векторы инициализации (CRYP_IV0L/R...1L/R)) используются, работают все ключи (K0...K3).

110: AES-CTR (AES режим счётчика): Перед подачей в алгоритм входной блок XOR-ится с выходным. Векторы инициализации (CRYP_IV0L/R...1L/R)) используются, работают все ключи (K0...K3). Кодирование в CTR не отличается от декодирования, ядро всегда кодирует текущий блок счётчика, что XOR-ится с блоком данных на входе. ALGODIR не влияет, готовить ключ НЕ надо.

111: Подготовка ключа AES для декодирования. Запись этого при CRYPEN = 1 сразу запускает создание ключа. В регистрах K0...K3 уже должен быть секретный ключ. Бит BUSY встаёт сам и после записи нового ключа в K0...K3 снимается.

— Бит 2 **ALGODIR**: Направление алгоритма

0: Кодирование, 1: Декодирование.

— Биты 1:0 Резерв, не трогать.

NB: Биты KEYSIZE, DATATYPE, ALGOMODE и ALGODIR при BUSY=1 не пишутся.

Бит FFLUSH надо писать при BUSY=0. Иначе в выходном FIFO после слива может появиться ещё какой-нибудь блок.

23.6.2. Регистр управления (CRYP_CR) для STM32F42xxx/43xxx

Смещение адреса: 0x00

По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved												ALGO MODE [3]	Res.	GCM_CCMPH	
												rw		rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRYPEN	FFLUSH	Reserved				KEYSIZE		DATATYPE		ALGOMODE[2:0]			ALGODIR	Reserved	
rw	w					rw	rw	rw	rw	rw	rw	rw	rw		

— Биты 31:16 Резерв, не трогать.

— Бит 19 **ALGOMODE[3]**: Включение процессора

— Бит 18 Резерв, не трогать.

— Биты 17:16 **GCM_CCMPH[1:0]**: Фаза только для алгоритмов GCM или CCM

00: Инициализация GCM_CCM

01: Заголовок GCM_CCM

10: Нагрузка GCM_CCM

11: Завершение GCM_CCM

— Бит 15 **CRYPEN**: Включение процессора

0: CRYP выкл.

1: CRYP вкл.

NB: Бит CRYPEN автоматически снимается по концу подготовки ключа (ALGOMODE=111b) или фазы инициализации GCM_CCM

— Бит 14 **FFLUSH**: Слив FIFO

При CRYPEN = 0, запись 1 сливает IN и OUT FIFO (сбрасываются указатели чтения и записи FIFO.). Запись 0 бессмысленна.

При CRYPEN = 1, любая запись бестолкова.
Чтение всегда возвращает 0.

— Биты 13:10 Резерв, не трогать.

— Биты 9:8 **KEYSIZE[1:0]**: Размер ключа (только для AES)

00: 128 бит

01: 192 бит

10: 256 бит

11: Резерв, не писать

— Биты 7:6 **DATATYPE[1:0]**: Тип данных в регистре CRYP_DIN.

00: 32-бит. Слова не меняются. Первое слово, передаваемое из/в FIFO это биты 1...32 блока данных, второе слово - биты 33…64.

01: 16-бит, полуслово. Каждое слово, передаваемое из/в FIFO это 2 полуслова, меняются местами.

10: 8-бит, байт. Каждое слово, передаваемое из/в FIFO это 4 байта, меняются местами.

11: биты, строка. Каждое слово, передаваемое из/в FIFO это 32 бита (1й бит в позиции 0), меняются местами друг с другом.

— Биты 19,5:3 **ALGOMODE[2:0]**: Режим алгоритма

0000: TDES-ECB (triple-DES): Без связи между блоками. Векторы инициализации (CRYP_IV0(L/R)) не используются, работают три ключа (K1, K2, и K3), K0 не используется.

0001: TDES-CBC (triple-DES, цепочка блоков): Перед подачей в алгоритм входной блок XOR-ится с выходным. Векторы инициализации (CRYP_IV0(L/R)) используются, работают три ключа (K1, K2, и K3), K0 не используется.

0010: DES-ECB (простой DES): Без связи между блоками. Векторы инициализации (CRYP_IV0(L/R)) используются, работает ключ K1, остальные (K0, K2, K3) не используются.

0011: DES-CBC (простой DES, цепочка блоков): Перед подачей в алгоритм входной блок XOR-ится с выходным. Векторы инициализации (CRYP_IV0(L/R)) используются, работает ключ K1, остальные (K0, K2, K3) не используются.

0100: AES-ECB (простой AES): Без связи между блоками. Векторы инициализации (CRYP_IV0L/R...1L/R) не используются, работают все ключи (K0...K3).

0101: AES-CBC (AES, цепочка блоков): Перед подачей в алгоритм входной блок XOR-ится с выходным. Векторы инициализации (CRYP_IV0L/R...1L/R)) используются, работают все ключи (K0...K3).

0110: AES-CTR (AES режим счётчика): Перед подачей в алгоритм входной блок XOR-ится с выходным. Векторы инициализации (CRYP_IV0L/R...1L/R)) используются, работают все ключи (K0...K3). Кодирование в CTR не отличается от декодирования, ядро всегда кодирует текущий блок счётчика, что XOR-ится с блоком данных на входе. ALGODIR не влияет, готовить ключ НЕ надо.

0111: Подготовка ключа AES для декодирования. Запись этого при CRYPEN = 1 сразу запускает создание ключа. В регистрах K0...K3 уже должен быть секретный ключ. Бит BUSY встаёт сам и после записи нового ключа в K0...K3 снимается.

1000: Режим счётчика Галуа (GCM). Также используется для алгоритма GMAC.

1001: Счётчик с CBC-MAC (CCM). Также используется для алгоритма CMAC.

— Бит 2 **ALGODIR**: Направление алгоритма

0: Кодирование,

1: Декодирование.

— Биты 1:0 Резерв, не трогать.

NB: Биты KEYSIZE, DATATYPE, ALGOMODE и ALGODIR при BUSY=1 не пишутся.

Бит FFLUSH надо писать при BUSY=0. Иначе в выходном FIFO после слива может появиться ещё какой-нибудь блок.

23.6.3. Регистр состояния (CRYP_SR)

Смещение адреса: 0x04

По сбросу: 0x0000 0003.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											BUSY	OFFU	OFNE	IFNF	IFEM
											r	r	r	r	r

— Биты 31:5 Резерв, не трогать.

— Бит 4 **BUSY**: Процессор занят

0: Ядро CRYP ничего не делает. Причина:

– Выключено (CRYPEN=0 в CRYP_CR) или всё уже сделано,

– Ждёт достаточно данных в IN FIFO или места в OUT FIFO (2 слова в DES, 4 в AES).

1: Ядро CRYP работает над блоком или ключом для декодирования AES.

— Бит 3 **OFFU**: Выходной FIFO полон

0: OUT FIFO не полон

1: OUT FIFO полон

— Бит 2 **OFNE**: Выходной FIFO не пуст

0: OUT FIFO пуст

1: OUT FIFO не пуст

— Бит 1 **IFNF**: Входной FIFO не полон

0: IN FIFO полон

1: IN FIFO не полон

— Бит 0 **IFEM**: Входной FIFO пуст

0: IN FIFO не пуст

1: IN FIFO пуст

23.6.4. Входной регистр данных (CRYP_DIN)

Смещение адреса: 0x08

По сбросу: 0x0000 0000.

Предназначен для записи блоков данных в IN FIFO словами. Первыми пишутся старшие биты блока. Не учитывая обмен данных, это даёт:

- В режиме DES/TDES: блок это последовательность битов с номерами от 1 (самый левый) до 64 (наиправейший). Бит 1 это MSB (бит 31) первого слова в FIFO, бит 64 это LSB (бит 0) второго слова в FIFO.
- В режиме AES: блок это последовательность битов с номерами от 0 (самый левый) до 127 (наиправейший). Бит 0 это MSB (бит 31) первого слова в FIFO, бит 127 это LSB (бит 0) четвёртого слова в FIFO.

Данные из регистра CRYP_DIN перед обработкой могут проходить обмен в соответствии с битами DATATYPE регистра CRYP_CR. См. Секцию 23.3.3.

Данные из CRYP_DIN идут в IN FIFO. Процессор CRYP начинает обработку при появлении в IN FIFO не меньше двух 32-бит слов для DES/TDES (или четырёх 32-бит слов для AES) и не менее 2 свободных слов в OUT FIFO. Процесс сам переносит обработанные данные в OUT FIFO.

Чтение регистра CRYP_DIN:

- При CRYPEN = 0, из FIFO первым извлекается самое старое записанное слово, последним - самое раннее записанное слово. И конечно же надо глядеть на флаг IFEM на предмет опустения FIFO.
- При CRYPEN = 1, возвращается вообще что-нибудь.

После чтения данных из CRYP_DIN, перед новой обработкой FIFO нужно слить битом FFLUSH.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATAIN															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAIN															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:0 **DATAIN**: Ввод данных
Чтение: При CRYPEN = 0 выдаётся слово из IN FIFO, иначе - белиберда.
Запись: Слово пихается в IN FIFO.

23.6.5. Выходной регистр данных (CRYP_DOUT)

Смещение адреса: 0x0C

По сбросу: 0x0000 0000.

Предназначен для чтения блоков данных из OUT FIFO словами. Первыми читаются старшие биты блока. Не учитывая обмен данных, это даёт:

- В режиме DES/TDES: блок это последовательность битов с номерами от 1 (самый левый) до 64 (наиправейший). Бит 1 это MSB (бит 31) первого слова в FIFO, бит 64 это LSB (бит 0) второго слова в FIFO.
- В режиме AES: блок это последовательность битов с номерами от 0 (самый левый) до 127 (наиправейший). Бит 0 это MSB (бит 31) первого слова в FIFO, бит 127 это LSB (бит 0) четвёртого слова в FIFO.

Данные после обработки могут проходить обмен в соответствии с битами DATATYPE регистра CRYP_CR. См. Секцию 23.3.3.

При чтении из CRYP_DOUT возвращается слово по указателю чтения.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATAOUT															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAOUT															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

— Биты 31:0 **DATAOUT**: Вывод данных
Чтение: Выдаётся слово из OUT FIFO.
Запись: А ничего.

23.6.6. Регистр управления CRYP DMA (CRYP_DMACR)

Смещение адреса: 0x10

По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														DOEN	DIEN
														rw	rw

— Биты 31:2 Резерв, не трогать.

— Бит 1 **DOEN**: Разрешение выходного DMA

0: Нельзя

1: Можно

— Бит 0 **DIEN**: Разрешение входного DMA

0: Нельзя

1: Можно

23.6.7. Регистр маски прерываний CRYP (CRYP_IMSCR)

Смещение адреса: 0x14

По сбросу: 0x0000 0000.

Запись 1 ставит маску, разрешая прерывание, запись 0 запрещая его. По сбросу обнуляется.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														OUTIM	INIM
														rw	rw

— Биты 31:2 Резерв, не трогать.

— Бит 1 **OUTIM**: Разрешение прерываний выходного FIFO

0: Нельзя

1: Можно

— Бит 0 **INIM**: Разрешение прерываний входного FIFO

0: Нельзя

1: Можно

23.6.8. Регистр состояния необработанных прерываний (CRYP_RISR)

Смещение адреса: 0x18

По сбросу: 0x0000 0001.

Выдаёт состояние запросов прерываний до маскирования.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														OUTRIS	INRIS
														r	r

— Биты 31:2 Резерв, не трогать.

— Бит 1 **OUTRIS**: Состояние запроса прерываний выходного FIFO

0: Нету

1: Есть

— Бит 0 **INRIS**: Состояние запроса прерываний входного FIFO

0: Нету

1: Есть

23.6.9. Регистр состояния маскированных прерываний (CRYP_MISR)

Смещение адреса: 0x1C

По сбросу: 0x0000 0000.

Выдаёт состояние запросов прерываний после маскирования.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														OUTMIS	INMIS
														r	r

— Биты 31:2 Резерв, не трогать.

— Бит 1 **OUTMIS**: Состояние запроса прерываний выходного FIFO

0: Нету

1: Есть

— Бит 0 **INMIS**: Состояние запроса прерываний входного FIFO

0: Нету

1: Есть при CRYPEN=0

23.6.10.Регистры ключа (CRYP_K0...3(L/R)R)

Смещение адреса: 0x20 до 0x3C

По сбросу: 0x0000 0000.

В режиме TDES ключ это 64-бит число (нумерация слева направо, 1-64, зовут K1, K2 и K3, K0 не используется), из 56 информационных битов и 8 битов чётности. В текущем блоке Kx[1:64] биты чётности (8, 16, 24, 32, 40, 48, 56 и 64) не используются.

В режиме AES ключ это число, длиной 128-, 192- или 256-бит, k0k1k2...k127/191/255 (k0 это самый левый бит). Ключ AES пишется в регистры так:

- для AES-128: k0..k127 соответствует b127..b0 (b255..b128 не используются),
- для AES-192: k0..k191 соответствует b191..b0 (b255..b192 не используются),
- для AES-256: k0..k255 соответствует b255..b0.

В любом случае, b0 это самый правый бит.

CRYP_K0LR (смещение адреса: 0x20)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
b255	b254	b253	b252	b251	b250	b249	b248	b247	b246	b245	b244	b243	b242	b241	b240
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
b239	b238	b237	b236	b235	b234	b233	b232	b231	b230	b229	b228	b227	b226	b225	b224
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

CRYP_K0RR (смещение адреса: 0x24)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
b223	b222	b221	b220	b219	b218	b217	b216	b215	b214	b213	b212	b211	b210	b209	b208
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
b207	b206	b205	b204	b203	b202	b201	b200	b199	b198	b197	b196	b195	b194	b193	b192
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

CRYP_K1LR (смещение адреса: 0x28)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
k1.1 b191	k1.2 b190	k1.3 b189	k1.4 b188	k1.5 b187	k1.6 b186	k1.7 b185	k1.8 b184	k1.9 b183	k1.10 b182	k1.11 b181	k1.12 b180	k1.13 b179	k1.14 b178	k1.15 b177	k1.16 b176
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
k1.17 b175	k1.18 b174	k1.19 b173	k1.20 b172	k1.21 b171	k1.22 b170	k1.23 b169	k1.24 b168	k1.25 b167	k1.26 b166	k1.27 b165	k1.28 b164	k1.29 b163	k1.30 b162	k1.31 b161	k1.32 b160
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

CRYP_K1RR (смещение адреса: 0x2C)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
k1.33 b159	k1.34 b158	k1.35 b157	k1.36 b156	k1.37 b155	k1.38 b154	k1.39 b153	k1.40 b152	k1.41 b151	k1.42 b150	k1.43 b149	k1.44 b148	k1.45 b147	k1.46 b146	k1.47 b145	k1.48 b144
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
k1.49 b143	k1.50 b142	k1.51 b141	k1.52 b140	k1.53 b139	k1.54 b138	k1.55 b137	k1.56 b136	k1.57 b135	k1.58 b134	k1.59 b133	k1.60 b132	k1.61 b131	k1.62 b130	k1.63 b129	k1.64 b128
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

CRYP_K2LR (смещение адреса: 0x30)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
k2.1 b127	k2.2 b126	k2.3 b125	k2.4 b124	k2.5 b123	k2.6 b122	k2.7 b121	k2.8 b120	k2.9 b119	k2.10 b118	k2.11 b117	k2.12 b116	k2.13 b115	k2.14 b114	k2.15 b113	k2.16 b112
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
k2.17 b111	k2.18 b110	k2.19 b109	k2.20 b108	k2.21 b107	k2.22 b106	k2.23 b105	k2.24 b104	k2.25 b103	k2.26 b102	k2.27 b101	k2.28 b100	k2.29 b99	k2.30 b98	k2.31 b97	k2.32 b96
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

CRYP_K2RR (смещение адреса: 0x34)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
k2.33 b95	k2.34 b94	k2.35 b93	k2.36 b92	k2.37 b91	k2.38 b90	k2.39 b89	k2.40 b88	k2.41 b87	k2.42 b86	k2.43 b85	k2.44 b84	k2.45 b83	k2.46 b82	k2.47 b81	k2.48 b80
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
k2.49 b79	k2.50 b78	k2.51 b77	k2.52 b76	k2.53 b75	k2.54 b74	k2.55 b73	k2.56 b72	k2.57 b71	k2.58 b70	k2.59 b69	k2.60 b68	k2.61 b67	k2.62 b66	k2.63 b65	k2.64 b64
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

CRYP_K3LR (смещение адреса: 0x38)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
k3.1 b63	k3.2 b62	k3.3 b61	k3.4 b60	k3.5 b59	k3.6 b58	k3.7 b57	k3.8 b56	k3.9 b55	k3.10 b54	k3.11 b53	k3.12 b52	k3.13 b51	k3.14 b50	k3.15 b49	k3.16 b48
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
k3.17 b47	k3.18 b46	k3.19 b45	k3.20 b44	k3.21 b43	k3.22 b42	k3.23 b41	k3.24 b40	k3.25 b39	k3.26 b38	k3.27 b37	k3.28 b36	k3.29 b35	k3.30 b34	k3.31 b33	k3.32 b32
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

CRYP_K3RR (смещение адреса: 0x3C)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
k3.33 b31	k3.34 b30	k3.35 b29	k3.36 b28	k3.37 b27	k3.38 b26	k3.39 b25	k3.40 b24	k3.41 b23	k3.42 b22	k3.43 b21	k3.44 b20	k3.45 b19	k3.46 b18	k3.47 b17	k3.48 b16
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
k3.49 b15	k3.50 b14	k3.51 b13	k3.52 b12	k3.53 b11	k3.54 b10	k3.55 b9	k3.56 b8	k3.57 b7	k3.58 b6	k3.59 b5	k3.60 b4	k3.61 b3	k3.62 b2	k3.63 b1	k3.64 b0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

NB: При занятом процессоре (бит BUSY = 1 в CRYP_SR) запись в эти регистры игнорируется.

23.6.11.Регистры вектора инициализации (CRYP_IV0...1(L/R)R)

Смещение адреса: 0x40 до 0x4C

По сбросу: 0x0000 0000.

Вектор инициализации (64 бита для DES/TDES и 128 бит для AES) используется в режимах CBC и CTR.

IV0 это самый левый бит, а IV63 (DES, TDES) или IV127 (AES) это самый левый бит вектора инициализации. IV1(L/R)R используется только в AES.

CRYP_IV0LR (смещение адреса: 0x40)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IV0	IV1	IV2	IV3	IV4	IV5	IV6	IV7	IV8	IV9	IV10	IV11	IV12	IV13	IV14	IV15
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IV16	IV17	IV18	IV19	IV20	IV21	IV22	IV23	IV24	IV25	IV26	IV27	IV28	IV29	IV30	IV31
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

CRYP_IV0RR (смещение адреса: 0x44)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IV32	IV33	IV34	IV35	IV36	IV37	IV38	IV39	IV40	IV41	IV42	IV43	IV44	IV45	IV46	IV47
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IV48	IV49	IV50	IV51	IV52	IV53	IV54	IV55	IV56	IV57	IV58	IV59	IV60	IV61	IV62	IV63
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

CRYP_IV1LR (смещение адреса: 0x48)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IV64	IV65	IV66	IV67	IV68	IV69	IV70	IV71	IV72	IV73	IV74	IV75	IV76	IV77	IV78	IV79
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IV80	IV81	IV82	IV83	IV84	IV85	IV86	IV87	IV88	IV89	IV90	IV91	IV92	IV93	IV94	IV95
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

CRYP_IV1RR (смещение адреса: 0x4C)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IV96	IV97	IV98	IV99	IV100	IV101	IV102	IV103	IV104	IV105	IV106	IV107	IV108	IV109	IV110	IV111
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IV112	IV113	IV114	IV115	IV116	IV117	IV118	IV119	IV120	IV121	IV122	IV123	IV124	IV125	IV126	IV127
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

NB: В режимах DES/3DES используется только `CRYP_IV0(L/R)`.

При занятом процессоре (бит `BUSY` = 1 в `CRYP_SR`) запись в эти регистры игнорируется.

23.6.12.Регистры смены контекста (CRYP_CSGCMCCM0..7R и CRYP_CSGCM0..7R) для STM32F42xxx STM32F43xxx

Смещение адреса:

- `CRYP_CSGCMCCM0..7`: 0x050 до 0x06C: только для GCM/GMAC или CCM/CMAC
- `CRYP_CSGCM0..7`: 0x070 до 0x08C: только для GCM/GMAC

По сбросу: 0x0000 0000.

Содержат всё состояние внутренних регистров CRYP при алгоритмах GCM/GMAC и CCM/CMAC. Крайне полезно сохранять и восстанавливать при переключении задач ОС.

CRYP_CSGCMCCMxR: где x=[7:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRYP_CSGCMCCMxR															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRYP_CSGCMCCMxR															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

CRYP_CSGCMxR: где x=[7:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRYP_CSGCMxR															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRYP_CSGCMxR															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

23.6.13.Карта регистров CRYP

Таблица 114. Для STM32F415/417xx

Offset	Register name and reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00 0x00	CRYP_CR	Reserved																CRYPEN	FFLUSH	Reserved				KEYSIZE		DATATYPE		ALOMODE[2:0]			ALGODIR	Res.	
	Reset value																	0	0					0	0	0	0	0	0	0	0		
0x04	CRYP_SR	Reserved																											BUSY	OFFU	OFNE	IFNF	IFEM
	Reset value																												0	0	0	1	1
0x08	CRYP_DIN	DATAIN																															
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	CRYP_DOUT	DATAOUT																															
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	CRYP_DMACR	Reserved																														DOEN	DIEN
	Reset value																															0	0
0x14	CRYP_IMSCR	Reserved																														OUTIM	INIM
	Reset value																															0	0
0x18	CRYP_RISR	Reserved																														OUTRIS	INRIS
	Reset value																															0	1
0x1C	CRYP_MISR	Reserved																														OUTMIS	IN%IS
	Reset value																															0	0
0x20	CRYP_K0LR	CRYP_K0LR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	CRYP_K0RR	CRYP_K0RR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...																																	
0x38	CRYP_K3LR	CRYP_K3LR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	CRYP_K3RR	CRYP_K3RR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	CRYP_IV0LR	CRYP_IV0LR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	CRYP_IV0RR	CRYP_IV0RR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	CRYP_IV1LR	CRYP_IV1LR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	CRYP_IV1RR	CRYP_IV1RR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Таблица 115. Для STM32F43xxx

0x1C	CRYP_MISR	Reserved																														OUTMIS	IN%IS
	Reset value																															0	0
0x20	CRYP_K0LR	CRYP_K0LR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	CRYP_K0RR	CRYP_K0RR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		...																															
		...																															
0x38	CRYP_K3LR	CRYP_K3LR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	CRYP_K3RR	CRYP_K3RR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	CRYP_IV0LR	CRYP_IV0LR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	CRYP_IV0RR	CRYP_IV0RR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	CRYP_IV1LR	CRYP_IV1LR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	CRYP_IV1RR	CRYP_IV1RR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	CRYP_ CSGCMCCM R	CRYP_CSGCMCCM0R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x54	CRYP_ CSGCMCCM 1R	CRYP_CSGCMCCM1R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x58	CRYP_ CSGCMCCM 2R	CRYP_CSGCMCCM2R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C	CRYP_ CSGCMCCM 3R	CRYP_CSGCMCCM3R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

0x00 0x00	CRYP_CR	Reserved												ALGOMODE[3]	Res.	GCM_CCMPH		CRYPEN	FFLUSH	Reserved				KEYSIZE		DATATYPE		ALOMODE[2:0]			ALGODIR	Res..	
	Reset value															0	0	0	0					0	0	0	0	0	0	0	0		
0x04	CRYP_SR	Reserved																											BUSY	OFFU	OFNE	IFNF	IFEM
	Reset value																												0	0	0	1	1
0x08	CRYP_DIN	DATAIN																															
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	CRYP_DOUT	DATAOUT																															
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	CRYP_DMAC R	Reserved																														DOEN	DIEN
	Reset value																															0	0
0x14	CRYP_IMSC R	Reserved																														OUTIM	INIM
	Reset value																															0	0
0x18	CRYP_RISR	Reserved																														OUTRIS	INRIS
	Reset value																															0	1

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x60	CRYP_CSGCMCCM4R	CRYP_CSGCMCCM4R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x64	CRYP_CSGCMCCM5R	CRYP_CSGCMCCM5R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x68	CRYP_CSGCMCCM6R	CRYP_CSGCMCCM6R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6C	CRYP_CSGCMCCM7R	CRYP_CSGCMCCM7R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x70	CRYP_CSGCM0R	CRYP_CSGCM0R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x74	CRYP_CSGCM1R	CRYP_CSGCM1R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x78	CRYP_CSGCM2R	CRYP_CSGCM2R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x7C	CRYP_CSGCM3R	CRYP_CSGCM3R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x80	CRYP_CSGCM4R	CRYP_CSGCM4R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x84	CRYP_CSGCM5R	CRYP_CSGCM5R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x88	CRYP_CSGCM6R	CRYP_CSGCM6R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8C	CRYP_CSGCM7R	CRYP_CSGCM7R																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

24. Генератор случайных чисел (RNG)

24.1. Введение в RNG

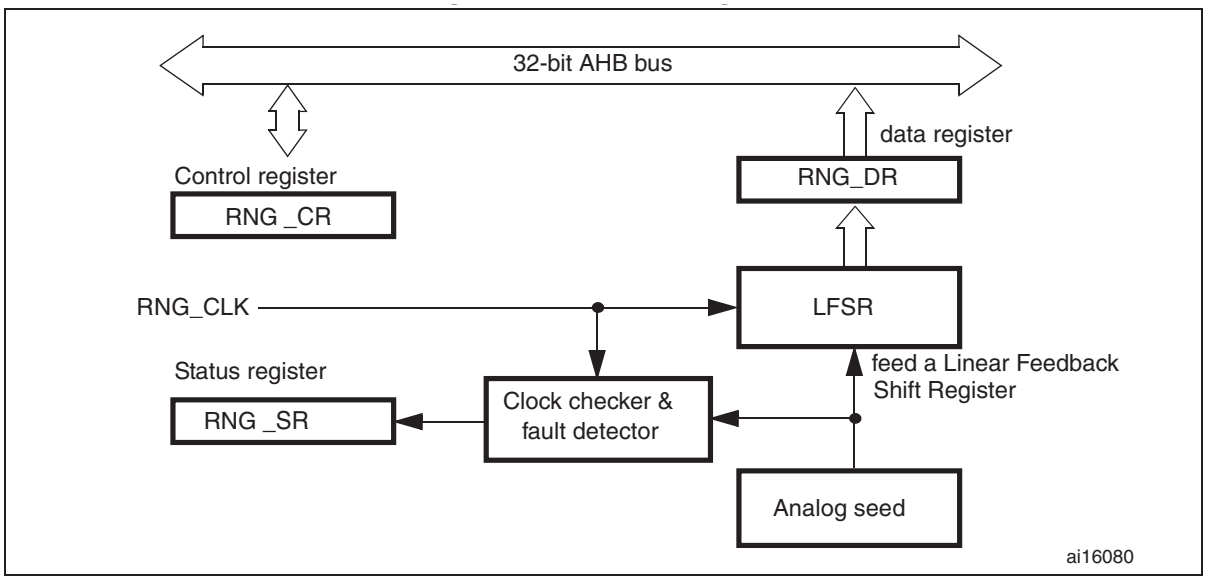
Процессор RNG основан аналоговом генераторе белого шума и выдаёт случайные 32-бит числа. Он прошёл тесты FIPS PUB 140-2 (2001 October 10) с успехом 99%.

24.2. Основные свойства RNG

- Выдаёт случайные 32-бит числа от аналогового генератора
- Последовательные случайные числа выдаются через 40 тактов RNG_CLK
- Контроль энтропии RNG с отметкой стабильных значений и их последовательности
- Можно выключать экономии электронов для.

24.3. Функциональное описание RNG

Рис. 232. Блок-схема



Аналоговый генератор подаёт начальное число ("посев") на регистр сдвига с линейной обратной связью (RNG_LFSR), с которого эти 32-бит числа и берут.

Начальное число создаётся по XOR от выходов нескольких кольцевых генераторов. RNG_LFSR тактируется постоянной частотой RNG_CLK и не зависит от HCLK. После подачи в RNG_LFSR весомого количества начальных чисел, его содержимое пишется в регистр данных (RNG_DR).

Параллельно отслеживаются "посев" и такты RNG_CLK. В битах состояния (регистр RNG_SR) указываются ненормальная последовательность "посева" и падение частоты RNG_CLK. При появлении ошибки может выдаваться прерывание.

24.3.1. Работа

Последовательность запуска RNG:

1. Можно разрешить прерывания (бит IE в регистре RNG_CR) по готовности генератора и по появлению ошибки.
2. Битом RNGEN в регистре RNG_CR включаем аналоговую часть, RNG_LFSR и детектор ошибок.
3. Если при входе в прерывание, в регистре RNG_SR биты SEIS=CEIS='0',бит DRDY='1', то регистр RNG_DR можно читать.

Стандарт FIPS PUB (Federal Information Processing Standard Publication) 140-2 требует, чтобы после установки бита RNGEN первое случайное число не использовалось, а сохранялось для сравнения со следующим, и далее парами. Одинаковых быть не должно.

24.3.2. Обработка ошибок

Бит CEIS = '1' (ошибка тактов)

Проверьте конфигурацию генератора тактов и снимите бит CEIS. RNG может работать при бите CECS = '0'. Ошибка тактов на предыдущее число не влияет и можно читать регистр RNG_DR.

Бит SEIS = '1' (ошибка "посева")

При бите SECS = '1' выдача чисел прекращается, потому что энтропия упала. Надо очистить бит SEIS, снять и поставить бит RNGEN.

24.4. Регистры RNG

Доступны словами.

24.4.1. Регистр управления (RNG_CR)

Смещение адреса: 0x00

По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												IE	RNGEN	Reserved	
												rw	rw		

— Биты 31:4 Резерв, не трогать.

— Бит 3 **IE**: Разрешение прерываний

0: Выкл.

1: Вкл. Нужны DRDY=1, SEIS=1 или CEIS=1 в регистре RNG_SR

— Бит 2 **RNGEN**: Включение RNG

0: Выкл.

1: Вкл.

— Биты 1:0 Резерв, не трогать.

24.4.2. Регистр состояния (RNG_SR)

Смещение адреса: 0x04

По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									SEIS	CEIS	Reserved		SECS	CECS	DRDY
									rc_w0	rc_w0			r	r	r

— Биты 31:7 Резерв, не трогать.

— Бит 6 **SEIS**: Состояние прерывания ошибки "посева"

Ставится одновременно с SECS, снимается записью 0.

0: Не было.

1: Обнаружено:

– Более 64 последовательных битов в одном значении (0 или 1)

– Более 32 последовательных чередований 0 и 1.

Прерывание выдаётся при IE = 1 в RNG_CR.

— Бит 5 **CEIS**: Состояние прерывания ошибки тактов

Ставится одновременно с CECS, снимается записью 0.

0: Всё норм.

1: RNG_CLK неверен ($f_{RNG\_CLK} < f_{HCLK}/16$).

Прерывание выдаётся при IE = 1 в RNG_CR.

— Биты 4:3 Резерв, не трогать.

— Бит 2 **SECS**: Состояние ошибки "посева"

0: Ошибок нет, если бит SEIS стоит, то ошибка была, но исправлена.

1: Обнаружено:

– Более 64 последовательных битов в одном значении (0 или 1)

– Более 32 последовательных чередований 0 и 1.

— Бит 1 **CECS**: Состояние ошибки тактов

0: Ошибок нет, если бит CEIS стоит, то ошибка была, но исправлена.

1: RNG_CLK неверен ($f_{RNG\_CLK} < f_{HCLK}/16$).

— Бит 0 **DRDY**: Данные готовы

0: RNG_DR не готов

1: RNG_DR готов к чтению нового числа

NB: Прерывание выдаётся при IE = 1 в RNG_CR.

После чтения RNG_DR этот бит снимается до появления нового числа.

24.4.3. Регистр данных (RNG_SR)

Смещение адреса: 0x08

По сбросу: 0x0000 0000.

Только чтение. Новое число появляется максимум через 40 тактов RNG_CLK. При этом встаёт бит DRDY. Пожалуйста, проверяйте сей факт.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RNDATA															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RNDATA															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

— Биты 31:0 **RNDATA**. То самое число.

24.4.4. Карта регистров RNG

Таблица 116. Они так лежат

Offset	Register name reset value	Register size																												
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	RNG_CR 0x0000000	Reserved																												IE	RNGEN	Reserved	
0x04	RNG_SR 0x0000000	Reserved																									SEIS	CEIS	Reserved		SECS	CECS	DRDY
0x08	RNG_DR 0x0000000	RNDATA[31:0]																															

25. Hash процессор (HASH)

25.1. Введение в HASH

Процессор полностью совместим с алгоритмами SHA-1, SHA-224, SHA-256, MD5 и HMAC. Он вычисляет дайджест (160 бит для SHA-1, 256 бит для SHA-256 и 224 бит для SHA-224,128 бит для MD5) для сообщений длиной до $(2^{64} – 1)$ бит, а HMAC даёт способ аутентификации сообщений с помощью hash-функций. Алгоритм HMAC состоит из двух вызовов hash-функций SHA-1, SHA-224, SHA-256 или MD5.

25.2. Основные свойства HASH

- Совместимость с:
  – FIPS PUB 180-2 (Federal Information Processing Standards Publication 180-2)
  – Secure Hash Standard specifications (SHA-1, SHA-224 and SHA-256)
  – IETF RFC 1321 (Internet Engineering Task Force Request For Comments number 1321) specifications (MD5)
- Быстрое вычисление SHA-1, SHA-224, SHA-256 и MD5 (SHA-224 и SHA-256 на STM32F43xxx)
- Ведомое устройство AHB
- 32-бит ввод, поддержка слов, полуслов, байтов и битовых строк толко в формате *little-endian*.
- Автоматический обмен для совместимости с *big-endian* SHA1, SHA-224 и SHA-256
- Автоматическое расширение битовых строк до модуля 512 (16 × 32 бита) при вычислении
- 5 × 32-бит слов (H0 до H5) на STM32F415/417xx и 8 × 32-бит слов (H0 до H7) на STM32F43xxx в выходном дайджесте, возможность перезапуска прерванных вычислений.
- Соответствующие 32-бит слова дайджеста из последовательных блоков сообщения добавляются друг к другу для получения дайджеста всего сообщения

- Поддержка DMA

NB: Расширение (см. алгоритмы SHA-1, SHA-224 и SHA-256, это добавление бита bx1 с последующими N битами bx0 для получения общей длины, конгруэнтной *448 modulo 512*. :-)) После сего сообщение завершается 64-бит целым с двоичным представлением исходной длины сообщения.

Ввод обрабатываемых сообщений только по 32 бита, включая последнюю порцию. Дополняйте и расширяйте.

25.3. Функциональное описание HASH

Рис. 233. Блок-схема для STM32F415/417xx

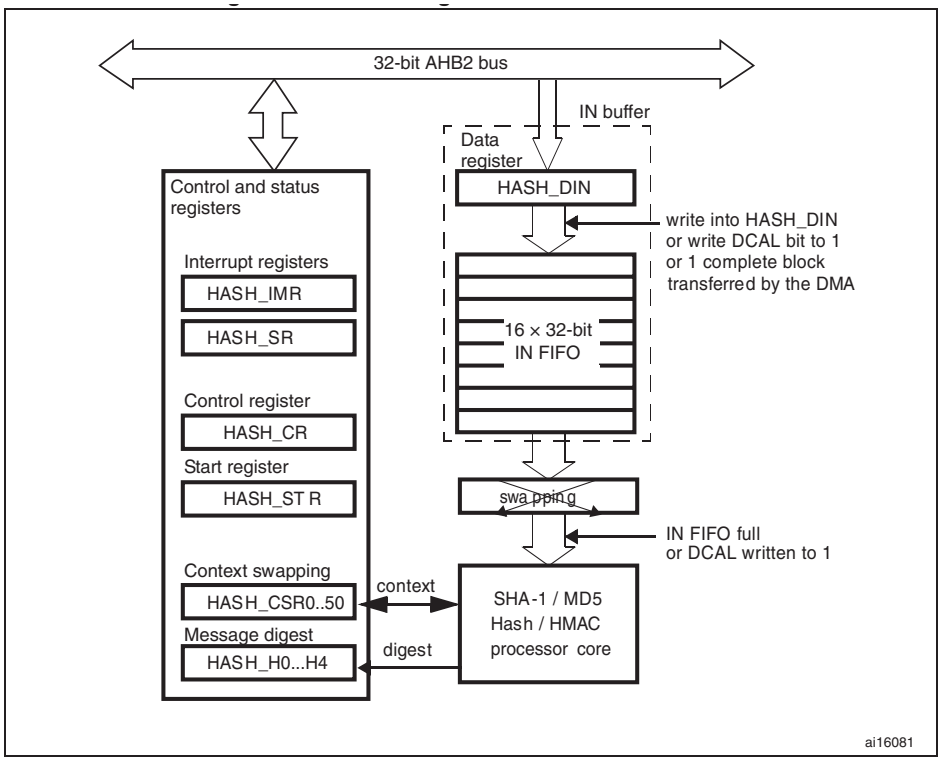
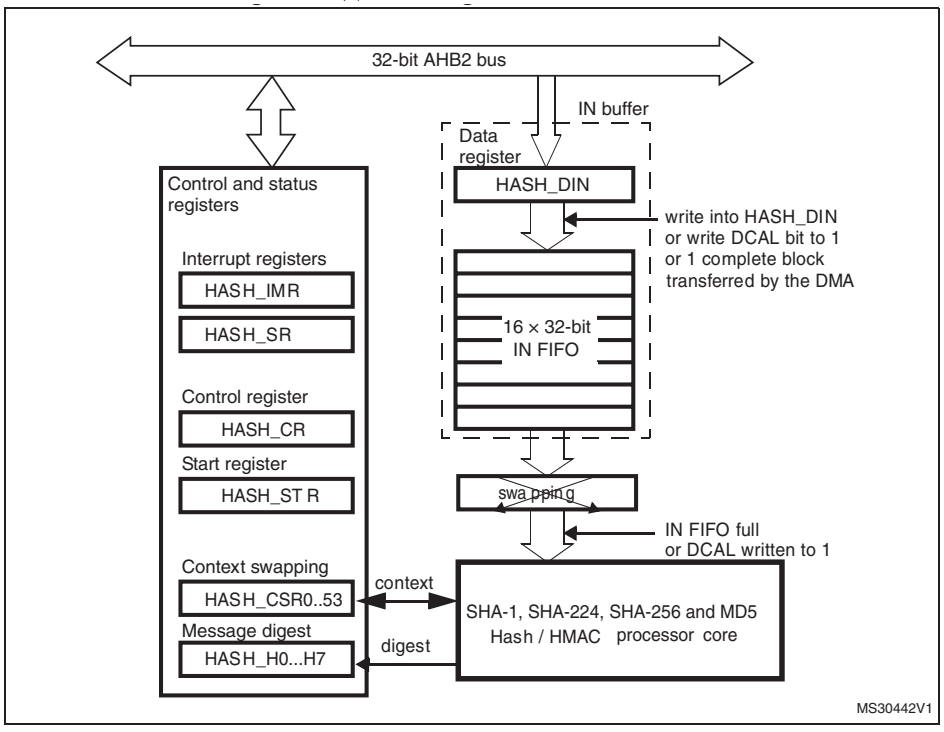


Рис. 234. Блок-схема для STM32F43xxx



Алгоритмы SHA-1, SHA- 224, SHA-256 и MD5 создают сжатое представление данных. Для строки короче 264 бит создаётся дайджест длиной 160, 224, 256 и 128 бит соответственно. Его можно использовать для создания и проверки сигнатуры.

Текущая реализация стандарта работает с *little-endian* форматом входных данных.

25.3.1. Длительность обработки

Вычисление промежуточного блока сообщения занимает:

- 66 HCLK тактов в SHA-1
- 50 HCLK тактов в SHA-224
- 50 HCLK тактов в SHA-256
- 50 HCLK **clock cycles in** MD5

к которому надо добавить время на загрузку 16 слов блока в процессор (не менее 16 тактов на один 512-бит блок).

В зависимости от длины последнего блока (или ключа в HMAC) время обработки может быть больше. По сравнению с промежуточным блоком его надо умножить на:

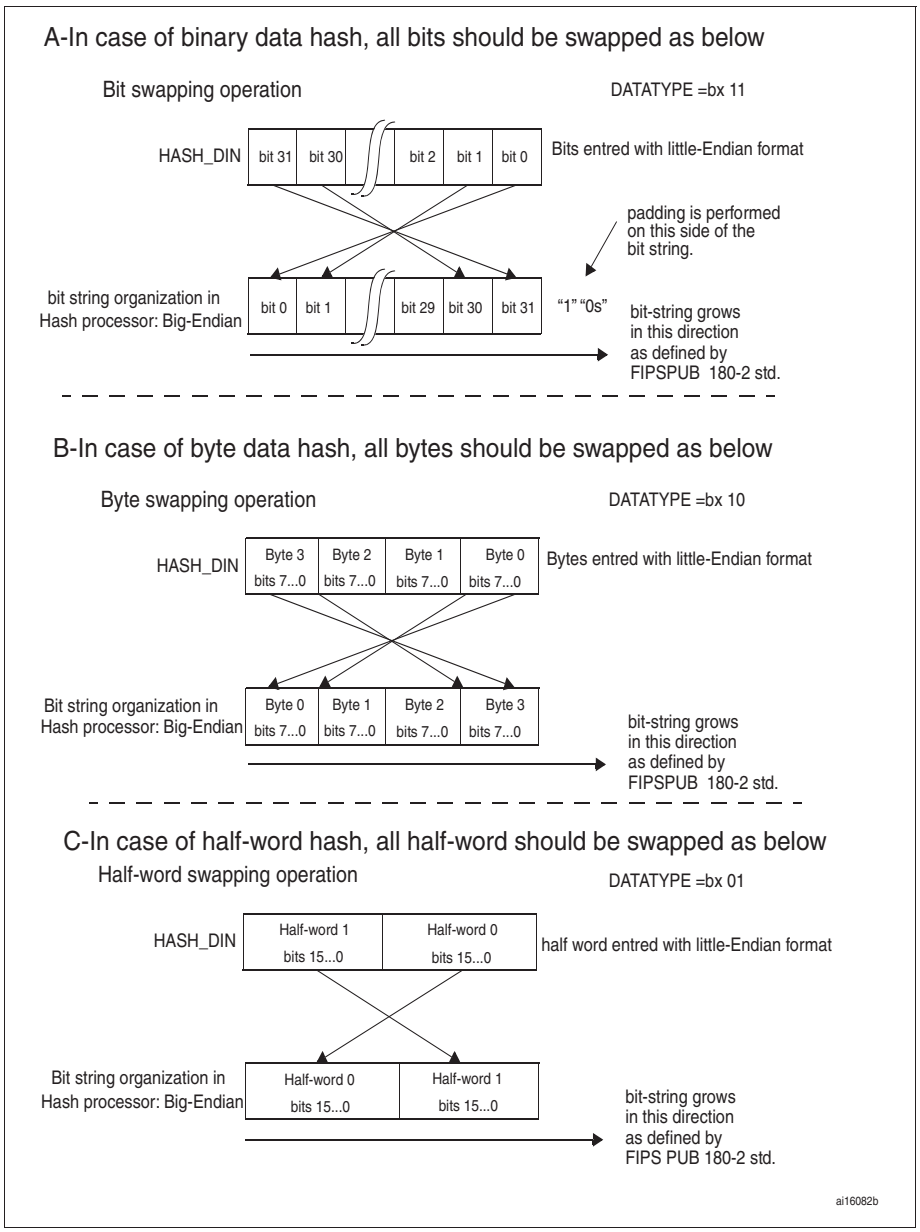
- 1 до 2.5 для hash-сообщения
- около 2.5 для входного ключа HMAC
- 1 до 2.5 для сообщения HMAC
- около 2.5 для выходного короткого ключа HMAC
- 3.5 до 5 для выходного длинного ключа HMAC

25.3.2. Типы данных

Данные идут в процессор по 32 бита за раз через регистр HASH_DIN. При этом они автоматически приводятся в формат *big-endian*.

Тип входных данных задают в поле DATATYPE регистра HASH_CR.

Рис. 235. Обмен битов, байтов и полуслов



25.3.3. Вычисление дайджеста сообщения

После получения в IN FIFO через регистр HASH_DIN полного блока 16 × 32-бит слов (= 512 бит), процессор начинает частичное вычисление дайджеста. HASH_DIN IN FIFO создают IN буфер из 17 слов.

С началом обработки процессору надо знать, есть ли в блоке последний бит сообщения или нет. Ситуаций две:

- Без использования DMA:
  - При частичном вычислении в регистр HASH_DIN пишут ещё одно слово (первое слово следующего блока) и перед записью нового слова ждут готовности процессора (DINIS=1).
  - Для последнего блока пишут бит DCAL=1 в регистре HASH_STR.
- При работе с DMA:

Содержимое HASH_DIN определяется автоматически по информации от контроллера DMA, равно как и расширение и пуск вычисления.

- При одинарных передачах: Для STM32F43xxx нужно снимать бит MDMAT. При передаче последнего блока бит DCAL встанет сам.
- При множественных передачах DMA (только на STM32F43xxx): надо программно ставить бит MDMAT, а для последней передачи и каждой фазы HMAC его надо снимать.

Процесс *—ввод данных + частичное вычисление—* продолжается до записи последнего бита исходного сообщения. В поле NBLW регистра HASH_STR надо иметь число битов в последнем слове сообщения (от 1 до 32), это надо для правильного расширения сообщения.

Установка DCAL = 1 в регистре HASH_STR пускает обработку сообщения:

- Автоматически расширяется сообщение для кратности 512 бит.
- HASH последовательно обрабатывает блоки по 512 бит
- Вычисляется финальный дайджест сообщения

25.3.4. Расширение сообщения

Сообщение расширяется добавлением “1” в последнее слово блока в позицию, заданную полем NBLW с заполнением остатка нулями.

Пример: посылаем ASCII строку “abc”, длиной L = 24:

```
 byte 0   byte 1   byte 2   byte 3
01100001 01100010 01100011 UUUUUUUU
<-- 1st word written to HASH_DIN -->    .
```

В NBLW надо писать число 24: бит “1” добавляется в эту позицию (счёт слева направо), что соответствует биту 31 регистра HASH_DIN (в *little-endian*):

```
01100001 01100010 01100011 1UUUUUUU
```

Теперь в строке 25 битов, и добавляются 423 “0”, получается 448. Получаем (*big-endian*):

```
61626380 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000
```

Добавляем длину L (00000000 00000018). Теперь в шестнадцатиричном:

```
61626380 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000018
```

При *little-endian* входном формате делаем так:

1. В регистр HASH_DIN пишем 0xUU636261 ('U’ - без разницы)
2. В регистр HASH_STR пишем 0x18 (число занятых битов в регистре HASH_DIN)
3. Записью 0x10 в HASH_STR пускаем расширение и вычисление дайджеста.
   Если NBLW ≠ 0x00, то “1” пишется в HASH_DIN в позицию NBLW и нули в [31:(NBLW+1)].
   Если NBLW == 0x00, то добавляется новое слово 0x0000 0001. Остаток блока заполняется нулями до получения длины 16 x 32-бит слов.

После вычисления по алгоритму SHA-1 дайджест доступен в регистрах HASH_Hx (x = 0…4). Как пример:

H0 = 0xA9993E36
H1 = 0x4706816A
H2 = 0xBA3E2571
H3 = 0x7850C26C
H4 = 0x9CD0D89D

25.3.5. Работа Hash

Хэш-функция (SHA-1, SHA-224, SHA-256 и MD5) выбирается при битах INIT=1 и MODE=0 в регистре HASH_CR. Алгоритм (SHA-1, SHA-224,SHA-256 или MD5) выбирается битами ALGO при стоящем бите INIT.

После записи в HASH_DIN первого слова следующего блока процессор начинает частичное вычисление дайджеста. Он остаётся занятым на 66 тактов для алгоритма the SHA-1 или 50 тактов для алгоритмов MD5, SHA-224 и SHA- 256. При работе без DMA процессору надо указывать последний блок сообщения.

На STM32F415/417xx вычисленный дайджест читают из регистров HASH_H0...HASH_H4 (для алгоритма MD5 регистр HASH_H4 не релевантен) на STM32F43xxx из HASH_H0...HASH_H7, где:

HASH_H4..HASH_H7 не релевантны для алгоритма MD5,

HASH_H5.. HASH_H7 для SHA-1,

HASH_H7 для SHA-224.

25.3.6. Работа HMAC

Алгоритм HMAC необратимо связывает сообщение с ключом пользователя.

В основном он состоит из двух вложенных hash-операций:

HMAC(message) = Hash[((key | pad) XOR 0x5C)
| Hash(((key | pad) XOR 0x36) | message)]

где:

- pad это последовательность нулей расширения ключа до длины блока hash-функции (512 бит)
- | это оператор конкатенации

HMAC вычисляется за 4 фазы:

1. Инициализация блока: Ставят биты INIT и MODE, и в битах ALGO задают алгоритм. Если ключ длиннее 64 байтов, то тут же ставят бит LKEY (тогда вместо реального ключа может быть использован его hash).
2. Ядру задают ключ для внутренней hash-функции. Как и для сообщений, данные пишут в регистр HASH_DIN и, наконец, в HASH_STR.
3. После записи последнего слова процессор обрабатывает ключ и изготавливается к приёму сообщения уже известным способом.
4. После первого hash-раунда процессор извещает о готовности принять ключ для внешней hash-функции (обычно он тот же самый). Вычисление начинается после ввода последнего слова ключа, результат HMAC появляется в регистрах HASH_H0...HASH_H4 на STM32F415/417xx и в регистрах HASH_H0...HASH_H7 на STM32F43xxx.

NB: Задержка вычисления примитива HMAC зависит от длины ключей и сообщения. HMAC возможен как две вложенные hash-функции с одинаковой длиной ключей (длинный или короткий).

25.3.7. Обмен контекстов

Он нужен при переключении задач ОС. Есть два способа: при использовании DMA для потока данных и без оного.

При программной загрузке данных

Сначала надо дождаться конца работы процессора: бит DINIS = 1 (всё сделано и IN FIFO пуст) или NBW ≠ 0 (FIFO не полон и работа не идёт).

- Сохранение контекста:
  Пишем в память регистры:
  - HASH_IMR
  - HASH_STR
  - HASH_CR
  - HASH_CSR0 — HASH_CSR50 на STM32F415/417xx, и HASH_CSR0 — HASH_CSR53 на STM32F43xxx.
- Восстановление контекста:
  Делаем так:
  a) Восстанавливаем из памяти регистры: HASH_IMR, HASH_STR и HASH_CR
  b) Инициализируем процессор установкой бита INIT в регистре HASH_CR
  c) Пишем из памяти регистры HASH_CSR0 — HASH_CSR50 (STM32F415/417xx) и HASH_CSR0 — HASH_CSR53 (STM32F43xxx)
- Можно пустить обработку с точки останова.

При использовании DMA

Сначала надо остановить передачи DMA и дождаться готовности процессора к прерыванию обработки.

- Прерывание обработки:
  - Выключаем интерфейс DMA битом DMAE
  - Ждём завершения передачи DMA (DMAES = 0 в HASH_SR). При этом процессору может быть передан не весь блок.
  - Выключаем соотвествующий канал контроллера DMA
  - Ждём готовности процессора (бит DINIS = 1)
- Сохраняют и восстанавливают контекст уже описанным способом.

Снова конфигурируем контроллер DMA для передачи конца сообщения. Можно пустить обработку с точки останова установкой бита DMAE.

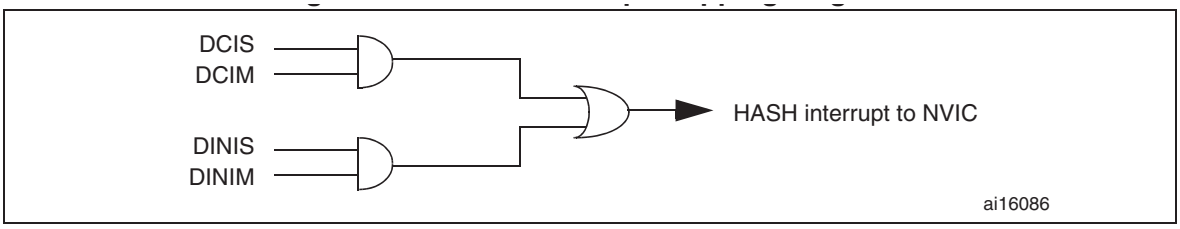
NB: Если обмен контекстов не затрагивает операции HMAC, то регистры HASH_CSR38 — HASH_CSR50 (STM32F415/417xx) и HASH_CSR38 — HASH_CSR53 (STM32F43xxx) можно не сохранять и восстанавливать.

При смене контекстов между двумя блоками (последний блок обработан, а новый ещё на запихнули в IN FIFO, NBW = 000 в регистре HASH_CR), регистры HASH_CSR22 — HASH_CSR37 можно не сохранять и восстанавливать.

25.3.8. Прерывания HASH

Два маскируемых прерывания подключены к одному вектору. Маски лежат в регистре HASH_IMR. Состояние прерываний читают из регистра HASH_SR.

Рис. 236. Прерывания HASH



25.4. Регистры HASH

Доступны словами.

25.4.1. Регистр управления (HASH_CR) для STM32F415/417xx

Смещение адреса: 0x00

По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															LKEY
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			DINNE	NBW				ALGO[0]	MODE	DATATYPE		DMAE	INIT	Reserved	
			r	r	r	r	r	rw	rw	rw	rw	rw	w		

— Биты 31:17 Резерв, не трогать.

— Бит 16 **LKEY**: Длинный ключ

0: Короткий (≤64 bytes)

1:Длинный (> 64 bytes)

NB: Учитывается только при битах INIT = MODE = 1.

— Биты 15:13 Резерв, не трогать.

— Бит 12 **DINNE**: DIN не пуст

Встаёт при наличии в HASH_DIN данного. Падает когда встаёт бит INIT (инициализация) или DCAL (конец обработки).

0: Пуст.

1: Не пуст.

— Биты 11:8 **NBW**: Число уже записанных в IN FIFO слов

Инкрементируется (+1) после записи в регистр HASH_DIN при DINNE = 1.

Обнуляется при подъёме бита INIT (инициализация) или пуске вычисления дайджеста (DCAL встаёт или DMA кончает передачу).

- При работе без DMA:

0000 и DINNE=0: регистр HASH_DIN и IN FIFO опустошены

0000 and DINNE=1: 1 слово в регистре HASH_DIN, IN FIFO пуст

0001: 2 слова писали (по одному в HASH_DIN и IN FIFO)

…

1111: 16 слов писано в буфер DIN

- При работе с DMA, NBW это точное число слов, засунутых в IN FIFO.

— Бит 7 **ALGO[0]**: Выбор алгоритма

0: SHA-1.

1: MD5.

NB: Учитывается только при битах INIT = MODE = 1.

— Бит 6 **MODE**: Выбор режима HASH или HMAC:

0: HASH

1: HMAC. Для ключей длиннее 64 байт бит LKEY должен стоять.

NB: Учитывается только при бите INIT = 1.

— Биты 5:4 **DATATYPE**: Тип данных в регистре HASH_DIN.

00: 32-бит. Из HASH_DIN в процессор идут напрямую.

01: 16-бит, полуслова. Полуслова из HASH_DIN обмениваются местами.

10: 8-бит, байты. При передаче из HASH_DIN обмениваются местами.

11: битовая строка. При передаче из HASH_DIN биты обмениваются местами (начало/конец).

— Бит 3 **DMAE**: Включение DMA

0: Выкл.

1: Вкл. Запрос DMA посылается при готовности ядра HASH принять данные.

NB:

1: Бит снимается аппаратно при передаче последнего данного сообщения, но не битом INIT=1.

2: При снятии бита во время передачи она всё таки завершается.

— Бит 2 **INIT**: Пуск вычисления дайджеста с предварительным сбросом процессора.

Запись 0 бессмысленна, читается как 0.

— Биты 1:0 Резерв, не трогать.

25.4.2. Регистр управления (HASH_CR) для STM32F43xxx

Смещение адреса: 0x00

По сбросу: 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													ALGO[1]	Reserved	LKEY
													rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		MDMAT	DINNE	NBW				ALGO[0]	MODE	DATATYPE		DMAE	INIT	Reserved	
		rw	r	r	r	r	r	rw	rw	rw	rw	rw	w		

— Биты 31:19 Резерв, не трогать.

— Бит 17 Резерв, не трогать.

— Бит 16 **LKEY**: Длинный ключ

0: Короткий (≤64 bytes)

1:Длинный (> 64 bytes)

NB: Учитывается только при битах INIT = MODE = 1.

— Биты 15:14 Резерв, не трогать.

— Бит 13 **MDMAT**: Множественные передачи DMA для большого числа данных

0: DCAL автоматически ставится по концу передачи DMA.

1: DCAL автоматически не ставится.

— Бит 12 **DINNE**: DIN не пуст

Встаёт при наличии в HASH_DIN данного. Падает когда встаёт бит INIT (инициализация) или DCAL (конец обработки).

0: Пуст.

1: Не пуст.

— Биты 11:8 **NBW**: Число уже записанных в IN FIFO слов

Инкрементируется (+1) после записи в регистр HASH_DIN при DINNE = 1.

Обнуляется при подъёме бита INIT (инициализация) или пуске вычисления дайджеста (DCAL встаёт или DMA кончает передачу).

- При работе без DMA:

0000 и DINNE=0: регистр HASH_DIN и IN FIFO опустошены

0000 and DINNE=1: 1 слово в регистре HASH_DIN, IN FIFO пуст

0001: 2 слова писали (по одному в HASH_DIN и IN FIFO)

…

1111: 16 слов писано в буфер DIN

- При работе с DMA, NBW это точное число слов, засунутых в IN FIFO.

— Бит 18 и 7 **ALGO[1:0]**: Выбор алгоритма

00: SHA-1.

01: MD5.

10: SHA-224.

11: SHA-256.

NB: Учитывается только при битах INIT = MODE = 1.

— Бит 6 **MODE**: Выбор режима HASH или HMAC:

0: HASH

1: HMAC. Для ключей длиннее 64 байт бит LKEY должен стоять.

NB: Учитывается только при бите INIT = 1.

— Биты 5:4 **DATATYPE**: Тип данных в регистре HASH_DIN.

00: 32-бит. Из HASH_DIN в процессор идут напрямую.

01: 16-бит, полуслова. Полуслова из HASH_DIN обмениваются местами.

10: 8-бит, байты. При передаче из HASH_DIN обмениваются местами.

11: битовая строка. При передаче из HASH_DIN биты обмениваются местами (начало/конец).

— Бит 3 **DMAE**: Включение DMA

0: Выкл.

1: Вкл. При готовности ядра HASH принять данные посылается запрос DMA.

NB: 1: Бит снимается аппаратно при передаче последнего данного сообщения, но не битом INIT=1.

2: При снятии бита во время передачи она всё таки завершается.

— Бит 2 **INIT**: Пуск вычисления дайджеста с предварительным сбросом процессора. Запись 0 бессмысленна, читается как 0.

— Биты 1:0 Резерв, не трогать.

25.4.3. Регистр ввода данных (HASH_DIN)

Смещение адреса: 0x04

По сбросу: 0x0000 0000.

32-бит регистр для ввода сообщений блоками по 512 бит. При записи в HASH_DIN прежнее число впихивается в IN FIFO, а регистр поучает новой данное с шины AHB. Биты DATATYPE регистра HASH_CR уже должны показывать тип данных.

После записи блока из 16 слов пускается промежуточное вычисление дайджеста:

- записью в HASH_DIN первого слова нового блока (при работе без DMA)
- при работе с DMA автоматически

После записи последнего блока сообщения пускается конечное вычисление дайджеста (включая расширение):

- установкой бита DCAL в регистре HASH_STR
- при работе с DMA и бите MDMAT = ‘0’ автоматически =.

При записи в HASH_DIN во время вычисления дайджеста на шине AHB добавляются состояния ожидания вплоть до конца работы HASH.

При чтении из HASH_DIN выдаётся последнее записанное слово по этому адресу (ноль после сброса).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATAIN															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAIN															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:0 **DATAIN**. То самое число.

25.4.4. Регистр старта (HASH_STR)

Смещение адреса: 0x08

По сбросу: 0x0000 0000.

Имеет две функции:

- Задаёт число действительных битов в последнем слове последнего блока сообщения (номер первого свободного)
- Пускает обработку последнего блока установкой бита DCAL

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							DCAL	Reserved			NBLW				
							w				rw	rw	rw	rw	rw

— Биты 31:9 Резерв, не трогать.

— Бит 8 **DCAL**: Запуск обработки последнего блока

— Биты 7:5 Резерв, не трогать.

— Биты 4:0 **NBLW**: Число действительных битов в последнем слове последнего блока сообщения (номер первого свободного, начиная со старшего). Писать надо до установки бита DCAL.

25.4.5. Регистры дайджеста (HASH_HR0..4/5/6/7)

Смещение адреса: 0x0C до 0x1C (STM32F415/417xx), плюс 0x310 до 0x32C (STM32F43xxx)

По сбросу: 0x0000 0000.

Содержат результат дайджеста сообщения с именами в описаниях алгоритмов:

1. H0, H1, H2, H3 и H4, при алгоритме SHA1, HASH_H5 — HASH_H7 не используются и читаются нулями.
2. A, B, C и D, при алгоритме MD5, HASH_H4 — HASH_H7 не используются и читаются нулями.
3. H0 — H6, при алгоритме SHA224, HASH_H7 не используется и читается нулями.
4. H0 — H7, при алгоритме SHA256

Чтение регистров во время работы процессора (DCAL=1), задерживает чтение вплоть до завершения вычислений.

H0, H1, H2, H3 и H4 отображены в двух регионах.

Регистр	Смещение адреса
HASH_HR0	0x0C и 0x310
HASH_HR1	0x10 и 0x314
HASH_HR2	0x14 и 0x318
HASH_HR3	0x18 и 0x31C
HASH_HR4	0x1C и 0x320
HASH_HR5	0x324
HASH_HR6	0x328
HASH_HR7	0x32C

NB: При запуске вычислений нового потока эти регистры сбрасываются.

25.4.6. Регистр разрешения прерываний (HASH_IMR)

Смещение адреса: 0x20

По сбросу: 0x0000 0000.

'1' в бите разрешает прерывание.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														DCIE	DINIE
														rw	rw

— Биты 31:2 Резерв, не трогать.

— Бит 1 **DCIE**: Разрешение прерывания конца вычислений

— Бит 0 **DINIE**: Разрешение прерывания ввода данных

25.4.7. Регистр состояния (HASH_IMR)

Смещение адреса: 0x24

По сбросу: 0x0000 0001.

'1' в бите - работает (запрос прерывания есть).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												BUSY	DMAS	DCIS	DINIS
												r	r	rc_w0	rc_w0

— Биты 31:4 Резерв, не трогать.

— Бит 3 **BUSY**: Процессор занят вычислениями

— Бит 2 **DMAS**: Состояние DMA

— Бит 1 **DCIS**: Состояние прерывания конца вычислений

— Бит 0 **DINIS**: Состояние прерывания ввода данных

25.4.8. Регистр смены контекстов (HASH_CSRx)

Смещение адреса: 0x0F8 до 0x1C0

- для HASH_CSR0: По сбросу 0x0000 0002.
- для других: 0x0000 0000, у STM32F43xxx регистр HASH_CSR2 по сбросу 0x2000 0000

Дополнительные регистры на STM32F43xxx: смещения от 0x1C1 to 0x1CC , сброс: 0x0000 0000.

В регистрах HASH_CSRx есть состояния всех внутренних регистров процессора.

HASH_CSRx

Смещение адреса: 0x0F8 до 0x1C0 на STM32F415/417xx

Смещение адреса: 0x0F8 до 0x1CC на STM32F43xxx

25.4.9. Карта регистров HASH

Таблица 117. Регистры HASH на STM32F415/417xx

Offset	Register name reset value	Register size																													
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	HASH_CR	Reserved															LKEY	Reserved			DINNE	NBW				ALGO[0]	MODE	DATATYPE		DMAE	INIT	Reserved	
	Reset value																0				0	0	0	0	0	0	0	0	0	0	0		
0x04	HASH_DIN	DATAIN																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	HASH_STR	Reserved																							DCAL	Reserved			NBLW				
	Reset value																								0				0	0	0	0	0
0x0C	HASH_HR0	H0																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	HASH_HR1	H1																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	HASH_HR2	H2																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	HASH_HR3	H3																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	HASH_HR4	H4																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	HASH_IMR	Reserved																													DCIE	DINIE	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	HASH_SR	Reserved																												BUSY	DMAS	DCIS	DINIS
	Reset value																													0	0	0	1
0xF8	HASH_CSR0	CSR0																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
	...																																
	...																																
0x1C0	HASH_CSR50	CSR50																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Reserved																																
0x310	HASH_HR0	H0																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x314	HASH_HR1	H1																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x318	HASH_HR2	H2																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x31C	HASH_HR3	H3																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x320	HASH_HR4	H4																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Таблица 118. Регистры HASH на STM32F43xxx

Offset	Register name reset value	Register size																													
		31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	HASH_CR	Reserved													ALGO[1]	Reserved	LKEY	Reserved		MDMAT	DINNE	NBW				ALGO[0]	MODE	DATATYPE		DMAE	INIT	Reserved	
	Reset value														0		0			0	0	0	0	0	0	0	0	0	0	0		
0x04	HASH_DIN	DATAIN																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	HASH_STR	Reserved																							DCAL	Reserved			NBLW				
	Reset value																								0				0	0	0	0	0
0x0C	HASH_HR0	H0																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	HASH_HR1	H1																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	HASH_HR2	H2																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	HASH_HR3	H3																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	HASH_HR4	H4																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	HASH_IMR	Reserved																													DCIE	DINIE	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	HASH_SR	Reserved																												BUSY	DMAS	DCIS	DINIS
	Reset value																													0	0	0	1
0xF8	HASH_CSR0	CSR0																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
...																																	
...																																	
0x1CC	HASH_CSR53	CSR53																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reserved																																	
0x310	HASH_HR0	H0																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x314	HASH_HR1	H1																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x318	HASH_HR2	H2																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x31C	HASH_HR3	H3																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x320	HASH_HR4	H4																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x324	HASH_HR5	H5																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x328	HASH_HR6	H6																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x32C	HASH_HR7	H7																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

26. Таймер реального времени (RTC)

26.1. Введение в RTC

Это независимый BCD таймер/счётчик. Он считает время суток/дату, имеет два программируемых прерывания будильников и периодический флаг побудки с прерыванием и блок управления экономными режимами.

Два 32-бит регистра содержат секунды, минуты, часы (12- или 24-часовом формате), день недели, день месяца, месяц и год в BCD формате. В двоичном формате есть суб-секундные данные. Учёт високосных годов и количества дней в месяцах автоматический, есть учёт светлого времени суток.

Дополнительные 32-бит регистры будильников содержат суб-секунды, секунды, минуты, часы, день и дату.

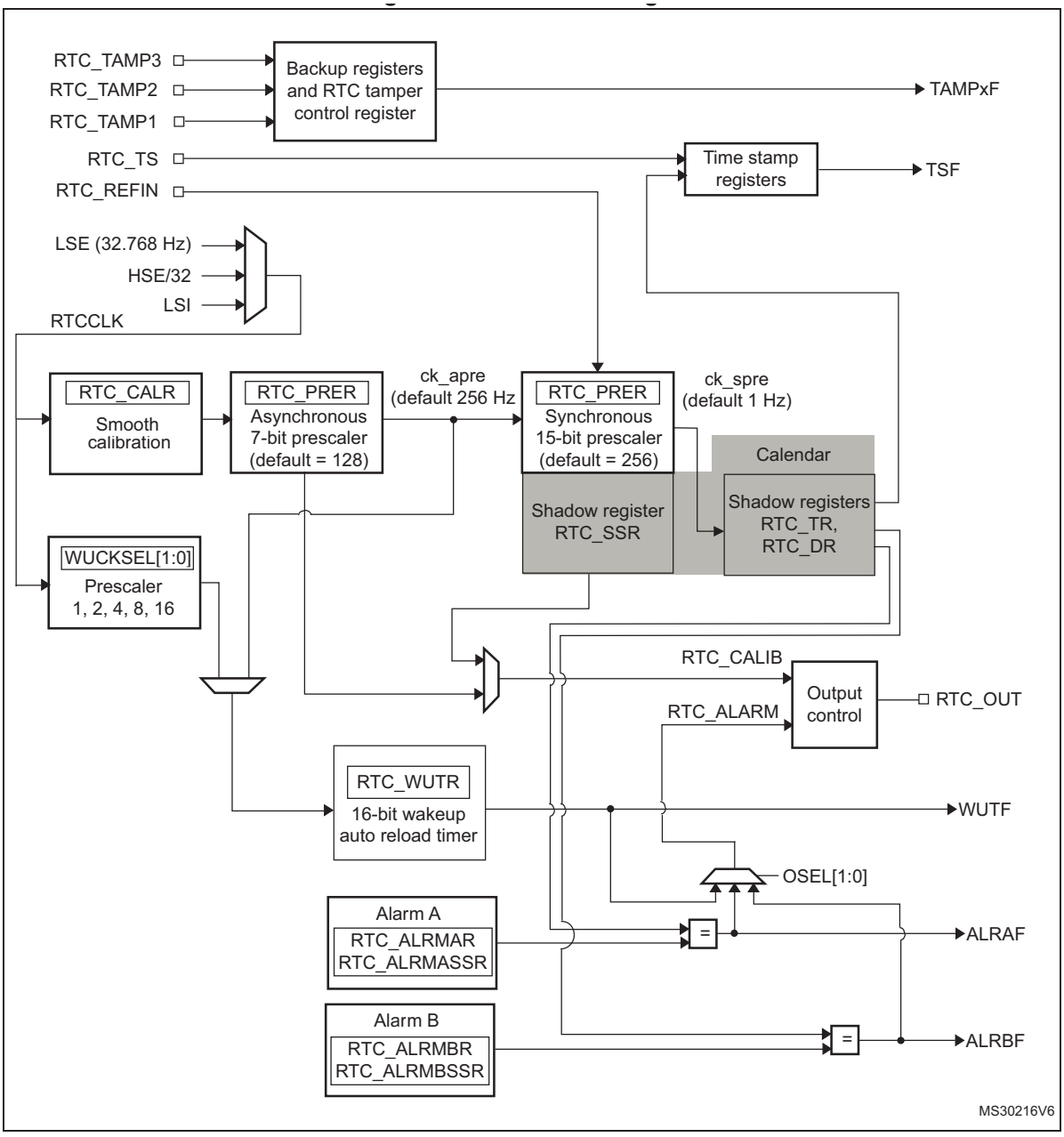
Есть цифровая калибрация для компенсации девиации точности кварцевой генерации. После сброса резервного домена все регистры RTC защищены от записи. Пока питание остаётся в норме RTC не стопорится, независимо от режима устройства.

26.2. Основные свойства RTC

Они такие:

- Календарь: субсекунды, секунды, минуты, часы (12 / 24), день недели, число, месяц и год.
- Программируемое летнее время.
- Два будильника с прерываниями и запуском по календарю.
- Блок автоматической побудки с периодическим флагом и прерыванием.
- Детектор опорной частоты: второй источник (50 или 60 Hz) для лучшей точности календаря.
- Точная синхронизация с внешними тактами по сдвигу субсекунд.
- Маскируемые прерывания/события:
  - Alarm A
  - Alarm B
  - Прерывание побудки
  - Метки времени
  - Обнаружение вмешательства
- Схема цифровой калибровки (периодическая коррекция счётчика)
  - точность 5 ppm (импульс/минуту)
  - точность 0.95 ppm, в окне калибровки (несколько секунд)
- Функция меток времени для сохранения событий (1 событие)
- Обнаружение вмешательства:
  - 2 события с конфигурируемым фильтром и внутренней подпоркой.
- 20 резервных регистров (80 байт). При вмешательстве они сбрасываются.
- Вывод одной из двух альтернативных функций (RTC_OUT):
  - RTC_CALIB: такты 512 Hz или 1 Hz с (при частоте LSE = 32.768 kHz). Разрешается битом `COE` регистра `RTC_CR`. Приведена к функции RTC_AF1 устройства.
  - RTC_ALARM (Alarm A, Alarm B или побудка). Выбирается битами `OSEL`[1:0] регистра `RTC_CR` register. Приведена к функции RTC_AF1 устройства.
- Входы альтернативных функций RTC:
  - RTC_TS: обнаружение меток времени. Приведён к функциям RTC_AF1 и RTC_AF2.
  - RTC_TAMP1: Обнаружение TAMPER1. Приведён к функциям RTC_AF1 и RTC_AF2.
  - RTC_TAMP2: Обнаружение события TAMPER2.
  - RTC_REFIN: вход опорных тактов (обычно сетевая, 50 или 60 Hz).

Рис. 237. Блок-схема RTC



1. На STM32F4xx функции RTC_AF1 RTC_AF2 подключены к PC13 и PI8, соответственно.

26.3. Функциональное описание RTC

26.3.1. Такты и предделители

Такты RTC (RTCCLK) выбирают из:

- тактов LSE,
- тактов генератора LSI,
- тактов HSE.

На календарь подаются такты после предделителя (точнее их 2):

- Асинхронный 7-бит, конфигурация через биты `PREDIV_A` регистра `RTC_PRER`.
- Синхронный 15-бит, конфигурация через биты `PREDIV_S` регистра `RTC_PRER`.

NB: При работе обоих предделителей экономнее будет писать в асинхронный предделитель число поболее.

При частоте LSE в 32.768 kHz асинхронный предделитель ставят на 128, а синхронный на 256, итог - внутренняя частота 1 Hz (ck_spre).

Минимальный делитель 1, максимальный 222. Это соответствует максимальной входной частоте около 4 MHz.

f_{ck_apre} получают по формуле:

$$f_{CK\_APRE} = \frac{f_{RTCCLK}}{PREDIV\_A + 1}$$

Такты ck_apre идут на двоичный обратный счётчик субсекунд RTC_SSR. После обнуления он перегружается значением из PREDIV_S.

f_{ck_apre} получают по формуле:

$$f_{CK\_SPRE} = \frac{f_{RTCCLK}}{(PREDIV\_S + 1) \times (PREDIV\_A + 1)}$$

Такты ck_spre можно использовать для календаря или как базу для 16-базу таймера побудки. Последний также можно кормить от RTCCLK с асинхронным 4-бит предделителем.

26.3.2. Часы реального времени и календарь

Регистры календарного времени и даты RTC читают через теневые регистры, синхронизируемые от PCLK1 (такты APB1). Но напрямую читать будет быстрее.

- RTC_SSR — субсекунды
- RTC_TR — время
- RTC_DR — дату

Каждые два периода RTCCLK текущие значения календаря копируются в теневые регистры и ставится бит RSF в регистре RTC_ISR (см. Секцию 26.6.4). В режимах Stop и Standby ничего не копируется. При выходе из них регистры обновляются через 2 периода RTCCLK.

Установка бита BYPSHAD в регистре RTC_CR переключает чтение данных непосредственно из регистров календаря. обходя теневые регистры. По умолчанию этот бит сброшен.

При чтении регистров RTC_SSR, RTC_TR и RTC_DR при BYPSHAD=0, частота APB (f_{APB}) должна быть в 7 раб выше частоты тактов RTC (f_{RTCCLK})

Системный сброс чистит теневые регистры.

26.3.3. Будильники

Их два, Alarm A и Alarm B. Включаются они битами ALRAIE и ALRBIE регистра RTC_CR. При совпадении времени и дат в регистрах RTC_ALRMASSR/RTC_ALRMAR и RTC_ALRMBSSR/RTC_ALRMBR встают флаги ALRAF и ALRBF. Все поля календаря можно маскировать битами MSKx в регистрах RTC_ALRMAR и RTC_ALRMBR, и битами MASKSSx в регистрах RTC_ALRMASSR и RTC_ALRMBSSR. Прерывания разрешаются битами ALRAIE и ALRBIE в регистре RTC_CR.

Alarm A и Alarm B (если разрешён битами OSEL[1:0] в регистре RTC_CR) можно направить на выход RTC_ALARM. Полярность RTC_ALARM задают битом POL регистра RTC_CR.

Увага: В случае выбора секунд (снят бит MSK0 в RTC_ALRMAR или RTC_ALRMBR) синхронный предделитель в регистре RTC_PRER должен быть не меньше 3.

26.3.4. Периодическая автопобудка

16-бит периодический обратный счётчик побудки можно расширить до 17 бит. Есть свой флаг.

Побудка разрешается битом WUTE в регистре RTC_CR.

На вход можно подавать:

- Такты RTC (RTCCLK), делённые на 2, 4, 8 или 16. Если RTCCLK это LSE(32.768kHz), то период можно задавать от 122 µs до 32 s, с разрешением 61µs.
- ck_spre (обычно внутренние такты 1 Hz). При ck_spre равной 1Hz можно задать период от 1 s до около 36 часов с разрешением 1 s. Этот диапазон делится на две части:
  - от 1s до 18 часов при WUCKSEL[2:1] = 10
  - от 18h до 36h при WUCKSEL[2:1] = 11. В этом случае к текущему значению 16-бит счётчика добавляется 216. Таймер начинает счёт по завершению последовательности инициализации. Включённая побудка работает всегда. При достижении 0 ставится флаг WUTF в регистре RTC_ISR и счётчик перегружается из регистра RTC_WUTR.

Флаг WUTF надо снимать программно.

Прерывание побудки разрешается битом WUTIE в регистре RTC_CR2, оно может выводить устройство из экономных режимов.

Флаг побудки можно направить на выход RTC_ALARM битами OSEL[1:0] регистра RTC_CR. Полярность RTC_ALARM задают битом POL регистра RTC_CR.

Ни системный сброс, ни экономные режимы на таймер побудки не влияют.

26.3.5. Инициализация и конфигурация RTC

Доступ к регистрам RTC

При обращении к регистрам RTC интерфейс APB вставляет 2 состояния ожидания, кроме чтения регистров календаря с битом BYPSHAD=0.

Защита записи регистров RTC

После системного сброса регистры RTC защищены от паразитной записи. Разрешают запись установкой бита DBP в регистре PWR_CR.

После сброса резервного домена все регистры RTC защищены от записи. Разрешают её записью ключа в регистр RTC_WPR. Разрешают запись в регистры RTC (кроме RTC_ISR[13:8], RTC_TAFCR, и RTC_BKPxR) так:

1. Пишут '0xCA' в регистр RTC_WPR.
2. Пишут '0x53' в регистр RTC_WPR.

Любая другая запись включает защиту. Системный сброс на этот механизм не влияет.

Инициализация и конфигурация календаря

Для установки начальных значений даты, времени и всего другого надо:

1. Включаем режим инициализации установкой бита INIT в регистре RTC_ISR. Счётчик календаря стопорится, можно писать.
2. Ждём подъёма бита INITF регистра RTC_ISR. Это занимает от 1 до 2 тактов RTCCLK (Синхронизация-с!).
3. Для выдачи 1 Hz тактов счётчика календаря в регистре RTC_PRER задаём сначала синхронный предделитель, затем асинхронный двумя раздельными операциями записи.
4. В теневые регистры (RTC_TR и RTC_DR) пишем начальное значение времени и даты, битом FMTв регистре RTC_CR задаём формат времени (12 или 24 часа).
5. Выключаем режим снятием бита INIT. Установки начнут работать через 4 такта RTCCLK.

Календарь начинает считать.

NB: После системного сброса по флагу INITS в регистре RTC_ISR можно узнать, работает календарь или нет. Если 0, то календарь не инициализирован (0 в поле года в его резервном домене).

Для чтения календаря после инициализации сначала надо поставить флаг RSF регистра RTC_ISR.

Летнее время

Добавить или отнять час у календаря можно битами SUB1H, ADD1H в регистре RTC_CR.

Бит BKP позволяет запомнить сиё действо.

Установка будильников

Процедура для обоих будильников (Alarm A и Alarm B):

1. Выключаем будильник чисткой бита ALRAE или ALRBIE в RTC_CR.
2. Ждём разрешения доступа к регистрам будильника по флагу ALRAWF или ALRBWF в регистре RTC_ISR. Это занимает от 1 до 2 тактов RTCCLK (Синхронизация, батенька-с!).
3. Пишем регистры Alarm (RTC_ALRMASSR/RTC_ALRMAR или RTC_ALRMBSSR/RTC_ALRMBR).
4. Включаем будильник установкой бита ALRAE or ALRBIE регистра RTC_CR.

NB: Каждое изменение регистра RTC_CR вступает в свои права после 1 или 2 тактов RTCCLK (Синхронизировать-то надо!).

Таймер побудки

Меняем самоперезагружаемое значение счётчика (WUT[15:0] в регистре RTC_WUTR):

1. Выключаем таймер чисткой бита WUTE в регистре RTC_CR.

2. Ждём установки бита WUTWF в регистре RTC_ISR для разрешения доступа к значению счётчика и битам WUCKSEL[2:0]. Это занимает от 1 до 2 тактов RTCCLK.
3. Пишем перегружаемое значение WUT[15:0] и выбор тактов (биты WUCKSEL[2:0] в регистре RTC_CR). Включаем таймер установкой бита WUTE в регистре RTC_CR. Отсчёт начинается сызнова. Бит WUTWF снимается через 2 такта RTCCLK после снятия бит WUTE.

26.3.6. Чтение календаря

Бит BYPSHAD регистра RTC_CR чист

При чтении регистров RTC_SSR, RTC_TR и RTC_DR частота тактов APB1 (f_{PCLK1}) должна быть равна или больше семикратной частоте тактов RTC (f_{RTCCLK}). Иначе регистры даты и времени надо читать несколько раз и добиваться одного и того же результата.

Бит RSF регистра RTC_ISR ставится каждые два такта RTCCLK при копировании регистров календаря в теневые регистры RTC_SSR, RTC_TR и RTC_DR. Чтение из регистра RTC_SSR или RTC_TR фиксирует значения в регистрах высшего порядка вплоть до чтения из регистра RTC_DR. При очередном обращении к календарю за период времени, меньший 2 RTCCLK: после первого чтения календаря бит RSF нужно программно снять, дождаться его подъёма и затем снова читать регистры RTC_SSR, RTC_TR и RTC_DR.

После выхода из экономных режимов (Stop или Standby), RSF надо снять программно и перед чтением регистров RTC_SSR, RTC_TR и RTC_DR дождаться его подъёма.

Чистить RSF надо после побудки, но не перед входом в экономный режим.

NB: После системного сброса, инициализации и синхронизации перед чтением RTC_SSR, RTC_TR and RTC_DR надо дождаться бита RSF (регистры сброшены).

Бит BYPSHAD регистра RTC_CR стоит (обход теневых регистров)

Регистры календаря читаются непосредственно и ждать бита RSF не надо. Это весьма полезно при выходе из экономных режимов (STOP или Standby), теневые регистры в это время не обновляются.

Из-за того, что во время чтения нескольких регистров календаря может попасть фронт тактового сигнала RTCCLK, регистры надо читать несколько раз и добиваться одного и того же результата.

NB: При BYPSHAD=1 команда чтения регистра требует дополнительного такта APB.

26.3.7. Сброс RTC

Теневые регистры (RTC_SSR, RTC_TR и RTC_DR) и некоторые биты регистра RTC_ISR приводятся в исходное состояние всеми доступными видами сброса.

Сбросом резервного домена, но не системным, сбрасываются: рабочие регистры календаря RTC, RTC_CR, RTC_PRER, RTC_CALIBR или RTC_CALR, RTC_SHIFTR, RTC_TSSSR, RTC_TSTR и RTC_TSDR, RTC_TAFCR, RTC_BKPxR, RTC_WUTR, RTC_ALRMASSR/RTC_ALRMAR и RTC_ALRMBSSR/RTC_ALRMBR.

При тактировании RTC от LSE, он продолжает работать под системным сбросом, если его источник отличен от источника сброса резервного домена.

По сбросу резервного домена RTC стопорится и его регистры сбрасываются.

26.3.8. Синхронизация RTC

RTC можно с высокой точностью синхронизировать с внешними тактами. По полю субсекунд (RTC_SSR или RTC_TSSSR), вычисляют точный сдвиг между внешними тактами и RTC и уменьшают его сдвигая такты RTC на доли секунды с помощью RTC_SHIFTR.

RTC_SSR это счётчик синхронного предделителя. Он позволяет вычислять время RTC с точностью до 1/(PREDIV_S+1) секунды. Максимальное разрешение (30.52 μs с тактами 32768 Hz) достигается при PREDIV_S равном 0x7FFF.

Но для сохранения частоты 1 Hz с увеличением PREDIV_S надо уменьшать PREDIV_A, а это может увеличить расход электронов на выходе асинхронного предделителя и самого RTC.

Точно RTC можно подстроить с помощью регистра сдвига (RTC_SHIFTR). Он смещает импульс (вперёд или назад) вплоть до секунды с точностью 1/(PREDIV_S+1) секунды.

Операция сдвига состоит из добавления значения SUBFS[14:0] к счётчику синхронного предделителя SS[15:0]: это задержит такт. Если при этом стоит бит ADD1S, то добавится одна секунда и, в результате вычтется доля секунды, такт продвинется вперёд.

Внимание: Перед запуском операции сдвига проверьте SS[15] = 0, может быть переполнение.

После записи в регистр RTC_SHIFTR аппаратно ставится флаг SHPF, указывая, что сдвиг ждёт. Он аппаратно снимается после завершения операции сдвига.

Внимание: Синхронизация несовместима с обнаружением опорных тактов: нельзя писать в регистр RTC_SHIFTR при REFCKON=1.

26.3.9. Детектор опорных тактов

Обновление календаря RTC можно синхронизировать с опорными тактами RTC_REFIN, обычно сетевыми (50 или 60 Hz). Точность RTC_REFIN должна быть выше, чем у 32.768 kHz LSE. Если детектор RTC_REFIN включён (бит REFCKON в RTC_CR стоит), то календарь тактируется от LSE, а RTC_REFIN используется для компенсации неточности частоты обновления календаря (1 Hz).

Каждый такт 1 Hz сравнивается с ближайшим фронтом опорных тактов (если он есть в заданном временном окне). В большинстве случаев, оба фронта выравнены верно. При нарушении выравнивания тактов 1 Hz из-за погрешности LSE, RTC чуть смещает такты 1 Hz и их будущие фронты выравниваются.

Наличие опорных тактов обнаруживается с помощью 256 Hz тактов (ck_apre), полученных от кварца 32.768 kHz, во временном окне около обновления календаря. Для первого такта окно имеет ширину 7 ck_apre и 3 ck_apre для последующих обновлений календаря.

При обнаружении опорного такта в окне перегружается синхронный предделитель, который и выдаёт ck_spre, и убежавшие такты 1 Hz чуть-чуть смещаются.

Если фронт опорного такта не появился в окне 3 ck_apre, то календарь обновляется от тактов LSE и RTC ждёт его в окне шириной 7 ck_apre с центром на фронте ck_spre.

При работе детектора тактов PREDIV_A и PREDIV_S должны стоять по умолчанию:

- PREDIV_A = 0x007F
- PREDIV_S = 0x00FF

NB: В режиме Standby детектор недоступен.

Внимание: Детектор недоступен при грубой цифровой калибровке: при REFCKON=1 RTC_CALIBR должен быть равен 0x0000 0000.

26.3.10.Грубая цифровая калибровка

Есть два метода калибровки: грубая и тонкая. Вместе их использовать нельзя.

Грубая калибровка компенсирует неточность кварца добавлением (позитивная) или маскированием (негативная) тактов на выходе асинхронного предделителя (ck_apre).

Тип калибровки выбирается битом DCS в регистре RTC_CALIBR.

При позитивной (DCS=0), каждую минуту (около 15360 тактов ck_apre) с интервалом 2xDC минут добавляются 2 такта ck_apre.

При негативной (DCS=1), каждую минуту (около 15360 тактов ck_apre) с интервалом 2xDC минут удаляется 1 ck_apre.

DC (от 0 до 31) задаётся битами DC[4:0] регистра RTC_CALIBR.

Грубую калибровку конфигурируют только в режиме инициализации, работать она начинает при снятии бита INIT. Полный цикл калибровки длится 64 минуты. Первые 2xDC минуты 64-минутного цикла всё меняется как описано.

Максимальный диапазон калибровки от –63 до 126 импульсов в минуту (ppm) с точностью 2 ppm для негативной и около 4 ppm для позитивной.

Калибровать можно такты от LSE и HSE.

Внимание: Калибровка брешет при PREDIV_A < 6.

26.3.11.Тонкая цифровая калибровка

Добавляя или удаляя отдельные импульсы RTCCLK, частоту RTC можно калибровать с разрешением около 0.954 ppm в диапазоне от -487.1 ppm до +488.5 ppm.

Цикл тонкой калибровки равен около 2^{20} импульсов RTCCLK или 32 секунды при входной частоте 32768 Hz. Он обеспечивается 20-бит счётчиком, cal_cnt[19:0], с тактами от RTCCLK.

Биты CALM[8:0] регистра RTC_CALR задают число маскируемых тактов RTCCLK за 32-секундный период:

- CALM[0] — 1 импульс в момент cal_cnt[19:0]=0x80000.
- CALM[1] — ещё 2 импульса в моменты cal_cnt[19:0]=0x40000 и 0xC0000
- CALM[2] — ещё 4 импульса в моменты cal_cnt[19:0]=0x20000/0x60000/0xA0000/0xE0000
- и т.д. до - CALM[8] — 256 импульсов в моменты cal_cnt=0xXX800.

Установка бита CALP добавляет один импульс RTCCLK на каждые 2^{11} тактов RTCCLK, то есть 512 импульсов за 32 секунды.

CALM уменьшает частоту RTC до 487.1, бит CALP увеличивает её до 488.5 ppm.

Совместная работа CALM и CALP даёт диапазон от -511 до +512 тактов RTCCLK за 32 секунды. То есть от -487.1 ppm до +488.5 ppm с разрешением около 0.954 ppm.

Калиброванную частоту (F_{CAL}) от заданной (F_{RTCCLK}) получают по формуле:

$$F_{CAL} = F_{RTCCLK} \times [1+(CALP \times 512 - CALM)/(2^{20}+CALM-CALP \times 512)]$$

Калибровка с PREDIV_A<3

Если значение PREDIV_A в регистре RTC_PRER меньше 3, то бит CALP игнорируется и считается равным 0.

При значении PREDIV_A меньше 3, предделитель PREDIV_S надо уменьшить так, чтобы ускорить секунду на 8 тактов RTCCLK. Так что, битами CALM можно просто добавить что-то между 255 и 256 импульсами (калибровка от 243.3 до 244.1 ppm).

При частоте RTCCLK в 32768 Hz и PREDIV_A=1 (делитель 2), PREDIV_S должен быть 16379, а не 16383 (меньше на 4), при PREDIV_A=0, PREDIV_S должен быть 32759, а не 32767 (меньше на 8).

При уменьшении PREDIV_S калиброванная частота вычисляется как:

$$F_{CAL} = F_{RTCCLK} \times [1+(256-CALM)/(2^{20}+CALM-256)]$$

В этом случае, CALM[7:0]=0x100 (середина диапазона CALM) верно при RTCCLK=32768.00 Hz.

Проверка калибровки RTC

Для этого наружу можно вывести сигнал 1 Hz и там его измерять.

Точность измерения в заданный интервал времени зависит от выравнивания длины цикла калибровки с периодом измерения. Ошибка может составить до 2 тактов RTCCLK. Лучше чтобы они были равны, тогда играет роль только разрешающая способность цифровой калибровки.

- По умолчанию, длительность калибровки равна 32 секундам.
  В этом случае гарантируется точность 0.477 ppm (0.5 такта RTCCLK за 32 секунды).
- Установка бита CALW16 в регистре RTC_CALR задаёт 16-секундный цикл калибровки.
  В этом случае максимальная ошибка и точность RTC равны 0.954 ppm.
- Установка бита CALW8 в регистре RTC_CALR задаёт 8-секундный цикл калибровки.
  В этом случае максимальная ошибка и точность RTC равны 1.907 ppm.

Ре-калибровка налету

В регистр RTC_CALR можно писать ещё и при RTC_ISR/INITF=0:

1. Смотрим флаг RTC_ISR/RECALPF (идёт рекалибровка).
2. При 0, пишем новое значение в RTC_CALR. RECALPF встанет сам
3. Новые установки заработают через три цикла ck_apre после записи в to RTC_CALR.

26.3.12.Метки времени

Включаются они установкой бита TSE в регистре RTC_CR.

При появлении события на ножке, куда выведена альтернативная функция TIMESTAMP, ставится флаг TSF в регистре RTC_ISR и в регистры метки времени (RTC_TSSSR, RTC_TSTR, RTC_TSDR) пишутся регистры календаря. Прерывание разрешают битом TSIE регистра RTC_CR.

Если новое событие возникает при стоящем флаге TSF, то ставится флаг переполнения (TSOVF), а регистры метки (RTC_TSTR и RTC_TSDR) сохраняют данные предыдущего события.

NB: TSF встаёт через 2 такта ck_apre после появления события, а TSOVF без задержки. То есть, у близких событий флаг TSOVF может встать при ещё нулевом TSF. Так что, сначала смотрите TSF, а затем TSOVF.

Внимание: Если событие появляется после записи в TSF, но перед его снятием, то встают и TSF и TSOVF. Так что писать ‘0’ в TSF нужно только когда он читается как ‘1’.

Кроме того, запись метки времени может вызвать событие вмешательства, если направить их на одну ножку.

Альтернативная функция TIMESTAMP (RTC_TS)

Её можно направить на RTC_AF1 или RTC_AF2 битом TSINSEL регистра RTC_TAFCR. Если RTC_AF1 используется как TAMPER с фильтром (TAMPFLT не нулевой), то метки времени на RTC_AF2 направлять нельзя.

26.3.13.Детектор вмешательства

Есть два входа обнаружения вмешательства с работой по фронту или по уровню с фильтрацией.

Резервные регистры RTC

RTC_BKPxR это двадцать 32-регистров в резервном домене с питанием от V_{BAT} если нет V_{DD}. Они сбрасываются только сбросом резервного домена, но не системным сбросом, при выходе из режима Standby и событию вмешательства.

Инициализация детектора вмешательства

Два детектора связаны с флагами TAMP1F/TAMP2F в регистре RTC_ISR2. Входы включаются установкой битов TAMP1E/TAMP2E в регистре RTC_TAFCR.

Событие вмешательства сбрасывает регистры RTC_BKPxR.

Бит TAMPIE регистра RTC_TAFCR разрешает прерывание вмешательства.

Метка времени события вмешательства

Бит TAMPTS разрешает запись метки времени. В этом случае, одновременно с битами флагов вмешательства (TAMP1F, TAMP2F) ставится флаг TSF и/или TSOVF.

Детектор фронта на входе

Если биты TAMPFLT равны “00”, то ножки TAMPER выдают событие (RTC_TAMP[2:1]) по переднему или заднему фронту, в зависимости от соответствующего бита TAMPxTRG. При этом резисторы подпорки отключены.

Внимание: Входной сигнал объединён по AND с TAMPxE дабы засечь вмешательство в случае его появления до разрешения ножки TAMPERx.

- При TAMPxTRG = 0: если функция TAMPERx была высокой до её включения (бит TAMPxE=1), то событие обнаруживается при разрешении TAMPERx, даже если переднего фронта не было.
- При TAMPxTRG = 1: если функция TAMPERx была низкой до её включения, то событие обнаруживается при разрешении TAMPERx, даже если заднего фронта не было.

После поимки и очистки события вмешательства, функцию TAMPERx надо выключить и включить снова (TAMPxE=1) до перезаписи регистров RTC_BKPxR.

NB: Детектор вмешательства работает при выключенном V_{DD}. Во избежание нежелательного стирания резервных регистров, ножку для входа функции TAMPER надо снаружи подключать к нужному уровню.

Детектор уровня с фильтрацией

Разрешается установкой не-нулевого значения TAMPFLT. Событие выдаётся при обнаружении на входе 2, 4 или 8 (зависит от TAMPFLT) последовательных выборок уровня, заданного битами TAMPxTRG (TAMP1TRG/TAMP2TRG).

Предзаряд входов TAMPER до их опроса через внутренние резисторы подпорки можно отключить установкой бита TAMPPUDIS. Длительность предзаряда определена битами TAMPPRCH.

Частота опроса входов задаётся через TAMPFREQ.

Альтернативные функции TAMPER

Функцию TAMPER1 (RTC_TAMP1) можно направить на RTC_AF1(PC13) или RTC_AF2 (PI8) битом TAMP1INSEL регистра RTC_TAFCR. Бит TAMPE при записи TAMP1INSEL должен быть чист.

Функция TAMPER 2 соответствует ножке RTC_TAMP2.

26.3.14.Вывод тактов калибровки

Бит COE регистра RTC_CR разрешает выдачу опорных тактов на выход RTC_CALIB. Если бит COSEL регистра RTC_CR снят и PREDIV_A=0x7F, то частота RTC_CALIB равна $f_{RTCCLK/64}$. Это равно 512 Hz при частоте RTCCLK = 32.768 kHz.

Значение калибровки в регистре RTC_CALIBR на выход RTC_CALIB не влияет. Скважность сигнала на RTC_CALIB не постоянная, есть небольшой дребезг заднего фронта, использовать лучше передний фронт.

Если бит COSEL стоит и (PREDIV_S+1) ненулевое, кратное 256 (т.е. PREDIV_S[7:0]=0xFF),то частота RTC_CALIB равна f_{RTCCLK}/(256 * (PREDIV_A+1)). Это равно 1 Hz для значений предделителя по умолчанию s (PREDIV_A = 0x7F, PREDIV_S = 0xFF), при RTCCLK = 32.768 kHz. При включённой операции сдвига (SHPF=1) выход 1 Hz может переключаться во время сдвига.

Вывод альтернативной функции калибровки

Стоящий бит COE регистра RTC_CR разрешает вывод функции RTC_CALIB на RTC_AF1.

NB: При выборе RTC_CALIB или RTC_ALARM, RTC_AF1 автоматически ставится на вывод альтернативной функции.

26.3.15.Вывод будильников

Вывести можно ALRAF, ALRBF или WUTF. Это соответствующие биты регистра RTC_ISR.

Выводимую на RTC_ALARM альтернативную функцию задают биты OSEL[1:0] регистра RTC_CR.

Полярность вывода задаёт бит POL регистра RTC_CR. POL=1 инвертирует состояние флага.

Вывод альтернативной функции будильника

Открытый сток или двухтактный каскад выхода RTC_ALARM выбирается в регистре RTC_TAFCR битом ALARMOUTTYPE.

NB: RTC_ALARM приоритетнее RTC_CALIB (бит COE не влияет).

При выборе RTC_CALIB или RTC_ALARM, RTC_AF1 автоматически ставится на вывод альтернативной функции.

26.4. RTC и экономные режимы

Таблица 119. Экономные режимы и RTC

Режим	Описание
Sleep	Не влияет. Прерывание RTC пробуждает.
Stop	При тактировании от LSE или LSI, RTC остаётся активным. RTC alarm, RTC tamper, RTC time stamp и RTC Wakeup пробуждают.
Standby	При тактировании от LSE или LSI, RTC остаётся активным. RTC alarm, RTC tamper, RTC time stamp и RTC Wakeup пробуждают.

26.5. Прерывания RTC

Все прерывания RTC подключены к контроллеру EXTI.

Включаем прерывания.

RTC Alarm (будильник):

1. Ставим и включаем EXTI Line 17 в режим прерываний по переднему фронту.
2. Ставим и включаем канал RTC_Alarm IRQ в NVIC.
3. Ставим RTC на выдачу будильников (Alarm A или Alarm B).

RTC Wakeup (побудка):

1. Ставим и включаем EXTI Line 22 в режим прерываний по переднему фронту.
2. Ставим и включаем канал RTC_WKUP IRQ в NVIC.
3. Ставим RTC на выдачу побудки.

RTC Tamper (вмешательство):

1. Ставим и включаем EXTI Line 21 в режим прерываний по переднему фронту.
2. Ставим и включаем канал TAMP_STAMP IRQ в NVIC.
3. Ставим RTC на выдачу RTC tamper.

RTC TimeStamp (метки времени):

1. Ставим и включаем EXTI Line 21 в режим прерываний по переднему фронту.
2. Ставим и включаем канал TAMP_STAMP IRQ в NVIC.
3. Ставим RTC на выдачу меток времени.

Таблица 120. Управляющие биты прерываний

Прерывание	Флаг	Разрешение	Выход из Sleep	Выход из Stop	Выход из Standby
Alarm A	ALRAF	ALRAIE	yes	yes(1)	yes(1)
Alarm B	ALRBF	ALRBIE	yes	yes(1)	yes(1)
Wakeup	WUTF	WUTIE	yes	yes(1)	yes(1)
TimeStamp	TSF	TSIE	yes	yes(1)	yes(1)
Tamper1 detection	TAMP1F	TAMPIE	yes	yes(1)	yes(1)
Tamper2 detection(2)	TAMP2F	TAMPIE	yes	yes(1)	yes(1)

1. Выход из STOP и Standby возможен только при тактировании RTC от LSE или LSI.
2. Если ножка RTC_TAMPER2 есть на корпусе.

26.6. Регистры RTC

Регистры доступны словами.

26.6.1. Регистр времени (RTC_TR)

Теневой регистр времени календаря, пишут только в режиме инициализации. Всё в формате BCD.

Смещение адреса: 0x00

По сбросу: 0x0000 0000, при BYPSHAD = 1 не сбрасывается.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									PM	HT[1:0]		HU[3:0]			
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

— Биты 31:23 Резерв, не трогать.

— Бит 22 **PM**: Формат времени
  0: AM или 24 часа
  1: PM
— Биты 21:20 **HT[1:0]**: Десятки часов
— Биты 19:16 **HU[3:0]**: Единицы часов
— Бит 15 Резерв, не трогать.
— Биты 14:12 **MNT[1:0]**: Десятки минут
— Биты 11:8 **MNU[3:0]**: Единицы минут
— Бит 7 Резерв, не трогать.
— Биты 6:4 **ST[1:0]**: Десятки секунд
— Биты 3:0 **SU[3:0]**: Единицы секунд

Регистр защищён от записи. Процедура записи приведена ранее.

26.6.2. Регистр даты (RTC_DR)

Теневой регистр даты календаря, пишут только в режиме инициализации. Всё в формате BCD.

Смещение адреса: 0x04

По сбросу: 0x0000 2101, при BYPSHAD = 1 не сбрасывается.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								YT[3:0]				YU[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDU[2:0]			MT	MU[3:0]				Reserved		DT[1:0]		DU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

— Биты 31:24 Резерв, не трогать.
— Биты 23:20 **YT[3:0]**: Десятки года
— Биты 19:16 **YU[3:0]**: Единицы года
— Биты 15:13 **WDU[2:0]**: День недели (1-7, 0-нельзя)
— Бит 12 **MT**: Десятки месяца
— Биты 11:8 **MU[3:0]**: Единицы месяца
— Биты 7:6 Резерв, не трогать.
— Биты 6:4 **DT[1:0]**: Десятки даты
— Биты 3:0 **DU[3:0]**: Единицы даты

Регистр защищён от записи. Процедура записи приведена ранее.

26.6.3. Регистр управления (RTC_CR)

Смещение адреса: 0x08

По сбросу резервного домена: 0x0000 0000, системный сброс не трогает.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								COE	OSEL[1:0]		POL	COSEL	BKP	SUB1H	ADD1H
								rw	rw	rw	rw	rw	rw	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	DCE	FMT	BYPS HAD	REFCKON	TSEDGE	WUCKSEL[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:24 Резерв, не трогать.
— Бит 23 **COE**: Разрешение выхода RTC_CALIB
  0: Нельзя
  1: Можно
— Биты 22:21 **OSEL[1:0]**: Выбор флага для вывода на RTC_ALARM
  00: Ничего
  01: Alarm A
  10:Alarm B
  11: Wakeup (побудка)

— Бит 20 **POL**: Полярность выхода RTC_ALARM
0: Активный высокий
1: Активный низкий.
— Бит 19 **COSEL**: Выбор калибрами для вывода на RTC_CALIB при COE=1.
Действительно при RTCCLK = 32.768 kHz и PREDIV_A=127 и PREDIV_S=255.
0: 512 Hz
1: 1 Hz
— Бит 18 **BKP**: Сохранено ли летнее время или нет.
— Бит 17 **SUB1H**: Вычесть 1 час (на зимнее время), если время не нулевое
— Бит 16 **ADD1H**: Добавить 1 час (на летнее время)
— Бит 15 **TSIE**: Разрешение прерывания меток времени
0: Нельзя
1: Можно
— Бит 14 **WUIE**: Разрешение прерывания побудки
0: Нельзя
1: Можно
— Бит 13 **ALRBIE**: Разрешение прерывания будильника B
0: Нельзя
1: Можно
— Бит 12 **ALRAIE**: Разрешение прерывания будильника A
0: Нельзя
1: Можно
— Бит 11 **TSE**: Включение меток времени
0: Выкл.
1: Вкл.
— Бит 10 **WUTE**: Включение таймера побудки
0: Выкл.
1: Вкл.
NB: Если таймер выключен, то перед повторным включением дождитесь WUTWF=1.
— Бит 9 **ALRBE**: Включение будильника B
0: Выкл.
1: Вкл.
— Бит 8 **ALRAE**: Включение будильника A
0: Выкл.
1: Вкл.
— Бит 7 **DCE**: Включение грубой калибровки при PREDIV_A > 6
0: Выкл.
1: Вкл.
— Бит 6 **FMT**: Формат часов
0: 24 часа
1: AM/PM
— Бит 5 **BYPSHAD**: Обход теневых регистров
0: При чтении регистров RTC_SSR, RTC_TR и RTC_DR данные идут из них.
1: При чтении регистров RTC_SSR, RTC_TR и RTC_DR данные идут из счётчиков календаря.
NB: Если частота APB1 меньше семикратной частоты RTCCLK, то надо ставить BYPSHAD.
— Бит 4 **REFCKON**: Включение детектора опорной частоты (50 или 60 Hz)
0: Выкл.
1: Вкл.
NB: PREDIV_S должен быть 0x00FF.
— Бит 3 **TSEDGE**: Активный фронт события метки времени
0: Передний фронт TIMESTAMP
1: Задний фронт TIMESTAMP
NB: При изменении TSEDGE бит TSE должен быть снят, чтобы избежать лишней установки TSF

— Биты 2:0 **WUCKSEL[2:0]**: Выбор тактов побудки
000: RTC/16

001: RTC/8
010: RTC/4
011: RTC/2
10x: ck_spre (обычно 1 Hz)
11x: ck_spre (обычно 1 Hz) и к счётчику WUT добавляется 2^{16} (см. ниже)

NB: WUT = Единицы счётчика побудки = (0x0000 до 0xFFFF) + 0x10000 при WUCKSEL[2:1] = 11.

Биты 7, 6 и 4 пишут только при инициализации (RTC_ISR/INITF = 1).

Биты 2:0 пишут только при RTC_CR/WUTE = 0 и RTC_ISR/WUTWF = 1.

Изменение часа календаря во время его инкременты может маскировать сам инкремент.

Изменения ADD1H и SUB1H начинают работать на следующей секунде.

При изменении TSEDGE бит TSE должен быть снят, чтобы избежать лишней установки TSF.

Регистр защищён от записи. Процедура записи приведена ранее.

26.6.4. Регистр инициализации и состояния (RTC_ISR)

Смещение адреса: 0x0C

По сбросу резервного домена: 0x0000 0007, системный сброс снимает только `INIT`, `INITF` и `RSF`.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															RECALPF
															r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TAMP2F	TAMP1F	TSOVF	TSF	WUTF	ALRBF	ALRAF	INIT	INITF	RSF	INITS	SHPF	WUTWF	ALRBWF	ALRAWF
	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rw	r	rc_w0	r	r	r	r	r

— Биты 31:17 Резерв, не трогать.

— Бит 16 **RECALPF**: Флаг работающей рекалибровки
Сам ставится после записи в регистр RTC_CALR, блокируя его. Сам снимается после вступления новых установок в силу.

— Бит 15 Резерв, не трогать.

— Бит 14 **TAMP2F**: Флаг события на входе TAMPER2
Ставится аппаратно, снимается записью 0.

— Бит 13 **TAMP1F**: Флаг события на входе TAMPER1
Ставится аппаратно, снимается записью 0.

— Бит 12 **TSOVF**: Флаг переполнения меток времени
Ставится аппаратно при появлении метки времени и стоящем TSF, снимается записью 0 после снятия TSF.

— Бит 11 **TSF**: Флаг метки времени
Ставится аппаратно, снимается записью 0.

— Бит 10 **WUTF**: Флаг таймера побудки
Ставится при достижении самозагружаемым счётчиком 0, снимается записью 0 не менее чем на 1.5 периода RTCCLK перед новой установкой.

— Бит 9 **ALRBF**: Флаг Alarm B
Ставится аппаратно при совпадении RTC_TR и RTC_DR с RTC_ALRMBR, снимается записью 0.

— Бит 8 **ALRAF**: Флаг Alarm A
Ставится аппаратно при совпадении RTC_TR и RTC_DR с RTC_ALRMAR, снимается записью 0.

— Бит 7 **INIT**: Режим инициализации
0: Работа просто так
1: Можно писать регистры RTC_TR, RTC_DR и RTC_PRER. Счётчики стоят, считать начнут после снятия бита записью 0.

— Бит 6 **INITF**: Флаг режима инициализации
0: Нормальная работа
1: Можно писать регистры RTC_TR, RTC_DR и RTC_PRER.

— Бит 5 **RSF**: Флаг синхронизации регистров
Ставится аппаратно во время копирования регистров календаря в теневые (RTC_SSRx, RTC_TRx и RTC_DRx). Аппаратно снимается в режиме инициализации, операции сдвига (SHPF=1) или при обходе теневых регистров (BYPSHAD=1). Можно снимать программно.

0: Ещё не синхронизированы
1: Уже

— Бит 4 **INITS**: Флаг инициализации
Аппаратно ставится при ненулевом поле года.

— Бит 3 **SHPF**: Флаг работы операции сдвига
Аппаратно ставится при запуске операции сдвига записью в регистр RTC_SHIFTR. Снимается сам.

— Бит 2 **WUTWF**: Флаг разрешения записи таймера побудки
Аппаратно ставится через 2 такта RTCCLK после снятия бита WUTE в RTC_CR, аппаратно снимается через 2 такта RTCCLK после установки бита WUTE в RTC_CR. Таймер побудки можно писать при снятом бите WUTE и стоящем бите WUTWF.
0: Писать нельзя
1: Можно

— Бит 1 **ALRBWF**: Флаг разрешения записи Alarm B
Ставится аппаратно после снятия бита ALRBIE в RTC_CR. Снимается при инициализации.
0: Писать нельзя
1: Можно

— Бит 0 **ALRAWF**: Флаг разрешения записи Alarm A
Ставится аппаратно после снятия бита ALRAIE в RTC_CR. Снимается при инициализации.
0: Писать нельзя
1: Можно

NB: Биты ALRAF, ALRBF, WUTF и TSF снимаются через 2 такта APB после записи 0.
Регистр защищён от записи (кроме битов RTC_ISR[13:8]). Процедура записи есть выше.

26.6.5. Регистр предделителя (RTC_PRER)

Смещение адреса: 0x10

По сбросу резервного домена: 0x007F 00FF, системный сброс не трогает.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									PREDIV_A[6:0]						
									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PREDIV_S[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:23 Резерв, не трогать.

— Биты 22:16 **PREDIV_A[6:0]**: Асинхронный предделитель
Частота ck_apre = частота RTCCLK/(PREDIV_A+1)

— Бит 15 Резерв, не трогать.

— Биты 14:0 **PREDIV_S[14:0]**: Синхронный предделитель
Частота ck_spre frequency = частота ck_apre/(PREDIV_S+1)
NB: Писать можно только при инициализации за два отдельных обращения .
Регистр защищён от записи. Процедура записи есть выше.

26.6.6. Регистр таймера побудки (RTC_WUTR)

Смещение адреса: 0x14

По сбросу резервного домена: 0x0000 FFFF, системный сброс не трогает.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WUT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:16 Резерв, не трогать.

— Биты 15:0 **WUT[15:0]**: Перегружаемое значение
У включённого таймера (WUTE=1), флаг WUTF встаёт каждые (WUT[15:0] + 1) ck_wut тактов. Период ck_wut задаётся битами WUCKSEL[2:0] регистра RTC_CR. При WUCKSEL[2] = 1, таймер становится 17-битным, а WUCKSEL[1] - битом WUT[16] счётчика.

NB: Первый раз флаг WUTF встаёт через (WUT+1) ck_wut тактов после установки WUTE. Писать WUT[15:0] = 0x0000 и WUCKSEL[2:0] =011 (RTCCLK/2) запрещено.

NB: Писать можно только при WUTWF = 1 в RTC_ISR.

Регистр защищён от записи. Процедура записи есть выше.

26.6.7. Регистр калибровки (RTC_CALIBR)

Смещение адреса: 0x18

По сбросу резервного домена: 0x0000 0000, системный сброс не трогает.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								DCS	Reserved		DC[4:0]				
								rw			rw	rw	rw	rw	rw

— Биты 31:8 Резерв, не трогать.

— Бит 7 **DCS**: Знак калибровки

0: Позитивная: частота обновления календаря увеличивается

1: Негативная: частота обновления календаря уменьшается

— Биты 6:5 Резерв, не трогать.

— Биты 4:0 **DC[4:0]**: Цифровая калибровка

DCS = 0 (позитивная)

00000: + 0 ppm

00001: + 4 ppm (округлённо)

00010: + 8 ppm (округлённо)

..

11111: + 126 ppm (округлённо)

DCS = 1 (негативная)

00000: –0 ppm

00001: –2 ppm (округлённо)

00010: –4 ppm (округлённо)

..

11111: –63 ppm (округлённо)

NB: Писать можно только при RTC_ISR/INITF = ‘1’.

Регистр защищён от записи. Процедура записи есть выше.

26.6.8. Регистр будильника Alarm A (RTC_ALRMAR)

Смещение адреса: 0x1C

По сбросу резервного домена: 0x0000 0000, системный сброс не трогает.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	WDSEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Бит 31 **MSK4**: Маска даты Alarm A

0: При совпадении даты/дня ставить Alarm A

1: Не сравнивать

— Бит 30 **WDSEL**: Выбор дня недели

0: DU[3:0] это единицы даты

1: DU[3:0] это день недели. DT[1:0] безразличны.

— Биты 29:28 **DT[1:0]**: Десятки даты в формате BCD.

— Биты 27:24 **DU[3:0]**: Единицы даты или день в формате BCD
— Бит 23 **MSK3**: Маска часов Alarm A
0: При совпадении часов ставить Alarm A
1: Не сравнивать
— Бит 22 **PM**: Формат AM/PM
0: AM или 24 часа
1: PM
— Биты 21:20 **HT[1:0]**: Десятки часов в формате BCD.
— Биты 19:16 **HU[3:0]**: Единицы часов в формате BCD
— Бит 15 **MSK2**: Маска минут Alarm A
0: При совпадении минут ставить Alarm A
1: Не сравнивать
— Биты 14:12 **MNT[2:0]**: Десятки минут в формате BCD.
— Биты 11:8 **MNU[3:0]**: Единицы минут в формате BCD
— Бит 7 **MSK1**: Маска секунд Alarm A
0: При совпадении секунд ставить Alarm A
1: Не сравнивать
— Биты 6:4 **ST[2:0]**: Десятки секунд в формате BCD
— Биты 3:0 **SU[3:0]**: Единицы секунд в формате BCD

NB: Писать можно только в режиме инициализации и при RTC_ISR/ALRAWF = ‘1’.
Регистр защищён от записи. Процедура записи есть выше.

26.6.9. Регистр будильника Alarm B (RTC_ALRMBR)

Смещение адреса: 0x20

По сбросу резервного домена: 0x0000 0000, системный сброс не трогает.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	WDSEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Бит 31 **MSK4**: Маска даты Alarm B
0: При совпадении даты/дня ставить Alarm B
1: Не сравнивать
— Бит 30 **WDSEL**: Выбор дня недели
0: DU[3:0] это единицы даты
1: DU[3:0] это день недели. DT[1:0] безразличны.
— Биты 29:28 **DT[1:0]**: Десятки даты в формате BCD.
— Биты 27:24 **DU[3:0]**: Единицы даты или день в формате BCD
— Бит 23 **MSK3**: Маска часов Alarm B
0: При совпадении часов ставить Alarm B
1: Не сравнивать
— Бит 22 **PM**: Формат AM/PM
0: AM или 24 часа
1: PM
— Биты 21:20 **HT[1:0]**: Десятки часов в формате BCD.
— Биты 19:16 **HU[3:0]**: Единицы часов в формате BCD
— Бит 15 **MSK2**: Маска минут Alarm B
0: При совпадении минут ставить Alarm B
1: Не сравнивать
— Биты 14:12 **MNT[2:0]**: Десятки минут в формате BCD.
— Биты 11:8 **MNU[3:0]**: Единицы минут в формате BCD
— Бит 7 **MSK1**: Маска секунд Alarm B
0: При совпадении секунд ставить Alarm B
1: Не сравнивать

— Биты 6:4 **ST[2:0]**: Десятки секунд в формате BCD
— Биты 3:0 **SU[3:0]**: Единицы секунд в формате BCD

NB: Писать можно только в режиме инициализации и при RTC_ISR/ALRBWF = ‘1’.
Регистр защищён от записи. Процедура записи есть выше.

26.6.10.Регистр защиты записи (RTC_WPR)

Смещение адреса: 0x24

По сбросу резервного домена: 0x0000 0000, системный сброс не трогает.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								KEY							
								w	w	w	w	w	w	w	w

— Биты 31:8 Резерв, не трогать.
— Биты 7:0 **KEY**: Ключ защиты записи
Пишется программно, читается как 0. Описание есть выше.

26.6.11.Регистр субсекунд (RTC_SSR)

Смещение адреса: 0x28

По сбросу резервного домена и системным сбросом при BYPSHAD = 0 : 0x0000 0000, иначе системный сброс не трогает.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

— Биты 31:16 Резерв, не трогать.
— Биты 15:0 **SS[15:0]**: Субсекунды в синхронном предделителе
Доли секунд вычисляют как:
Second fraction = (PREDIV_S - SS) / (PREDIV_S + 1)
NB: SS может быть больше PREDIV_S только после операции сдвига. В этом случае регистры RTC_TR/RTC_DR показывают время/дату на секунду больше реального.

26.6.12.Регистр управления сдвигом (RTC_SHIFTR)

Смещение адреса: 0x2C

По сбросу резервного домена: 0x0000 0000, системный сброс не трогает.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD1S	Reserved														
w	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SUBFS[14:0]														
r	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

— Бит 31 **ADD1S**: Добавить 1 секунду
0: Ничего
1: Всё-таки добавить
Только запись, читается как 0. Запись не работает во время операции сдвига (SHPF=1, в RTC_ISR).
Бит нужен вместе с SUBFS для добавления долей секунды при атомарной операции.
— Биты 30:15 Резерв, не трогать.
— Биты 14:0 **SUBFS[14:0]**: Вычесть долю секунды

Только запись, читается как 0. Запись не работает во время операции сдвига (SHPF=1, в RTC_ISR).

Число из SUBFS добавляется к обратному счётчику синхронного предделителя, тем самым задерживая время:

Задержка(секунды) = SUBFS / (PREDIV_S + 1)

Бит ADD1S вместе с SUBFS в итоге отнимает доли секунды из счётчика, смещая время вперёд:

Вперёд(секунды) = (1 - (SUBFS / (PREDIV_S + 1))) .

NB: Запись в SUBFS чистит RSF. Для появления сдвинутого времени в теневых регистрах надо подождать RSF=1.

NB: Регистр защищён от записи. Процедура записи есть выше.

26.6.13.Регистр времени меток времени (RTC_TSTR)

Смещение адреса: 0x30

По сбросу резервного домена: 0x0000 0000, системный сброс не трогает.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved									PM	HT[1:0]		HU[3:0]			
									r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	r	r	r	r	r	r	r		r	r	r	r	r	r	r

— Биты 31:23 Резерв, не трогать.

— Бит 22 **PM**: Формат AM/PM

0: AM или 24 часа

1: PM

— Биты 21:20 **HT[1:0]**: Десятки часов в формате BCD.

— Биты 19:16 **HU[3:0]**: Единицы часов в формате BCD

— Бит 15 Резерв, не трогать.

— Биты 14:12 **MNT[2:0]**: Десятки минут в формате BCD.

— Биты 11:8 **MNU[3:0]**: Единицы минут в формате BCD

— Бит 7 Резерв, не трогать.

— Биты 6:4 **ST[2:0]**: Десятки секунд в формате BCD

— Биты 3:0 **SU[3:0]**: Единицы секунд в формате BCD

NB: Содержимое этого регистра хранится только при стоящем бите TSF в RTC_ISR. При снятии TSF регистр стирается.

26.6.14.Регистр даты меток времени (RTC_TSTR)

Смещение адреса: 0x34

По сбросу резервного домена: 0x0000 0000, системный сброс не трогает.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDU[1:0]			MT	MU[3:0]				Reserved		DT[1:0]		DU[3:0]			
r	r	r	r	r	r	r	r			r	r	r	r	r	r

— Биты 31:16 Резерв, не трогать.

— Биты 15:13 **WDU[1:0]**: Единицы дня недели.

— Бит 12 **MT**: Десятки месяца в формате BCD

— Биты 11:8 **MU[3:0]**: Единицы месяца в формате BCD

— Биты 7:6 Резерв, не трогать.

— Биты 5:4 **DT[2:0]**: Десятки даты в формате BCD

— Биты 3:0 **DU[3:0]**: Единицы даты в формате BCD

NB: Содержимое этого регистра хранится только при стоящем бите TSF в RTC_ISR. При снятии TSF регистр стирается.

26.6.15.Регистр субсекунд меток времени (RTC_TSSSR)

Смещение адреса: 0x38

По сбросу резервного домена: 0x0000 0000, системный сброс не трогает.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

— Биты 31:16 Резерв, не трогать.

— Биты 15:0 **SS[15:0]**: Субсекунды момента метки времени из счётчика синхронного предделителя.

NB: Содержимое этого регистра хранится только при стоящем бите TSF в RTC_ISR. При снятии TSF регистр стирается.

26.6.16.Регистр калибровки (RTC_CALR)

Смещение адреса: 0x3C

По сбросу резервного домена: 0x0000 0000, системный сброс не трогает.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CALP	CALW8	CALW16	Reserved				CALM[8:0]								
rw	rw	rw	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:16 Резерв, не трогать.

— Бит 15 **CALP**: Увеличение частоты RTC на 488.5 ppm

0: Импульсы RTCCLK не добавлять.

1: Добавлять 1 импульс RTCCLK на каждые 2^{11} импульсов (частота растёт на 488.5 ppm).

Используется совместно с CALM, понижающей частоту. При входной частоте RTCCLK = 32768 Hz, число добавленных импульсов в 32-секундном окне равно: (512 * CALP) - CALM.

— Бит 14 **CALW8**: 8-секундный период калибровки

CALM[1:0] залипает на “00” при CALW8=’1’.

— Бит 13 **CALW16**: 16-секундный период калибровки

CALM[0] залипает на “0” при CALW16=’1’. Нельзя ставить одновременно с CALW8.

— Биты 12:9 Резерв, не трогать.

— Биты 8:0 **CALM[8:0]**: Минусовая калибровка

Частота уменьшается маскированием CALM импульсов из 2^{20} RTCCLK (32 секунды при входной частоте равной 32768 Hz). Разрешение равно 0.9537 ppm. Используется совместно с CALP.

NB: Регистр защищён от записи. Процедура записи есть выше.

26.6.17.Регистр вмешательства и альтернативных функций (RTC_TAFCR)

Смещение адреса: 0x40

По сбросу резервного домена: 0x0000 0000, системный сброс не трогает.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													ALARMOUTTYPE	TSINSEL	TAMP1INSEL
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TAMP-PUDIS	TAMP-PRCH[1:0]		TAMPFLT[1:0]		TAMPFREQ[2:0]			TAMPTS	Reserved		TAMP2-TRG	TAMP2E	TAMPIE	TAMP1TRG	TAMP1E
rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw

— Биты 31:19 Резерв, не трогать.

— Бит 18 **ALARMOUTTYPE**: Тип выхода RTC_ALARM

0: Открытый сток

1: Двухтактный

— Бит 17 **TSINSEL**: Направление ножки TIMESTAMP
0: RTC_AF1
1: RTC_AF2
— Бит 16 **TAMP1INSEL**: Направление ножки TAMPER1
0: RTC_AF1
1: RTC_AF2
NB: При изменении TAMP1INSEL бит TAMP1E надо снимать, иначе может встать TAMP1F.
— Бит 15 **TAMPPUDIS**: Выключение подпорки ножек TAMPER для предзаряда
0: Предзаряд (подпорка) вкл.
1: Предзаряд (подпорка) выкл.
— Биты 14:13 **TAMPPRCH[1:0]**: Длительность предзаряда ножек TAMPER
0x0: 1 такт RTCCLK
0x1: 2 такта RTCCLK
0x2: 4 такта RTCCLK
0x3: 8 тактов RTCCLK
— Биты 12:11 **TAMPFLT[1:0]**: Счётчик фильтра ножек TAMPER
Это число последовательных выборок уровня TAMP*TRG для пуска события TAMPER.
0x0: Пуск по фронту выхода на активный уровень (подпорки нет).
0x1: После 2 последовательных выборок активного уровня.
0x2: После 4 последовательных выборок активного уровня.
0x3: После 8 последовательных выборок активного уровня.
— Биты 10:8 **TAMPFREQ[2:0]**: Частота выборки ножек TAMPER
0x0: RTCCLK / 32768 (1 Hz при RTCCLK = 32768 Hz)
0x1: RTCCLK / 16384 (2 Hz при RTCCLK = 32768 Hz)
0x2: RTCCLK / 8192 (4 Hz при RTCCLK = 32768 Hz)
0x3: RTCCLK / 4096 (8 Hz при RTCCLK = 32768 Hz)
0x4: RTCCLK / 2048 (16 Hz при RTCCLK = 32768 Hz)
0x5: RTCCLK / 1024 (32 Hz при RTCCLK = 32768 Hz)
0x6: RTCCLK / 512 (64 Hz при RTCCLK = 32768 Hz)
0x7: RTCCLK / 256 (128 Hz при RTCCLK = 32768 Hz)
— Бит 7 **TAMPTS**: Запись метки времени по событию TAMPER
0: Не надо
1: Писать даже при RTC_CR/TSE=0.
— Биты 6:5 Резерв, не трогать.
— Бит 4 **TAMP2TRG**: Активный уровень TAMPER2
При TAMPFLT != 00
0: Низкий.
1:Высокий.
При TAMPFLT = 00:
0: Передний фронт.
1: Задний фронт.
— Бит 3 **TAMP2E**: Включение TAMPER2
— Бит 2 **TAMPIE**: Разрешение прерываний TAMPER
— Бит 1 **TAMP1TRG**: Активный уровень TAMPER1
При TAMPFLT != 00
0: Низкий.
1:Высокий.
При TAMPFLT = 00:
0: Передний фронт.
1: Задний фронт.
NB: При TAMPFLT=0, при изменении TAMPxTRG бит TAMPxE надо снимать, дабы TAMPxF не встал.
— Бит 0 **TAMP1E**: Включение TAMPER1

26.6.18.Регистр субсекунд Alarm A (RTC_ALRMASSR)

Смещение адреса: 0x44

По сбросу резервного домена: 0x0000 0000, системный сброс не трогает.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				MASKSS[3:0]				Reserved							
r	r	r	r	rw	rw	rw	rw	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	SS[14:0]														
r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

— Биты 31:28 Резерв, не трогать.

— Биты 27:24 **MASKSS[3:0]**: Маскирование старших битов , начиная с указанного

0: Субсекунды Alarm A не сравнивать. Реакция по остальным полям.

1: Сравнение только SS[0].

2: Сравнение только SS[14:2].

3: Сравнение только SS[14:3].

…

12: Сравнение только SS[14:12].

13: Сравнение только SS[14:13].

14: Сравнение только SS[14].

15: Сравнение всех 15 битов SS.

Бит переполнения синхронного счётчика (бит 15) никогда не сравнивается. Он может отличаться от 0 только при сдвиге.

— Биты 23:15 Резерв, не трогать.

— Биты 14:0 **SS[14:0]**: Субсекунды для сравнения.

NB: Писать можно только при снятом бите ALRAE в RTC_CR или во время инициализации.

Регистр защищён от записи. Процедура записи есть выше.

26.6.19.Регистр субсекунд Alarm B (RTC_ALRMBSSR)

Смещение адреса: 0x48

По сбросу резервного домена: 0x0000 0000, системный сброс не трогает.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				MASKSS[3:0]				Reserved							
r	r	r	r	rw	rw	rw	rw	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	SS[14:0]														
r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

— Биты 31:28 Резерв, не трогать.

— Биты 27:24 **MASKSS[3:0]**: Маскирование старших битов , начиная с указанного

0: Субсекунды Alarm B не сравнивать. Реакция по остальным полям.

1: Сравнение только SS[0].

2: Сравнение только SS[14:2].

3: Сравнение только SS[14:3].

…

12: Сравнение только SS[14:12].

13: Сравнение только SS[14:13].

14: Сравнение только SS[14].

15: Сравнение всех 15 битов SS.

Бит переполнения синхронного счётчика (бит 15) никогда не сравнивается. Он может отличаться от 0 только при сдвиге.

— Биты 23:15 Резерв, не трогать.

— Биты 14:0 **SS[14:0]**: Субсекунды для сравнения.

NB: Писать можно только при снятом бите ALRBE в RTC_CR или во время инициализации.

Регистр защищён от записи. Процедура записи есть выше.

26.6.20.Резервные регистры (RTC_BKPxR)

Смещение адреса: от 0x48 до 0x9C

По сбросу резервного домена: 0x0000 0000, системный сброс не трогает.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BKP[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BKP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

— Биты 31:0 **BKP[31:0]**: Их можно читать и писать. При выключенном V_{DD} они питаются от V_{BAT}. Системный сброс на них не влияет, только факт вмешательства (всё время при TAMPxF=1).

26.7. Карта регистров RTC

Таблица 121. Карта регистров RTC

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**RTC_TR**	Reserved									PM	HT [1:0]		HU[3:0]				Reserved	MNT[2:0]			MNU[3:0]				Reserved	ST[2:0]			SU[3:0]			
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x04	**RTC_DR**	Reserved									YT[3:0]			YU[3:0]				WDU[2:0]			MT	MU[3:0]				Reserved	DT [1:0]		DU[3:0]			
	Reset value																	0	0	1	0	0	0	0	1		0	0	0	0	0	1
0x08	**RTC_CR**	Reserved								COE	OSEL [1:0]		POL	COSEL	BKP	SUB1H	ADD1H	TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	DCE	FMT	BYPSHAD	REFCKON	TSEDGE	WCKSEL [2:0]		
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	RTC_ISR	Reserved																	TAMP2F	TAMP1F	TSOVF	TSF	WUTF	ALRBF	ALRAF	INIT	INITF	RSF	INITS	SHPF	WUTWF	ALRBWF	ALRAWF
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
0x10	**RTC_PRER**	Reserved									PREDIV_A[6:0]							Reserved	PREDIV_S[14:0]														
	Reset value										1	1	1	1	1	1	1		0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0x14	**RTC_WUTR**	Reserved																WUT[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x18	**RTC_CALIBR**	Reserved																								DCS	Reserved	DC[4:0]					
	Reset value																									0		0	0	0	0	0	

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1C	**RTC_ALRMAR**	MSK4	WDSEL	DT [1:0]		DU[3:0]				MSK3	PM	HT [1:0]		HU[3:0]				MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**RTC_ALRMBR**	MSK4	WDSEL	DT [1:0]		DU[3:0]				MSK3	PM	HT [1:0]		HU[3:0]				MSK2	MNT[2:0]			MNU[3:0]				MSK2	ST[2:0]			SU[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	**RTC_WPR**	Reserved																								KEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x28	**RTC_SSR**	Reserved																SS[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	RTC_SHIFTR	ADD1S	Reserved																SUBFS[14:0]														
	Reset value	0																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	**RTC_TSTR**	Reserved								PM	HT[1:0]			HU[3:0]				Reserved	MNT[2:0]			MNU[3:0]				Reserved	ST[2:0]			SU[3:0]			
	Reset value									0	0	0		0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x38	**RTC_TSSSR**	Reserved																SS[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	**RTC_ CALR**	Reserved																CALP	CALW8	CALW16	Reserved					CALM[8:0]							
	Reset value																	0	0	0						0	0	0	0	0	0	0	0
0x40	RTC_TAFCR	Reserved												ALARMOUTTYPE	TSINSEL	TAMP1INSEL	TAMPPUDIS	TAMPPRCH[1:0]		TAMPFLT[1:0]		TAMPFREQ[2:0]			TAMPTS	Reserved	TAMP2TRG	TAMP2E	TAMPIE	TAMP1ETRG	TAMP1E	
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	
0x44	**RTC_ ALRMASSR**	Reserved				MASKSS[3:0]				Reserved									SS[14:0]														
	Reset value					0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	**RTC_ ALRMBSSR**	Reserved				MASKSS[3:0]				Reserved									SS[14:0]														
	Reset value					0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50 to 0x9C	**RTC_BKP0R**	BKP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	to **RTC_BKP19R**	BKP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

27. Интерфейс I²C

27.1. Введение в I²C

Шина "промеж-интегрального"интерфейса поддерживает несколько ведущих устройств и два режима скорости: стандартный (Sm, до 100 kHz) и быстрый (Fm, до 400 kHz).

Может использоваться в разных целях, включая вычисление и проверку CRC, SMBus (шину управления системой) и PMBus (шину управления питанием). На некоторых устройствах может пользоваться DMA.

27.2. Основные свойства I²C

- Преобразователь протокола параллельная шина/I²C
- Несколько ведущих: один интерфейс может работать Ведущим или Ведомым
- Ведущий I²C:
  - Выдача тактов
  - Выдача сигналов Start и Stop
- Ведомый I²C:
  - Определение программируемого адреса I²C
  - Подтверждение на 2 адреса ведомых
  - Обнаружение сигнала Stop
- Формирование и определение 7-бит/10-бит адресов и Общий Вызов
- Скорость связи:
  - Стандартная (до 100 kHz)
  - Быстрая (до 400 kHz)
- Аналоговый фильтр шума
- Программируемый цифровой фильтр шума для STM32F42xxx STM32F43xxx
- Флаги состояния:
  - Флаг Приём/Передача
  - Передача Конец-Байта
  - I²C занят
- Флаги ошибки:
  - Потеря арбитража ведущим
  - Сбой подтверждения передачи адрес/данные
  - Обнаружение неправильного появления старта и останова
  - Переполнение/Исчерпание при выключенной растяжке тактов
- 2 вектора прерываний:
  - Успешная передача адреса/данных
  - Ошибка
- Возможная растяжка тактов
- Однобайтовый буфер с DMA
- Конфигурируемая PEC (проверка ошибки пакетов):
  - Передача PEC последним байтом
  - Проверка PEC для последнего байта
- Совместимость с SMBus 2.0:
  - задержка таймаута тактов 25 ms
  - кумулятивное расширение тактов ведущего 10 ms
  - кумулятивное расширение тактов ведомого 25 ms
  - Аппаратная проверка/ вычисление PEC с контролем `ACK`
  - Протокол разрешения адреса (ARP)
- Совместимость с PMBus

NB: Не во всех устройствах есть всё приведённое выше.

27.3. Функциональное описание I²C

Кроме приёма и передачи есть последовательно/параллельный преобразователь. Прерывания управляются программно. К шине I²C подключается ножками данных (SDA) и тактов (SCL).

27.3.1. Выбор режима

Режимов четыре: Передатчик ведомого, Приёмник ведомого, Передатчик ведущего и Приёмник ведущего.

По умолчанию работает в режиме ведомого. Автоматически переключается в ведущего при генерации START, и в ведомого при потере арбитража или появления STOP.

Связь

Ведущий инициирует передачу данных и выдаёт такты. Последовательная передача начинается по START и завершается по STOP. Оба события задаются программно в режиме ведущего.

Ведомый узнаёт свой адрес (7 или 10-бит) и адрес Общего Вызова. Опознавание Общего Вызова разрешается программно.

Данные и адреса передаются байтами, старший бит первым. Первым после старта одним или двумя байтами передаётся адрес. Адрес всегда передаётся в режиме ведущего.

Девятым тактом после передачи байта передатчику выдаётся подтверждение. Размер адреса и выдача подтверждения определяются программно.

Рис. 238. Протокол шины I²C

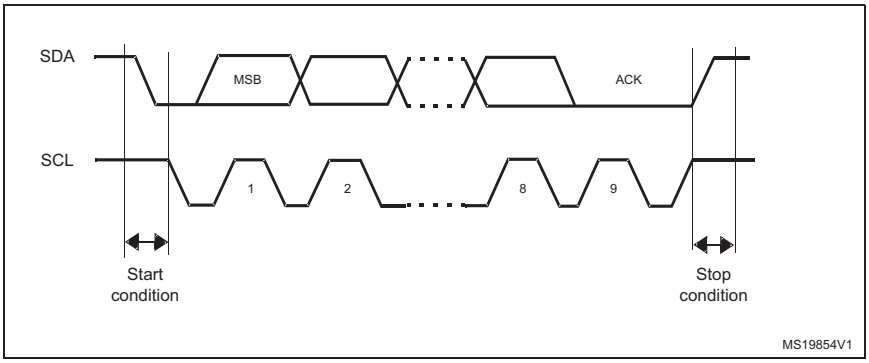


Рис. 239. Блок схема I²C для STM32F40x/41x

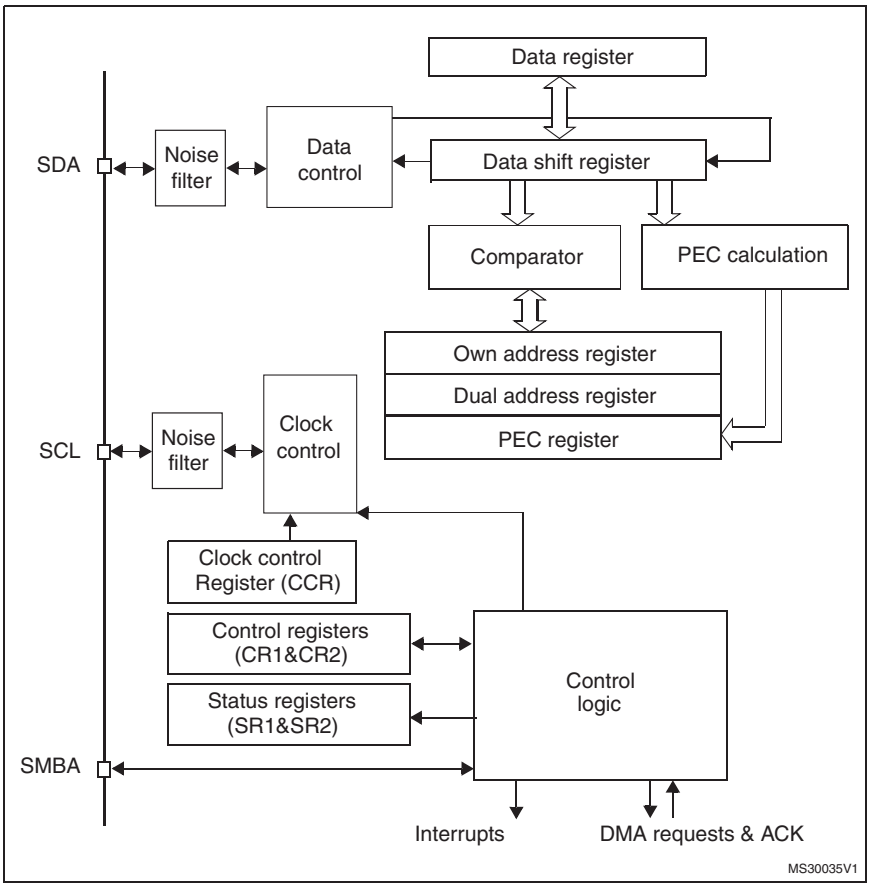
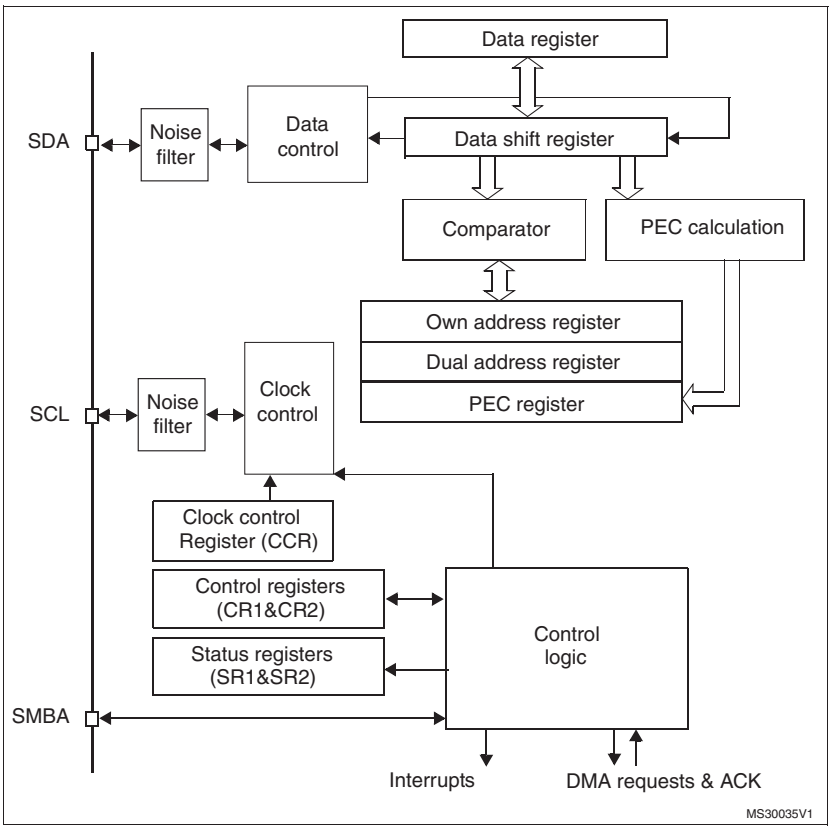


Рис. 240. Блок схема I²C для STM32F42x/43x



1. SMBA работает только при включённой SMBus.

27.3.2. Ведомый I²C

Частота входных тактов программируется в регистре I2C_CR2 и должна быть не менее:

- 2 MHz в режиме Sm
- 4 MHz в режиме Fm

По событию старта с линии SDA в регистр сдвига принимается адрес интерфейса и сравнивается с адресом в OAR1 и с OAR2 (если ENDUAL=1) или адресом Общего Вызова (при ENGC = 1).

NB: В режиме 10-бит адресации сравнение включает заголовок (11110xx0), где xx это старшие биты адреса.

Заголовок или адрес не совпали: игнорируем и ждём нового Старта.

Заголовок совпал (10-бит режим): при стоящем бите ACK выдаём подтверждение и ждём 8-бит адрес ведомого.

Адрес совпал: генерируем последовательность:

- При стоящем бите ACK выдаём подтверждение
- Аппаратно ставится бит ADDR и при стоящем бите ITEVFEN выдаётся прерывание.
- При ENDUAL=1 по биту DUALF уточняем совпавший адрес ведомого.

В 10-бит режиме после приёма адреса ведомый остаётся на Приёме, на Передачу он пойдёт после получения повторного старта с совпадающим адресом и стоящим младшим битом (11110xx1). Бит TRA показывает Приём или Передачу.

Передача ведомого

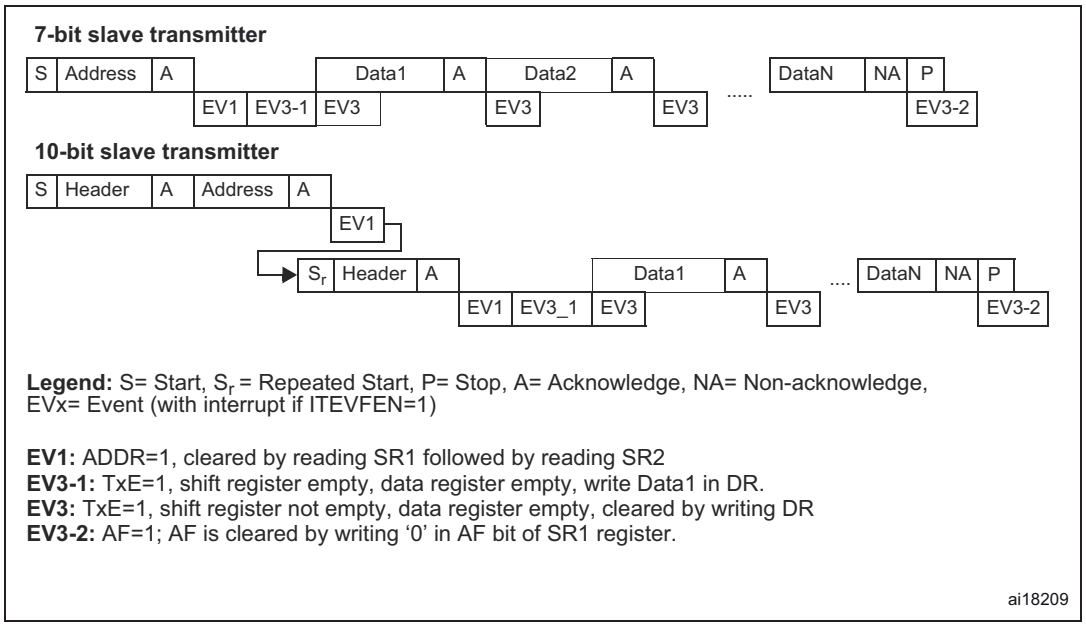
Вслед за приёмом адреса и очисткой ADDR, ведомый через регистр сдвига отсылает байты из регистра DR на линию SDA.

Ведомый растягивает низкий SCL вплоть до очистки ADDR и записи DR (EV1 EV3 в диаграмме).

После приёма подтверждения:

- Аппаратно ставится бит TxE с прерываниями при стоящих битах ITEVFEN и ITBUFEN.
- При стоящем бите TxE, когда до конца передачи в регистр I2C_DR ещё не записан следующий передаваемый байт, ставится бит BTF и растягивается низкий SCL вплоть до очистки BTF чтением I2C_SR1 с последующей записью в регистр I2C_DR.

Рис. 241. Последовательность передачи ведомым



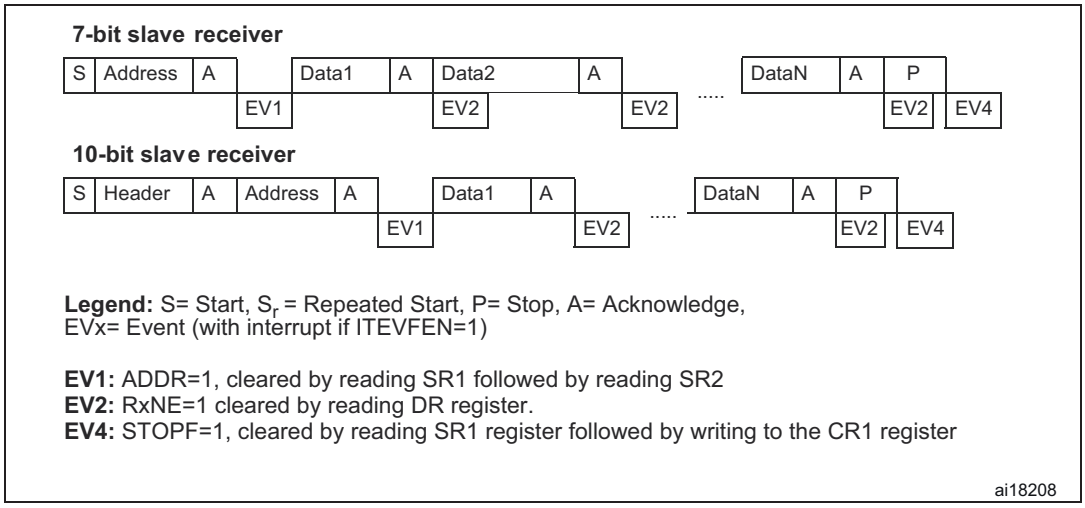
Приём ведомым

Следом за приёмом адреса после очистки ADDR ведомый из линии SDA через регистр сдвига принимает байты в регистр DR. После каждого байта:

- При стоящем бите ACK выдаётся импульс подтверждения
- Аппаратно ставится бит RxNE с прерываниями при стоящих битах ITEVFEN и ITBUFEN.

При стоящем бите RxNE, когда до конца приёма следующего байта регистр DR ещё не прочитан, ставится бит BTF и растягивается низкий SCL вплоть до очистки BTF чтением регистра I2C_SR1 с последующим чтением регистра I2C_DR.

Рис. 242. Последовательность приёма ведомым



1. Событие EV1 растягивает низкий SCL до конца соответствующей программной последовательности.
2. Программное событие EV2 должно завершиться до конца текущей.
3. При проверке регистра SR1 надо полностью обработать и снять все стоящие флаги.

Закрытие связи ведомого

После передачи последнего байта данных ведущий выдаёт сигнал Стоп, и ведомый по стоящему биту STOPF выдаёт прерывание, если разрешено битом ITEVFEN.

Бит STOPF снимается чтением регистра SR1 с последующей записью в регистр CR1 (см. EV4).

27.3.3. Ведущий I²C

В режиме ведущего I²C инициирует передачу данных и выдаёт тактовый сигнал. Передача данных начинается сигналом Старт и завершается сигналом Стоп.

Режим ведущего включается по сигналу Старт на шине, посылаемым битом START.

Ведущий должен:

- Установить времянку входных тактов регистром I2C_CR2
- Установить регистры управления тактами
- Установить регистр времени подъёма
- Включить периферию в регистре I2C_CR1
- Поставить бит START в регистре I2C_CR1 для выдачи сигнала Старт

Частота тактов должна быть не меньше:

- 2 MHz в режиме Sm
- 4 MHz в режиме Fm

Выдача тактов SCL

Выдачу высокого и низкого уровня SCL определяют биты CCR, начиная с переднего фронта. Поскольку ведомый может затягивать такты линии SCL, то I²C проверяет вход SCL в конце времени, заданного битами TRISE после переднего фронта тактового сигнала.

- Если линия SCL низкая, то есть ведомый растягивает такты, то счётчик высокого уровня стопорится до появления высокого уровня SCL. Так обеспечивается минимальный период параметра HIGH такта SCL.
- Если линия SCL высокая, то счётчик высокого уровня продолжает считать.

В действительности, от выдачи переднего фронта SCL до его обнаружения в ответе проходит некоторое время, даже если ведомый не растягивает такты. Длительность задержки связана с временем подъёма SCL (влияние обнаружения SCL VIH), плюс задержка входного аналогового фильтра шума линии SCL, плюс задержка внутренней синхронизации SCL и тактов APB. Максимальная задержка определена в битах TRISE, так что частота SCL остаётся стабильной.

Сигнал Старт

При чистом бите BUSY установка бита START выводит сигнал Старт и переключает в режим ведущего (стоит бит MSL).

NB: В режиме ведущего установка бита START приводит к Рестарту в конце передачи текущего байта.

После посылки Старта аппаратно ставится бит SB и при стоящем бите ITEVFEN выдаётся прерывание. Затем ведущий ждёт чтения регистра SR1 с последующей записью в регистр DR адреса ведомого (см, EV5).

Передача адреса ведомого

Далее на SDA выдвигается адрес ведомого.

- При 10-бит адресации посылка заголовка вызывает следующие события:
  - Аппаратно ставится бит ADD10 и при разрешении битом ITEVFEN выдаётся прерывание. Затем ведущий ждёт чтения регистра SR1 с последующей записью второго байта адреса.
  - Аппаратно ставится бит ADDR и при разрешении битом ITEVFEN выдаётся прерывание. Затем ведущий ждёт чтения регистра SR1 с последующим чтением регистра SR2.
- При 7-бит адресации после посылки байта аппаратно ставится бит ADDR и при разрешении битом ITEVFEN выдаётся прерывание. Затем ведущий ждёт чтения регистра SR1 с последующим чтением регистра SR2.

Переход ведущего в Передачу или Приём задаёт младший бит посланного адреса ведомого.

- При 7-бит адресации:
  - Для режима Передачи младший бит адреса ведомого сброшен.
  - Для режима Приёма младший бит адреса ведомого стоит.
- При 10-бит адресации:
  - Для режима Передачи ведущий посылает заголовок (11110xx0) и затем адрес ведомого (xx это два старших бита адреса).
  - Для режима Приёма ведущий посылает заголовок (11110xx0) и затем адрес ведомого. Далее он должен послать повторный Старт с заголовком (11110xx1) (xx это два старших бита адреса).

Режим приёма или передачи ведущего показывает бит TRA.

Ведущий передатчик

Байты данных посылаются из регистра DR после передачи адреса и очистки ADDR.

Ведущий ждёт записи первого байта в I2C_DR (событие EV8_1).

После приёма импульса подтверждения аппаратно ставится бит TxE и при разрешении битами ITEVFEN и ITBUFEN выдаётся прерывание.

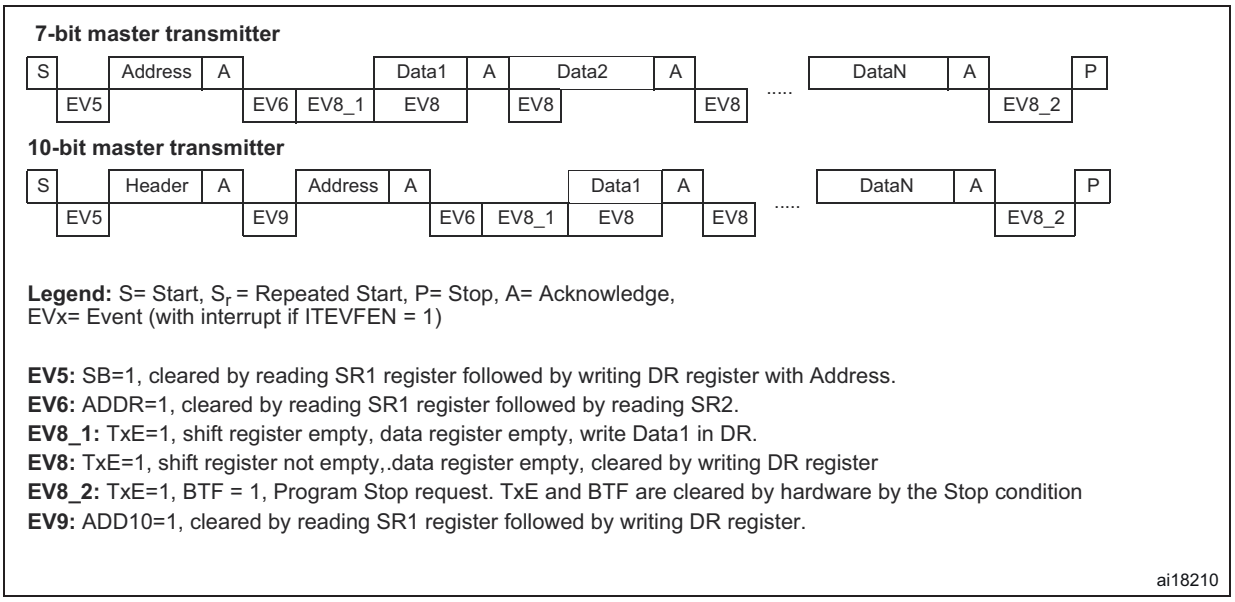
При стоящем бите TxE, когда до конца передачи текущего байта регистр DR ещё не записан, ставится бит BTF и растягивается низкий SCL вплоть до очистки BTF чтением регистра I2C_SR1 с последующей записью регистра I2C_DR.

Закрытие связи

После записи в регистр DR последнего байта, программа ставит бит STOP для выдачи сигнала Стоп (событие EV8_2). Аппарат возвращается в режим ведомого (снимается бит MSL).

NB: Бит STOP во время EV8_2 надо писать при стоящем бите TxE либо BTF.

Рис. 243. Последовательность работы ведущего передатчика.



1. EV5, EV6, EV9, EV8_1 и EV8_2 держат SCL низким до конца программной последовательности.
2. EV8 держит SCL низким если программная последовательность не завершилась до конца передачи следующего байта.

Ведущий приёмник

I²C переключается в ведущего после передачи адреса и очистки ADDR, теперь он принимает данные в регистр DR. После каждого байта:

1. При стоящем бите ACK выдаёт импульс подтверждения
2. При стоящем бите RxNE и разрешении битами ITEVFEN и ITBUFEN выдаёт прерывание (событие EV7).

При стоящем бите RxNE, когда до конца приёма следующего байта регистр DR ещё не прочитан, ставится бит BTF и растягивается низкий SCL вплоть до очистки BTF чтением регистра I2C_SR1 с последующим чтением регистра I2C_DR.

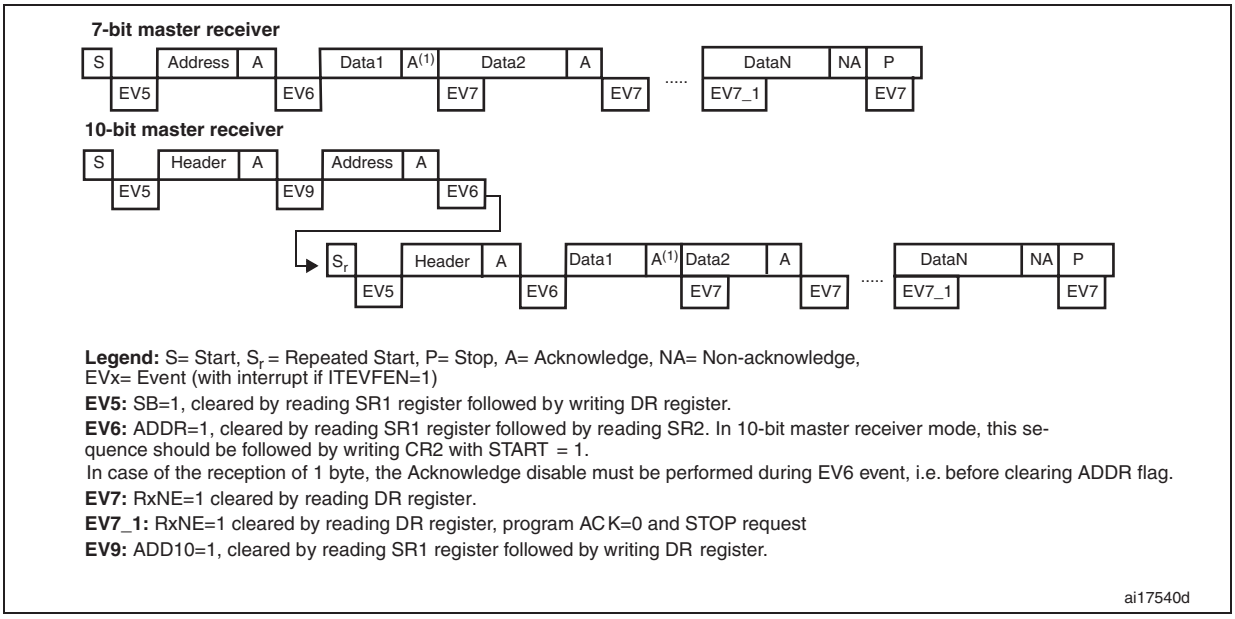
Закрытие связи

В ответ на последний принятый байт, ведущий посылает NACK и ведомый освобождает линии SCL и SDA. Теперь ведущий может сигнал Стоп/Рестарт.

1. Для посылки NACK нужно снять бит ACK сразу после приёма предпоследнего байта (после предпоследнего RxNE).
2. Для выдачи сигнала Стоп/Рестарт надо поставить бит STOP/START сразу после чтения предпоследнего байта (после предпоследнего RxNE).
3. В случае приёма одного байта, снятие подтверждения и выдача Стоп выполняются сразу после EV6 (в EV6_1, сразу после очистки ADDR).

После выдачи сигнала Стоп I²C переходит в режим ведомого (бит MSL снимается).

Рис. 244. Последовательность приёма ведущим.



1. Если принят один байт, то он NA.
2. События EV5, EV6 и EV9 растягивают низкий SCL до конца программной последовательности.
3. Программная последовательность EV7 должна завершиться до конца передачи текущего байта. Если это невозможно, то вместо RXNE надо использовать BTF.
4. Программные последовательности EV6_1 и EV7_1 должны завершиться до выдачи импульса ACK текущей передачи байта.

Если программная последовательность `EV7-1` не завершена до импульса `ACK` текущей передачи байта, то лучше убедиться, что:

- Бит `ACK` стал низким до конца приёма последнего данного
- Бит `ACK` стал высоким после приёма последнего данного без приёма дополнительных данных.

Для приёма двух байтов:

- Ждём `ADDR` = 1 (при снятом `ADDR` SCL остаётся низким)
- Пишем `ACK` низким, `POS` высоким
- Чистим флаг `ADDR`
- Ждём `BTF` = 1 (`Data1` в `DR`, `Data2` в регистре сдвига, SCL низкий до чтения `Data1`)
- Пишем `STOP` высоким
- Читаем `Data1` и `Data2`

Для приёма N >2 байтов, от N-2 данных

- Ждём `BTF` = 1 (`DataN-2` в `DR`, `DataN-1` в регистре сдвига, SCL низкий до чтения `DataN-2`)
- Пишем `ACK` низким
- Читаем `DataN-2`
- Ждём `BTF` = 1 (`DataN-1` в `DR`, `DataN` в регистре сдвига, SCL низкий до чтения `DataN-1`)
- Пишем `STOP` высоким
- Читаем `DataN-1` и `DataN`

27.3.4. Ошибки

Эти ошибки могут привести к сбою связи.

Ошибка шины (`BERR`)

Появляется при обнаружении I²C внешнего сигнала Старт или Стоп при передаче адреса или данных. В этом случае:

- Ставится бит `BERR` и при разрешении битом `ITERREN` выдаётся прерывание
- В режиме ведомого: данные отбрасываются и линии аппаратно освобождаются:
  - при несвоевременном Старт ведомый считает это Рестартом и ждёт адрес или сигнал Стоп
  - при несвоевременном Стоп ведомый, как и при нормальном Стоп освобождает линии

- В режиме ведущего: линии не освобождаются и текущая операция не прекращается, над этим пусть думает программа.

Сбой подтверждения (AF)

Возникает при обнаружении бита не-подтверждения. В этом случае:

- ставится бит AF и при разрешении битом ITERREN выдаётся прерывание
- передатчик, получивший NACK, должен сбросить связь:
  - Ведомый: аппаратно освобождает линии
  - Ведущий: программа должна выдать Стоп или Рестарт

Потеря арбитража (ARLO)

Обнаруживается I²C. В этом случае:

- аппаратно ставится бит ARLO (прерывание зависит от разрешения битом ITERREN)
- I²C автоматически возвращается в ведомый (снимается бит MSL). При потере арбитража I²C не может подтверждать адрес ведомого вплоть до Рестарта от ведущего-победителя.
- линии аппаратно освобождаются

Переполнение/Исчерпание (OVR)

Переполнение возникает при отключённой растяжке тактов, когда принимается байт (RxNE=1) при непрочитанном регистре DR. В этом случае:

- Последний принятый байт теряется.
- Программа должна очистить бит RxNE и передатчику надо повторить передачу.

Исчерпание возникает у ведомого передатчика при отключённом растягивании тактов. В регистр DR не был записан следующий байт (TxE=1), а такт за ним пришёл. В этом случае:

- Будет послан тот же байт из регистра DR
- Программа должна выбрасывать байты, полученные при состоянии исчерпания, и следующие записаны во время низкого такта, определённого стандартом шины I²C bus.

Первый передаваемый байт должен быть записан в DR после снятия ADDR до первого переднего фронта SCL. Если это невозможно, то первый байт надо выбрасывать.

27.3.5. Программируемый цифровой фильтр шума

Он есть только на STM32F42xxx и STM32F43xxx.

В режиме Fm стандарт I²C требует подавления пиков на линиях SDA и SCL.

На ножках SDA и SCL есть аналоговые фильтры шума, выключают их битом ANOFF в регистре I2C_FLTR.

Цифровой фильтр включают не-нулевым содержимым битов DNF[3:0]. Од давит всплески на линиях SDA и SCL длиной до DNF[3:0] * TPCLK1. При этом увеличивается время удержания SDA на (DNF[3:0] +1)* TPCLK.

Для совместимости со спецификацией шины I²C версии 2.1 (Thd:dat), биты DNF должны следовать ограничениям из Таблицы 122 при выключенном аналоговом фильтре.

NB: DNF[3:0] надо ставить при выключенном I²C (PE = 0). Цифровой фильтр добавляется к аналоговому.

Таблица 122. Максимальные значения DNF[3:0] для совместимости с Thd:dat(max)

Частота PCLK1	Максимальное значение DNF	
	Режим Sm	Режим Fm
2 <= F_{PCLK1} <= 5	2	0
5 < FPCLK1 <= 10	12	0
10 < F_{PCLK1} <= 20	15	1
20 < F_{PCLK1} <= 30	15	7
30 < F_{PCLK1} <= 40	15	13
40 < F_{PCLK1} <= 50	15	15

NB: Для каждого диапазона частот ограничения даны для минимальной частоты диапазона. Большие значения DNF можно использовать если система поддерживает нарушение максимального времени удержания.

27.3.6. Линии SDA/SCL

- Если растягивание тактов разрешено:
  – Передача: Если TxE=1 и BTF=1: перед передачей I²C должен удерживать низкий такт, чтобы MCU успел прочитать SR1, и затем писать байт в DR (буфер и регистр сдвига пусты).
  – Приём: Если RxNE=1 и BTF=1: после приёма I²C держит линию тактов низкой, чтобы MCU успел прочитать SR1, и затем прочесть байт из DR (буфер и регистр сдвига полные).
- Если растягивание тактов в режиме ведомого запрещено:
  – Ошибка переполнения при RxNE=1 и не читался DR до приёма следующего байта. Последний принятый байт теряется.
  – Ошибка исчерпания при TxE=1 и ещё не писался DR для передачи, а пора передавать. Посылается тот же байт.
  – Коллизия записи не управляется.

27.3.7. SMBus

Введение

Двухпроводная шина управления системой (SMBus) основана на принципах работы I²C.

В SMBus определены три типа устройств. *Ведомый* принимает и отвечает на команды. *Ведущий* выдаёт команды и такты и завершает передачи. *Хост* это специализированный ведущий для связи с CPU системы. Хост должен быть ведущим-ведомым и поддерживать протокол нотификации хоста SMBus. Хост в системе один.

Совпадения SMBus и I²C

- 2 провода шины (1 такты, 1 данные) + необязательная линия извещения SMBus
- Связь ведущий-ведомый, Такты выдаёт ведущий
- Несколько ведущих
- Формат данных SMBus подобен формату 7-бит адресации I²C.

Отличия SMBus и I²C

Таблица 123. SMBus против I²C

SMBus	I²C
Макс. частота 100 kHz	Макс. частота 400 kHz
Мин. частота 10 kHz	Не ограничена
35 ms таймаут низкого такта	Таймаута нет
Логические уровни фиксированы	Логические уровни зависят от V_{DD}
Разные типы адресации (резервный, динамический и пр.)	7-бит, 10-бит и Общий Вызов
Разные протокола шины (быстрая команда, вызов процесса и пр.)	Протокола шины нет

Использование SMBus

Через SMBus устройство выдаёт информацию производителя, номер модели/части, сохраняет своё состояние для режима задержки, извещает об ошибке, получает параметры управления и возвращает своё состояние.

Идентификация устройства

Все устройства на SMBus имеют свой уникальный адрес ведомого. Список резервных адресов ведомых выложен в спецификации SMBus версии 2.0 (*http://smbus.org/*).

Протоколы шины

SMBus поддерживает до девяти протоколов шины они реализуются программно.

Протокол разрешения адресов (ARP)

Конфликты адресов ведомых SMBus разрешаются динамически назначением новых адресов. Протокол Разрешения Адресов (ARP) имеет атрибуты:

- Назначение адресов использует стандартный механизм арбитража физического уровня SMBus

- Назначенные адреса сохраняются при включённом питании, можно и при выключенном.
- Назначенные адреса SMBus не требуют дополнительных пакетов шины по сравнению с фиксированными.
- Любой ведущий шины SMBus может выполнять нумерацию устройств.

Уникальный идентификатор устройства (UDID)

Все устройства должны иметь свой уникальный 128-бит идентификатор устройства (UDID).

Все подробности приведены в спецификации SMBus версии 2.0 (*http://smbus.org/*).

Режим извещения SMBus

Необязательная линия извещения с прерыванием SMBus (SMBA) это объединение по И сигналов устройств. SMBA используется совместно с адресом Общего Вызова. Сообщения, работающие с SMBus двухбайтовые.

Только ведомые устройства через SMBA извещают хост о желании побеседовать установкой бита ALERT в регистре I2C_CR1. Хост обрабатывает прерывание и одновременно обращается ко всем SMBA устройствам с *Адресом Ответа Извещения* (ARA равным 0001 100X). На адрес отвечают только устройства, удерживающие низким SMBA. Этому служит флаг SMBALERT регистра I2C_SR1. Хост исполняет модифицированный Приём Байта. 7-битный адрес от ведомого передатчика лежит в старших битах байта, младший бит может быть любым.

Из нескольких запрашивающих устройств с низким SMBA связь получает устройство с меньшим адресом в соответствии со стандартным арбитражем при передаче адреса ведомого. После подтверждения адреса ведомого, устройство снимает свой сигнал SMBA. Если SMBA ещё низкий, то хоста заново обращается к ARA.

Хосты без сигнала SMBA могут обращаться к ARA периодически.

Ошибка таймаута

Таймаут I²C и SMBus различается. SMBus определяет таймаут тактов TIMEOUT в 35 ms.

TLOW: SEXT это суммарное время растяжки низкого такта ведомого.

TLOW: MEXT это суммарное время растяжки низкого такта ведущего.

На ошибку таймаута TLOW указывает флаг TIMEOUT в регистре I2C_SR1.

Работа в режиме SMBus

Переход из режима I²C в режим SMBus:

- Ставим бит SMBUS регистра I2C_CR1
- Пишем биты SMBTYPE и ENARP регистра I2C_CR1

Конфигурация ведущим выполняется процедурой Старт, см. Секцию 26.3.3, иначе см. Секцию 26.3.2. Протоколы SMBus реализуются программно.

- SMB Device Default Address подтверждается если ENARP=1 и SMBTYPE=0
- SMB Host Header подтверждается если ENARP=1 и SMBTYPE=1
- SMB Alert Response Address подтверждается если SMBALERT=1

27.3.8. Запросы DMA

Разрешённые запросы DMA используются только для передачи данных. Они выдаются пустым DR при передаче и полным DR при приёме. DMA инициализируют и включают до передачи данных. Бит DMAEN регистра I2C_CR2 ставят до события ADDR. При разрешённой растяжке тактов бит DMAEN можно ставить во время события ADDR, перед очисткой флага ADDR. Запрос DMA нужно обслужить до конца текущей передачи байта. При достижении установленного числа передач DMA контроллер DMA посылает I²C сигнал EOT (Конец Передачи), чем может вызвать прерывание:

- Ведущий передатчик: обработчик прерывания EOT выключает запросы DMA и ждёт события BTF для выдачи сигнала Стоп.
- Ведущий приёмник:
  - Если число передаваемых байтов больше или равно двум, то для предпоследнего байта контроллер DMA посылает аппаратный сигнал EOT_1. Если в регистре I2C_CR2 стоит бит LAST, то I²C автоматически посылает a NACK для следующего за EOT_1 байта. Сигнал Стоп можно выдавать из обработчика прерывания EOT.

- При приёме одного байта: NACK надо писать во время события EV6, т.е. писать ACK=0 до снятия стоящего флага ADDR. Сигнал STOP можно выдавать после чистки флага ADDR или в программе обработки прерывания конца передачи DMA.

Передача с DMA

Включается установкой бита DMAEN в регистре I2C_CR2. Данные пишутся из памяти в I2C_DR по установке бита TxE. Последовательность подключения потока DMA X (где X это номер потока) к передаче I²C такова:

1. Пишем адрес регистра I2C_DR в регистр DMA_SxPAR. Сюда по событию TxE будут писаться данные из памяти.
2. Адрес памяти пишем в регистр DMA_SxMA0R (и в DMA_SxMA1R при двух буферах). Отсюда по событию TxE будут писаться данные в I2C_DR.
3. Общее число передаваемых байт пишем в регистр DMA_SxNDTR. По каждому событию TxE оно декрементируется.
4. Приоритет потока DMA пишем в биты PL[0:1] регистра DMA_SxCR.
5. Ставим бит DIR регистра DMA_SxCR, а прерывания половины и полной передачи, как нам надо.
6. Активируем поток установкой бита EN в регистре DMA_SxCR.

При достижении установленного числа передач, контроллер DMA посылает I²C сигнал EOT/EOT_1 и выдаёт прерывание по вектору потока DMA.

NB: Не ставьте бит ITBUFEN регистра I2C_CR2 при передаче с DMA.

Приём с DMA

Включается установкой бита DMAEN в регистре I2C_CR2. Данные из I2C_DR пишутся в память по приёму байта. Последовательность подключения потока DMA X (где X это номер потока) к приёму I²C такова:

1. Пишем адрес регистра I2C_DR в регистр DMA_SxPAR. Отсюда по событию RxNE будут писаться данные в память.
2. Адрес памяти пишем в регистр DMA_SxMA0R (и в DMA_SxMA1R при двух буферах). Сюда по событию RxNE будут писаться данные из I2C_DR.
3. Общее число передаваемых байт пишем в регистр DMA_SxNDTR. По каждому событию RxNE оно декрементируется.
4. Приоритет потока DMA пишем в биты PL[0:1] регистра DMA_SxCR.
5. Чистим бит DIR регистра DMA_SxCR, а прерывания половины и полной передачи, как нам надо.
6. Активируем поток установкой бита EN в регистре DMA_SxCR.

При достижении установленного числа передач, контроллер DMA посылает I²C сигнал EOT/EOT_1 и выдаёт прерывание по вектору потока DMA.

NB: Не ставьте бит ITBUFEN регистра I2C_CR2 во время приёма с DMA.

27.3.9. Контроль ошибки пакета

Контроль ошибки пакета PEC вычисляется по полиному CRC-8 $C(x) = x^8 + x^2 + x + 1$.

- Вычисление PEC включается установкой бита ENPEC регистра I2C_CR1. PEC вычисляется для всех байтов пакета, включая адреса и биты R/W.
  - При передаче: бит передачи PEC в I2C_CR1 ставят после события TxE последнего байта.
  - При приёме: бит PEC в I2C_CR1 ставят после события RxNE последнего байта, так что при отличии следующего байта от принятого PEC приёмник пошлёт NACK. У Ведущего приёмника NACK выдаётся независимо от результата сравнения PEC. PEC должен стоять до импульса ACK приёма текущего байта.
- Флаг/прерывание PECERR лежит в регистре I2C_SR1.
- Если разрешены и DMA и PEC:
  - При передаче: PEC автоматически посылается после получения сигнала EOT от DMA.
  - При приёме: после получения сигнала EOT_1 от DMA, следующий байт воспринимается как PEC и проверяется. Запрос DMA выдаётся после приёма PEC.

- Промежуточной передаче PEC помогает бит LAST регистра I2C_CR2, показывающий действительно последнюю передачу DMA. После последнего принятого байта DMA ведущего приёмника NACK посылается автоматически.
- Вычисление PEC разрушается потерей арбитража.

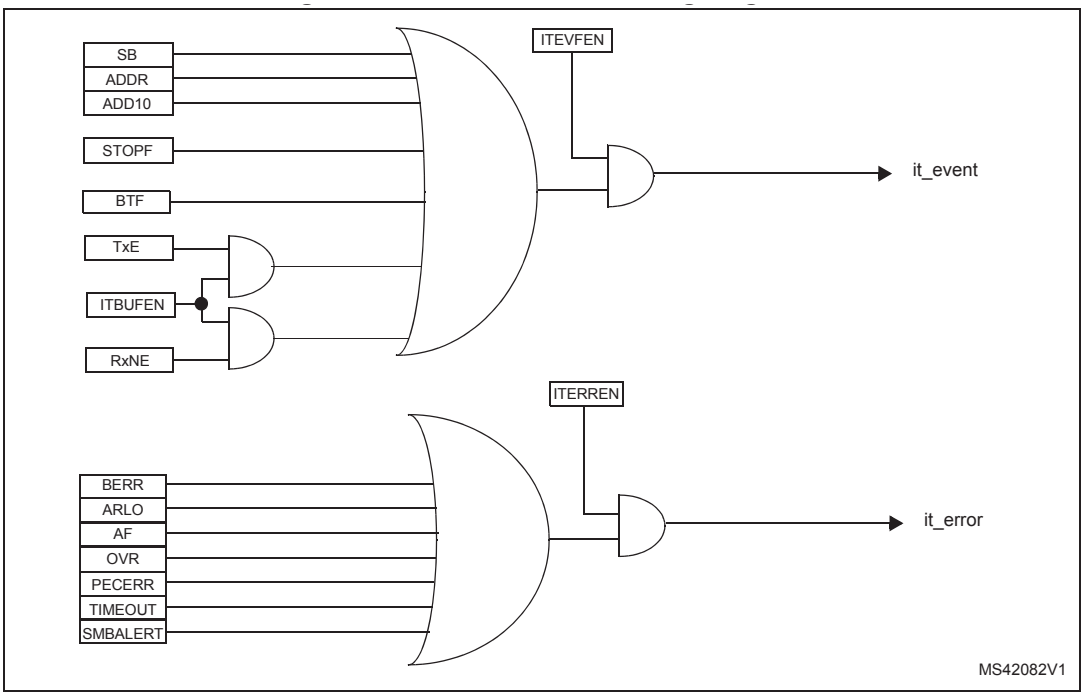
27.4. Прерывания I²C

Таблица 124. Запросы прерываний I²C

Событие	Флаг	Бит разрешения
Бит Старт послан (Ведущий)	SB	ITEVFEN
Адрес послан (Ведущий) или Адрес совпал (Ведомый)	ADDR	
10-бит заголовок послан (Ведущий)	ADD10	
Стоп принят (Ведомый)	STOPF	
Байт данных передан	BTF	
Буфер приёма не пуст	RxNE	ITEVFEN и ITBUFEN
Буфер передачи пуст	TxE	
Ошибка шины	BERR	ITERREN
Арбитраж потерян loss ()	ARLO	
Сбой подтверждения	AF	
Переполнение/Исчерпание	OVR	
Ошибка PEC	PECERR	
Ошибка Timeout/Tlow	TIMEOUT	
Предупреждение SMBus	SMBALERT	

NB: SB, ADDR, ADD10, STOPF, BTF, RxNE и TxE логически складываются в один канал. BERR, ARLO, AF, OVR, PECERR, TIMEOUT и SMBALERT логически складываются в другой канал.

Рис. 245. Прерывания I²C



27.5. Отладка I²C

В зависимости от битов конфигурации DBG_I2Cx_SMBUS_TIMEOUT модуля DBG таймаут SMBus может останавливаться или работать дальше.

27.6. Регистры I²C

Регистры доступны полусловами (16 бит) или словами (32 бита).

27.6.1. Регистр 1 управления I²C (I2C_CR1)

Смещение адреса: 0x00

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWRST	Res.	ALERT	PEC	POS	ACK	STOP	START	NO STRETCH	ENGC	ENPEC	ENARP	SMB TYPE	Res.	SMBUS	PE
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

— Бит 15 **SWRST**: Программный сброс

При снятии этого бита, линии и шина I²C должны быть освобождены.

0: I²C не под сбросом

1: I²C под сбросом

NB: Может использоваться для инициализации шины после ошибки или блокирования шины, например, при установке бита BUSY после глитча на шине.

— Бит 14 Резерв, не трогать.

— Бит 13 **ALERT**: Извещение SMBus

Ставится и снимается программно, аппаратно снимается при PE=0.

0: Освобождает ножку SMBA высокой. Заголовок Ответа Адреса сопровождается NACK.

1: Ставит ножку SMBA низкой. Заголовок Ответа Адреса сопровождается ACK.

— Бит 12 **PEC**: Контроль ошибки пакета

Ставится и снимается программно, аппаратно снимается после передачи PEC, сигналами Старт и Стоп и PE=0.

0: PEC не передаётся

1: PEC передаётся (в режимах Tx и Rx)

NB: Вычисление PEC разрушается потерей арбитража.

— Бит 11 **POS**: Позиция Подтверждения/PEC (для приёма данных)

Ставится и снимается программно, аппаратно снимается при PE=0.

0: Бит ACK определяет выдачу (N)ACK по приёму текущего байта в регистр сдвига. Бит PEC показывает, что в регистре сдвига лежит PEC.

1: Бит ACK определяет выдачу (N)ACK на следующий байт в регистре сдвига. Бит PEC показывает, что следующим в регистре сдвига будет PEC.

NB: Бит POS используется при приёме 2 байтов (см. Метод 2). Его надо писать до начала приёма данных. В этом случае, для выдачи NACK на второй байт, бит ACK надо снимать сразу после снятия ADDR. Для опознания второго байта как PEC, бит PEC должен стоять во время растяжки события ADDR после записи бита POS.

— Бит 10 **ACK**: Разрешение подтверждения

Ставится и снимается программно, аппаратно снимается при PE=0.

0: Подтверждение не выдаётся

1: Подтверждение выдаётся после приёма байта (совпал адрес или данные)

— Бит 9 **STOP**: Выдача сигнала Стоп

Ставится и снимается программно, аппаратно снимается при обнаружении сигнала Стоп, аппаратно ставится при обнаружении ошибки таймаута.

У ведущего:

0: Стоп не выдаётся.

1: Стоп выдаётся после передачи текущего байта или после текущей посылки Старт.

У ведомого:

0: Стоп не выдаётся.

1: Освобождаются линии SCL и SDA после передачи текущего байта.

— Бит 8 **START**: Выдача Старт

Ставится и снимается программно, аппаратно снимается при посылки Старта или PE=0.

У ведущего:

0: Старт не выдаётся

1: Повторная Выдача Старт

У ведомого:

0: Старт не выдаётся

1: Старт выдаётся при свободной шине

— Бит 7 **NOSTRETCH**: Запрет растяжки такта (Режим ведомого)

Запрещает растяжку такта ведомым при стоящем ADDR или флаге BTF до программного сброса.

0: Растягивать можно

1: Растягивать нельзя

— Бит 6 **ENGC**: Разрешение Общего Вызова

0: Запрещён. На адрес 00h выдаётся NACK.

1: Разрешён. На адрес 00h выдаётся ACK.

— Бит 5 **ENPEC:** Разрешение вычисления PEC

0: Выключено

1: Включено

— Бит 4 **ENARP**: Разрешение ARP

0: ARP выключен

1: ARP включён

Адрес устройства SMBus опознаётся при SMBTYPE=0

Адрес хоста SMBus опознаётся при SMBTYPE=1

— Бит 3 **SMBTYPE**: Тип SMBus

0: Устройство SMBus

1: Хост SMBus

— Бит 2 Резерв, не трогать.

— Бит 1 **SMBUS**: Режим SMBus

0: Режим I²C

1: Режим SMBus

— Бит 0 **PE**: Включение

0: Выключение

1: Включение

NB: Снятие бита при работающей связи выключает устройство и сбрасывает все биты после её завершения, возврат в состояние IDLE.

У ведущего бит нельзя снимать до конца обмена.

NB: При стоящих битах STOP, START или PEC, нельзя писать в регистр I2C_CR1 вплоть до аппаратной очистки этого бита. Иначе можно получить второй запрос STOP, START или PEC.

27.6.2. Регистр 2 управления I²C (I2C_CR2)

Смещение адреса: 0x04

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			LAST	DMAEN	ITBUFEN	ITEVTEN	ITERREN	Reserved		FREQ[5:0]					
			rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

— Биты 15:13 Резерв, не трогать.

— Бит 12 **LAST**: Последняя передача DMA

0: Следующий EOT от DMA не последняя передача

1: Следующий EOT от DMA последняя передача

NB: Используется ведущим приёмником для разрешения выдачи NACK по последнему данному.

— Бит 11 **DMAEN**: Разрешение запросов DMA

0: Запрос DMA запрещён

1: Запрос DMA при TxE=1 или RxNE =1 разрешён

— Бит 10 **ITBUFEN**: Разрешение прерывания буфера

0: TxE = 1 или RxNE = 1 не выдают прерывания.

1: TxE = 1 или RxNE = 1 выдают прерывание Event (при любом DMAEN)

— Бит 9 **ITEVTEN**: Разрешение прерывания Event

0: Event interrupt disabled

1: Event interrupt enabled

Прерывание выдаётся при:

– SB = 1 (Ведущий)

– ADDR = 1 (Ведущий/Ведомый)

- ADD10= 1 (Ведущий)
- STOPF = 1 (Ведомый)
- BTF = 1 без события TxE или RxNE
- TxE = 1 и ITBUFEN = 1
- RxNE = 1 и ITBUFEN = 1

— Бит 8 **ITERREN**: Разрешение прерывания ошибки

0: Error interrupt disabled

1: Error interrupt enabled

Прерывание выдаётся при:

- BERR=1
- ARLO=1
- AF=1
- OVR=1
- PECERR = 1
- TIMEOUT = 1
- SMBALERT = 1

— Биты 7:6 Резерв, не трогать.

— Биты 5:0 **FREQ[5:0]**: Частота тактов периферии

В биты FREQ пишется значение частоты тактов APB (I²C подключён к APB). Поле FREQ используется для обеспечения времён установки и удержания данных в соответствии с спецификацией I²C. Минимальная частота 2 MHz, максимальная - 50 MHz (максимальная внутренняя).

0b000000: Не дозволено

0b000001: Не дозволено

0b000010: 2 MHz

…

0b110010: 50 MHz

Больше 0b100100: Не дозволено

27.6.3. Регистр 1 собственного адреса I²C (I2C_OAR1)

Смещение адреса: 0x08

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADD MODE	Reserved					ADD[9:8]		ADD[7:1]							ADD0
rw						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Бит 15 **ADDMODE**: Режим адресации (Ведомый)

0: 7-бит ведомый

1: 10-бит ведомый

— Бит 14 Программно держится в 1

— Биты 13:10 Резерв, не трогать.

— Биты 9:8 **ADD[9:8]**: Адрес интерфейса

7-бит адрес: всё равно

10-бит адрес: биты 9:8 адреса

— Биты 7:1 **ADD[7:1]**: Биты 7:1 адреса

— Бит 0 **ADD0**: Адрес интерфейса

7-бит адрес: всё равно

10-бит адрес: бит 0 адреса

27.6.4. Регистр 2 собственного адреса I²C (I2C_OAR2)

Смещение адреса: 0x0C

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								ADD2[7:1]							ENDUAL
								rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:8 Резерв, не трогать.
— Биты 7:1 **ADD2[7:1]**: Биты 7:1 адреса при двойном адресе
— Бит 0 **ENDUAL**: Разрешение двух адресов
0: Только OAR1 при 7-бит адресации
1: И OAR1 и OAR2 при 7-бит адресации

27.6.5. Регистр данных I²C (I2C_DR)

Смещение адреса: 0x10

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								DR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:8 Резерв, не трогать.
— Биты 7:0 **DR[7:0]**: 8-бит регистр данных
– Передача: Начинается сразу после записи байта в регистр DR. Непрерывная передача возникает при записи в DR следующего байта после начала передачи (TxE=1)
– Приём: Принятый байт копируется в DR (RxNE=1). Непрерывная передача возникает при чтении DR до завершения приёма следующего байта (RxNE=1).

NB: В режиме ведомого, адрес в DR не копируется.

Коллизия записи не разбирается (DR можно писать при TxE=0).

При событии ARLO на импульс ACK, принятый байт в DR не копируется.

27.6.6. Регистр 1 состояния I²C (I2C_SR1)

Смещение адреса: 0x14

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMB ALERT	TIME OUT	Res.	PEC ERR	OVR	AF	ARLO	BERR	TxE	RxNE	Res.	STOPF	ADD10	BTF	ADDR	SB
rc_w0	rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	r	r		r	r	r	r	r

— Бит 15 **SMBALERT**: Извещение SMBus
Хост SMBus:
0: Нет SMBALERT
1: На ножке есть SMBALERT
Ведомый SMBus:
0: Заголовка адреса ответа SMBALERT нет
1: Заголовок адреса ответа SMBALERT принят
Программно снимается записью 0, аппаратно снимается при PE=0.
— Бит 14 **TIMEOUT**: Ошибка Timeout или Tlow
0: Таймаута нет
1: SCL оставался низким 25 ms (Таймаут),
— Суммарное время растяжки такта ведущего больше 10 ms (Tlow:mext) или
— Суммарное время растяжки такта ведомого больше 25 ms (Tlow:sext)
– У ведомого: сбрасывается связь и аппаратно освобождаются линии
– У ведущего: аппаратно посылается Стоп
Программно снимается записью 0, аппаратно снимается при PE=0.
NB: Работает только в режиме SMBus.
— Бит 13 Резерв, не трогать.
— Бит 12 **PECERR**: Ошибка PEC на приёме
0: ошибки PEC нет: после PEC приёмник возвращает ACK (если ACK=1)
1: ошибка PEC: после PEC приёмник возвращает NACK (при любом ACK)
– Программно снимается записью 0, аппаратно снимается при PE=0.
— Бит 11 **OVR**: Переполнение/Исчерпание
0: Не было
1: Было

– Аппаратно ставится у ведомого при NOSTRETCH=1 и:
– Принят новый байт (включая импульс ACK), а регистр DR ещё не прочитан. Байт теряется.
– Надо посылать новый байт, а регистр DR ещё не записан. Посылается старый.

Программно снимается записью 0, аппаратно снимается при PE=0.

NB: Если DR пишется слишком близко к переднему фронту SCL, то посылаемое данное не определено и возникает ошибка времени удержания.

— Бит 10 **AF**: Сбой подтверждения

0: Не было
1: Было

– Аппаратно ставится при отсутствии подтверждения.
– Программно снимается записью 0, аппаратно снимается при PE=0.

— Бит 9 **ARLO**: Потеря арбитража (Ведущий)

0: Не было
1: Было

Ставится аппаратно при передаче арбитража другому ведущему

– Программно снимается записью 0, аппаратно снимается при PE=0.

После события ARLO I^2C автоматически возвращается в ведомые (MSL=0).

NB: В SMBUS арбитраж данных ведомого появляется только в фазе данных и подтверждении передачи.

— Бит 8 **BERR**: Ошибка шины

0: Старт и Стоп были так, где надо
1: Старт и Стоп появились не там

– Аппаратно ставится при появлении фронта SDA при высоком SCL во время передачи байта.
– Программно снимается записью 0, аппаратно снимается при PE=0.

— Бит 7 **TxE**: Регистр данных пуст (Передача)

0: Регистр данных не пуст
1: Регистр данных пуст

– Ставится при опустении DR для передачи. В фазе адреса TxE не ставится.
– Снимается программно записью в DR или аппаратно сигналами Старт и Стоп, или при PE=0.

TxE не ставится при получении NACK и перед передачей PEC (PEC=1)

NB: TxE не снимается записью первого передаваемого байта или записью при стоящем BTF, так как регистр данных всё равно ещё пуст.

— Бит 6 **RxNE**: Регистр данных не пуст (Приём)

0: Регистр данных пуст
1: Регистр данных не пуст

– Ставится при появлении принятого данного. В фазе адреса RxNE не ставится.
– Снимается программно чтением или записью DR или аппаратно при PE=0.

RxNE не ставится при событии ARLO.

NB: RxNE не снимается чтением при стоящем BTF, так как регистр ещё полон.

— Бит 5 Резерв, не трогать.

— Бит 4 **STOPF**: Обнаружен Стоп (Ведомый)

0: Стопа нету
1: Появился Стоп

– Аппаратно ставится на при обнаружении Стоп на шине ведомого после подтверждения (ACK=1).
– Программно снимается чтением SR1 с последующей записью в CR1, аппаратно при PE=0

NB: Бит STOPF не ставится после приёма NACK.

— Бит 3 **ADD10**: 10-бит заголовок послан (Ведущий)

0: ADD10 не было.
1: Ведущий послал первый байт адреса (заголовок).

– Ставится аппаратно при посылке первого байта 10-бит адреса.
– Программно снимается чтением SR1 с последующей записью в DR второго байта адреса, аппаратно при PE=0.

NB: Бит ADD10 не ставится после приёма NACK.

— Бит 2 **BTF**: Передача байта завершена

0: Передача байта не завершена
1: Передача байта завершена

– Ставится аппаратно при NOSTRETCH=0 и:

– Приём: получен новый байт (включая импульс ACK), а DR ещё не читан (RxNE=1).

– Передача: пора посылать новый байт, а DR ещё не писан (TxE=1).

– Снимается программно чтением SR1 с последующим чтением или записью DR или аппаратно после Старт или Стоп или при стоящем PE=0.

NB: Бит BTF не ставится после приёма NACK.

Бит BTF не ставится перед передачей PEC (TRA=1 в регистре I2C_SR2 и PEC=1 в регистре I2C_CR1)

— Бит 1 **ADDR**: Адрес послан (Ведущий) / Совпал (Ведомый)

Программно снимается чтением SR1 с последующим чтением SR2, аппаратно при PE=0.

Адрес совпал (Ведомый)

0: Адрес не совпал или не принят.

1: Принятый адрес совпал.

– Аппаратно ставится при совпадении полученного адреса ведомого с регистрами OAR или Общим Вызовом, либо с адресом по умолчанию устройства SMBus или опознаны SMBus Host или SMBus Alert.

Адрес послан (Ведущий)

0: Передача адреса не завершена

1: Передача адреса завершена

– При 10-бит адресации ставится после ACK второго байта.

– При 7-бит адресации ставится после ACK байта.

NB: ADDR is not set after a NACK reception

— Бит 0 **SB**: Бит Старт (Ведущий)

Ставится при выдаче Старт, снимается чтением SR1 с записью DR, аппаратно при PE=0.

0: Старту нету

1: Выдача Старта.

27.6.7. Регистр 2 состояния I²C (I2C_SR2)

Смещение адреса: 0x18

По сбросу: 0x0000

NB: Чтение `I2C_SR2` после чтения `I2C_SR1` снимает флаг `ADDR`, даже если `ADDR` встал после чтения `I2C_SR1`. Следовательно, `I2C_SR2` нужно читать только после проверки `ADDR` в `I2C_SR1` или бит `STOPF` снят.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PEC[7:0]								DUALF	SMB HOST	SMBDE FAULT	GEN CALL	Res.	TRA	BUSY	MSL
r	r	r	r	r	r	r	r	r	r	r	r		r	r	r

— Биты 15:8 **PEC[7:0]** Регистр проверки PEC

Содержит внутренний PEC при ENPEC=1.

— Бит 7 **DUALF**: Флаг двойного адреса (Ведомый)

0: Принятый адрес совпал с OAR1

1: Принятый адрес совпал с OAR2

– Снимается аппаратно при Стоп или повторного Старт, или при PE=0.

— Бит 6 **SMBHOST**: Заголовок хоста SMBus (Ведомый)

0: Не было

1: Принят заголовок хоста SMBus при SMBTYPE=1 и ENARP=1.

– Снимается аппаратно при Стоп или повторного Старт, или при PE=0.

— Бит 5 **SMBDEFAULT**: Адрес устройства по умолчанию SMBus (Ведомый)

0: Не было

1: Принят при ENARP=1

– Снимается аппаратно при Стоп или повторного Старт, или при PE=0.

— Бит 4 **GENCALL**: Адрес Общего Вызова (Ведомый)

0: Не было

1: Принят при ENGC=1

– Снимается аппаратно при Стоп или повторном Старт, или при PE=0.

— Бит 3 Резерв, не трогать.

— Бит 2 **TRA**: Передатчик/Приёмник

0: Байт данных принят

1: Байт данных передан

Ставится в зависимости от бита R/W байта адреса в конце фаза адреса.

Снимается аппаратно при Стоп (STOPF=1), повторном Старт, потере арбитража (ARLO=1), при PE=0.

— Бит 1 **BUSY**: Шина занята

0: Свободен

1: Работаем

– Ставится аппаратно при низких SDA или SCL

– Снимается аппаратно при Стоп.

Обновляется даже при PE=0.

— Бит 0 **MSL**: Ведущий/Ведомый

0: Ведомый

1: Ведущий

Ставится аппаратно при SB=1.

Снимается аппаратно при Стоп, потере арбитража (ARLO=1), при PE=0.

27.6.8. Регистр управления тактами I²C (I2C_CCR)

Смещение адреса: 0x1C

По сбросу: 0x0000

NB: f_{PCLK1} должна быть не меньше 2 MHz в режиме Sm, и не меньше 4 MHz в режиме Fm. Шаг равен 10MHz для получения максимальных 400 kHz режима Fm.

Писать надо при выключенном I²C (PE = 0).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F/S	DUTY	Reserved		CCR[11:0]											
rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Бит 15 **F/S:** Режим ведущего I²C

0: режим Sm

1: режим Fm

— Бит 14 **DUTY:** Коэффициент заполнения режима Fm

0: $t_{low}/t_{high} = 2$

1: $t_{low}/t_{high} = 16/9$ (см. CCR)

— Биты 13:12 Резерв, не трогать.

— Биты 11:0 **CCR[11:0]:** Регистр управления тактами режимов Fm/Sm (Ведущий)

Задаёт такты SCL ведущего.

Режим Sm или SMBus:

$T_{high} = CCR * T_{PCLK1}$

$T_{low} = CCR * T_{PCLK1}$

Режим Fm:

При DUTY = 0:

$T_{high} = CCR * T_{PCLK1}$

$T_{low} = 2 * CCR * T_{PCLK1}$

При DUTY = 1: (для достижения 400 kHz)

$T_{high} = 9 * CCR * T_{PCLK1}$

$T_{low} = 16 * CCR * T_{PCLK1}$

Например: в режиме Sm выдаём 100 kHz SCL:

Если FREQR = 08, T_{PCLK1} = 125 ns, то в CCR надо писать 0x28 (0x28 <=> 40d x 125 ns = 5000 ns.)

NB:

Минимальное значение равно 0x04, в режиме FAST DUTY минимум равен 0x01

$t_{high} = t_{r(SCL)} + t_{w(SCLH)}$. См. описание устройства.

$t_{low} = t_{f(SCL)} + t_{w(SCLL)}$. См. описание устройства.

Скорость I²C равна $f_{SCL} \sim 1/(t_{high} + t_{low})$. Реальная частота может отличаться из-за задержки аналогового фильтра на входе.

27.6.9. Регистр TRISE I²C (I2C_TRISE)

Смещение адреса: 0x20

По сбросу: 0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										TRISE[5:0]					
										rw	rw	rw	rw	rw	rw

— Биты 15:6 Резерв, не трогать.

— Биты 5:0 **TRISE[5:0]**: Максимальное время подъёма в режиме Fm/Sm (Ведущий)

Это максимальное время обратной связи SCL в режиме ведущего для стабилизации частоты SCL. Должно быть взято из спецификации шины I²C плюс 1.

Например: в режиме Sm, максимальное время подъёма SCL равно 1000 ns.

Если значение битов FREQ[5:0] в регистре I2C_CR2 равно 0x08 и T_{PCLK1} = 125 ns, то TRISE[5:0] должно быть равно 09h.

(1000 ns / 125 ns = 8 + 1)

В TRISE[5:0] можно добавить значение фильтра.

Если результат не целочисленный, то в TRISE[5:0] надо писать целую часть для соблюдения параметра t_{HIGH}.

NB: TRISE[5:0] можно писать только при PE = 0.

27.6.10.Регистр фильтра FLTR I²C (I2C_ FLTR)

Смещение адреса: 0x24

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											ANOFF	DNF[3:0]			
											rw	rw	rw	rw	rw

— Биты 15:5 Резерв, не трогать.

— Бит 4 **ANOFF**: Выключение аналогового фильтра

Пишут при выключенном I²C (PE = 0)

— Биты 3:0 **DNF[3:0]**: Цифровой фильтр шума на входах SDA и SCL.

Давит всплески до DNF[3:0] * TPCLK1.

0000: Выключен

0001: до 1* TPCLK1.

…

1111: до 15* TPCLK1.

NB: DNF[3:0] пишут при выключенном I²C (PE = 0). Если аналоговый фильтр включён, то цифроввой добавляется к нему.

27.6.11.Карта регистров I²C

Таблица 125. Карта регистров I²C

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**I2C_CR1**	Reserved																SWRST	Reserved	ALERT	PEC	POS	ACK	STOP	START	NOSTRETCH	ENGC	ENPEC	ENARP	SMBTYPE	Reserved	SMBUS	PE
	Reset value																	0		0	0	0	0	0	0	0	0	0	0	0		0	0
0x04	**I2C_CR2**	Reserved																			LAST	DMAEN	ITBUFEN	ITEVTEN	ITERREN	Reserved	FREQ[5:0]						
	Reset value																				0	0	0	0	0		0	0	0	0	0	0	0
0x08	**I2C_OAR1**	Reserved																ADDMODE	Reserved					ADD[9:8]		ADD[7:1]							ADD0
	Reset value																	0						0	0	0	0	0	0	0	0	0	0
0x0C	**I2C_OAR2**	Reserved																								ADD2[7:1]							ENDUAL
	Reset value																									0	0	0	0	0	0	0	0
0x10	**I2C_DR**	Reserved																								DR[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x14	**I2C_SR1**	Reserved																SMBALERT	TIMEOUT	Reserved	PECERR	OVR	AF	ARLO	BERR	TxE	RxNE	Reserved	STOPF	ADD10	BTF	ADDR	SB
	Reset value																	0	0		0	0	0	0	0	0	0		0	0	0	0	0
0x18	**I2C_SR2**	Reserved																PEC[7:0]								DUALF	SMBHOST	SMBDEFAULT	GENCALL	Reserved	TRA	BUSY	MSL
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x1C	**I2C_CCR**	Reserved																F/S	DUTY	Reserved	CCR[11:0]												
	Reset value																	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**I2C_TRISE**	Reserved																										TRISE[5:0]					
	Reset value																											0	0	0	0	1	0
0x24	**I2C_FLTR**	Reserved																											ANOFF	DNF[3:0]			
	Reset value																												0	0	0	0	0

28. Последовательный интерфейс (SPI)

28.1. Введение в SPI

Интерфейс SPI выполняет две функции: протокол SPI или звуковой протокол I²S. По умолчанию выбран SPI. С SPI на I²S переключаются программно.

Интерфейс SPI обеспечивает полу-/полно- дуплексную синхронную последовательную связь с внешними устройствами. SPI можно сконфигурировать ведущим с подачей тактов связи (SCK) внешнему ведомому устройству. Интерфейс может работать в режиме конфигурации нескольких ведущих.

I²S это тоже синхронный последовательный интерфейс. Он может использовать четыре звуковых стандарта: I²S Philips, стандарты с MSB- и LSB-выравниванием и стандарт PCM. Он может работать ведомым и ведущим в полном дуплексе (4 ножки) или полу-дуплексе (6 ножек). При работе ведущим можно тактировать ведомого.

Внимание:

Некоторые ножки SPI1 и SPI3/I2S3 могут быть направлены на ножки JTAG (SPI1_NSS на JTDI, SPI3_NSS/I2S3_WS на JTDI и SPI3_SCK/I2S3_CK на JTDO) и вы можете:

- Направить SPI/I2S на другие ножки,
- Перед конфигурацией ножек SPI отключить JTAG и при отладке использовать SWD,
- Отключить оба интерфейса JTAG/SWD.

28.2. Основные свойства SPI и I²S

28.2.1. Свойства SPI

- Синхронный полный дуплекс по трём линиям
- Синхронный симплекс по двум линиям с/без двунаправленной линией данных
- 8- или 16-бит формат фрейма передачи
- Ведущий или ведомый
- Режим нескольких ведущих
- 8 предделителей скорости ведущего (f_{PCLK}/2 max.)
- Частота режима ведомого (f_{PCLK}/2 max)
- Ускоренная связь ведущего и ведомого
- Аппаратное и программное управление NSS для ведущего и ведомого: динамическая смена операций ведущий/ведомый
- Программируемая полярность и фаза тактов
- Программируемый порядок данных: первым старший или младший бит
- Отдельные флаги передачи и приёма с прерываниями
- Флаг занятости шины SPI
- Режим SPI TI
- Аппаратный CRC:
  - CRC можно передавать как последний байт в режиме Tx
  - Автоматическая проверка CRC для последнего принятого байта
- Флаги сбоя режима ведущего, переполнения и ошибки CRC с прерываниями
- 1-байт буфер приёма и передачи: запросы Tx и Rx

28.2.2. Свойства I²S

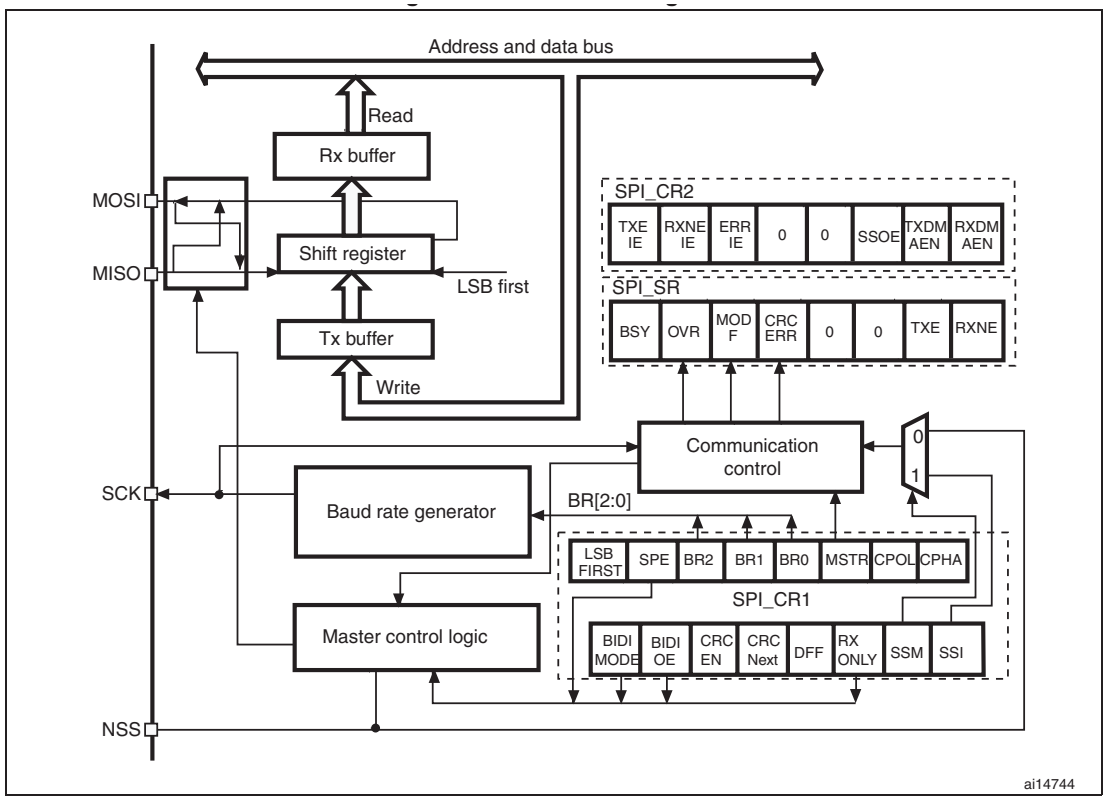
- Полный дуплекс
- Полу-дуплексная связь (только передача или приём)
- Ведущий или ведомый
- 8-бит программируемый предделитель частоты дискретизации звукового сигнала (8 - 192 kHz)
- Формат данных: 16-, 24- или 32-бит
- Фиксированный звуковым каналом фрейм пакета: 16-бит (16-бит фрейм данных) или 32-бит (16-, 24-, 32-бит фрейм данных)
- Программируемая полярность тактов (устойчивое состояние)
- Флаг исчерпания в режиме передачи ведомого и флаг переполнения в режиме приёма (ведущий и ведомый)
- 16-бит регистр для приёма и передачи с одним регистром данных на обеих сторонах канала
- Поддерживаемые протоколы I²S:
  - Стандарт I²S Philips
  - MSB-выравненный (влево)
  - LSB-выравненный (вправо)
  - Стандарт PCM (с коротким или длинным фреймом синхронизации для 16-бит фрейма канала или 16-бит фрейма данных, расширенного до 32-бит фрейма канала)
- Передача начиная с MSB

- DMA передача или приём (16-бит)
- Такты ведущего могут подаваться на внешнее звуковое устройство. Частота фиксирована на 256 × F_S (где F_S частота дискретизации звука)
- Оба I^2S (I2S2 и I2S3) имеют отдельный PLL тактирования (PLL3).
- Такты I^2S (I2S2 и I2S3) можно получить снаружи с ножки I2S_CKIN.

28.3. Функциональное описание SPI

28.3.1. Общее описание

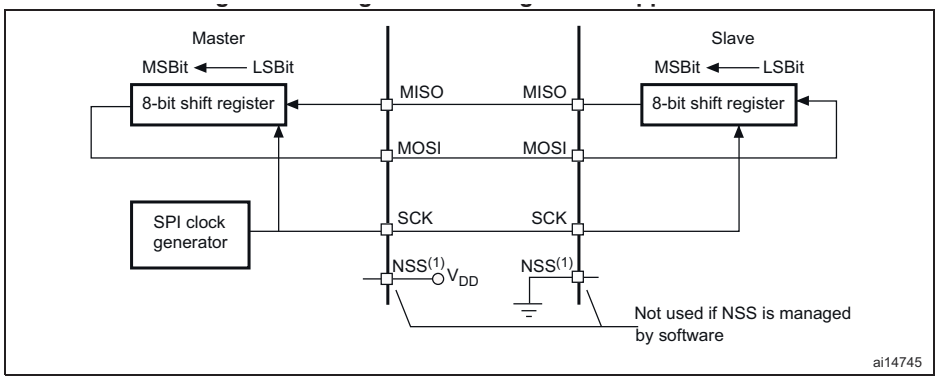
Рис. 246. Блок-схема SPI



Обычно SPI подключается к внешним устройствам четырьмя ножками:

- MISO: Приём данных ведущего и передача ведомого.
- MOSI: Приём данных ведомого и передача ведущего.
- SCK: Выход тактов ведущего и вход ведомого.
- NSS: Необязательная ножка выбора ведомого. Похоже на выбор чипа, входами NSS ведомых можно управлять через стандартные IO порты. Ножку NSS можно сделать выходом ведущего (бит `SSOE`) и выдавать активным низким. Теперь все подключённые сюда ножки NSS видят низкий уровень ведущего и становятся ведомыми, но конфигурировать нужно в аппаратном режиме. Если сделать NSS входом в режиме ведущего (`MSTR`=1 и `SSOE`=0), то при обнаружении на ней низкого уровня SPI уходит в ошибку режима ведущего: бит `MSTR` снимается и SPI переключается в ведомый.

Рис. 247. Один ведущий / Один ведомый



1. Здесь ножка NSS это вход.

Ножки MOSI, MISO и SCK соединяются попарно. Передача инициируется ведущим. Данные по MOSI и ответы по MISO синхронизируются по SCK. Это полный дуплекс.

Выбор ведомого (NSS)

Программный или аппаратный выбор ведомого определяется битом SSM регистра SPI_CR1.

- Программное управление NSS (SSM = 1)

  Выбирается битом SSI регистра SPI_CR1. Внешняя ножка NSS освобождается.

- Аппаратное управление NSS (SSM = 0)

  Есть две конфигурации, зависящие от разрешения вывода NSS (бит SSOE регистра SPI_CR2).

  – Вывод NSS разрешён (SSM = 0, SSOE = 1)

  Только для режима ведущего. NSS ставится низким вплоть до отключения SPI.

  – Вывод NSS запрещён (SSM = 0, SSOE = 0)

  Разрешает работу нескольких ведущих устройств. У ведомых устройств ножка NSS это обычный ввод NSS: ведомый выбирается при низком входе NSS.

Фаза и полярность тактов

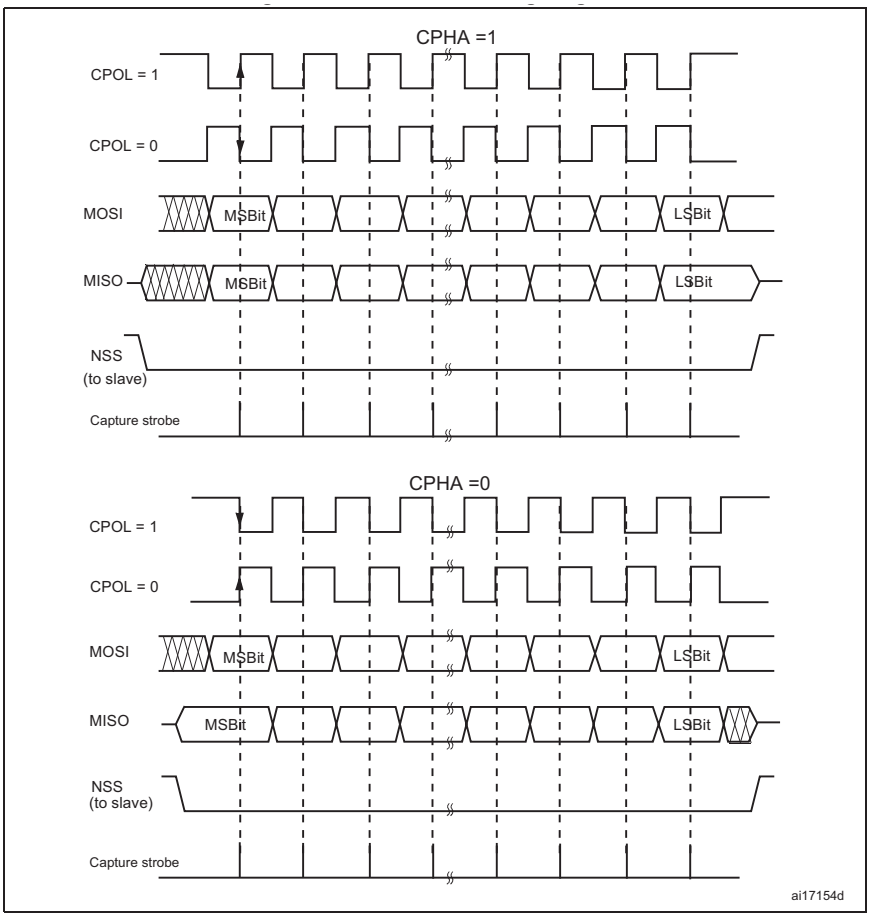
Биты CPOL и CPHA регистра SPI_CR1 определяют четыре возможных времянки. Бит CPOL (полярность) определяет значение устойчивого состояния тактового сигнала (данные не передаются). Действует и в ведущем и в ведомом режимах. Если CPOL сброшен, то SCK готовится низким, иначе - высоким.

Если бит CPHA (фаза) стоит, то передача бита стробируется вторым фронтом SCK (задним при снятом бите CPOL и передним при стоящем бите CPOL). Пришедший бит запоминается по второму тику. Если бит CPHA сброшен, то передача бита стробируется первым фронтом SCK (задним при стоящем бите CPOL и передним при снятом бите CPOL). Пришедший бит запоминается по первому же тику.

NB: Перед изменением битов CPOL/CPHA бит SPE надо снимать, отключая SPI. Ведомый и ведущий должны иметь одинаковый режим времянки.

Состояние простоя SCK должно соответствовать полярности из регистра SPI_CR1 (подпорка при CPOL=1 и подтяжка вниз при CPOL=0).

Рис. 248. Диаграмма тактирования данных



1. Здесь бит LSBFIRST регистра SPI_CR1 сброшен.

Формат фрейма данных

Данные выдвигаются старшим или младшим битом вперёд в зависимости от бита LSBFIRST в регистре SPI_CR1. Длина фрейма данных (8- или 16-бит) выбирается битом DFF регистра SPI_CR1.

28.3.2. Ведомый SPI

Такты поступают на ножку SCK, биты BR[2:0] регистра SPI_CR1 на частоту передачи не влияют.

NB: Включать ведомый SPI надо до подачи тактов. Регистр данных должен быть готов до первого фронта такта или до его конца. Ставить полярность тактов надо до включения ведущего и ведомого..

Конфигурация ведомого

1. Ставим бит DFF (8- или 16-бит данные)
2. Ставим биты CPOL и CPHA (такие же как для ведущего). В режиме TI этот шаг не нужен.
3. Ставим формат фрейма (первым старший или младший бит) битом LSBFIRST. В режиме TI этот шаг не нужен.
4. В Аппаратном режиме ножка NSS должна быть низкой во время передачи всего байта. В Программном режиме ставим бит SSM и снимаем бит SSI регистра SPI_CR1. В режиме TI этот шаг не нужен.
5. Ставим бит FRF в регистре SPI_CR2 для протокола TI.
6. Чистим бит MSTR и ставим бит SPE регистра SPI_CR1, переведя ножки на альтернативную функцию.

В этом режиме ножка MOSI это ввод, а MISO это вывод.

Последовательность передачи

Байт данных пишется в буфер Tx параллельно.

Передача начинается по получению ведомым тактового сигнала при старшем значащем бите данных на ножке MOSI. Оставшиеся биты (7 или 15 бит) загружены в регистр сдвига. После передачи данных из буфера Tx ставится флаг TXE в регистре SPI_SR и при стоящем бите TXEIE в регистре SPI_CR2 выдаётся прерывание.

Последовательность приёма

По завершении передачи данных:

- Данные из регистра сдвига передаются в буфер Rx и ставится флаг RXNE в регистре SPI_SR
- При стоящем бите RXNEIE в регистре SPI_CR2 выдаётся прерывание.

После чтения регистра SPI_DR снимается бит RXNE и SPI возвращается в приём.

Протокол SPI TI ведомого

Бит FRF регистра SPI_CR2 делает ведомого SPI совместимым с этим протоколом.

Полярность и фаза тактов становится совместимой с протоколом TI независимо от значений в регистре SPI_CR1. Специфичное для протокола TI управление NSS через регистры SPI_CR1 и SPI_CR2 (вроде SSM, SSI, SSOE) весьма прозрачно.

У Ведомого момент перехода ножки MISO в состояние HI-Z определяется предделителем скорости передачи. Но обычно скорость задаётся ведущим и время перехода сигнала MISO в состояние HI-Z ($t_{release}$) зависит от внутренней ресинхронизации и значения BR[2:0] в регистре SPI_CR1. Получают по формуле:

$$\frac{t_{baud\_rate}}{2} + 4 \times t_{pclk} < t_{release} < \frac{t_{baud\_rate}}{2} + 6 \times t_{pclk}$$

NB: Сие недоступно для протокола Motorola SPI (FRF = 0).

Для обнаружения ошибок фрейма TI при передаче Ведомого с помощью прерывания ошибки (ERRIE = 1), SPI надо поставить в 2-проводной однонаправленный режим установкой битов BIDIMODE и BIDIOE в регистре SPI_CR1. При BIDIMODE = 0, всегда прерывание OVR из-за нечитанного регистра данных ставится всегда, а при BIDIMODE = 1 данные не принимаются и OVR не ставится.

Рис. 249. Режим TI - Ведомый, одинарная передача

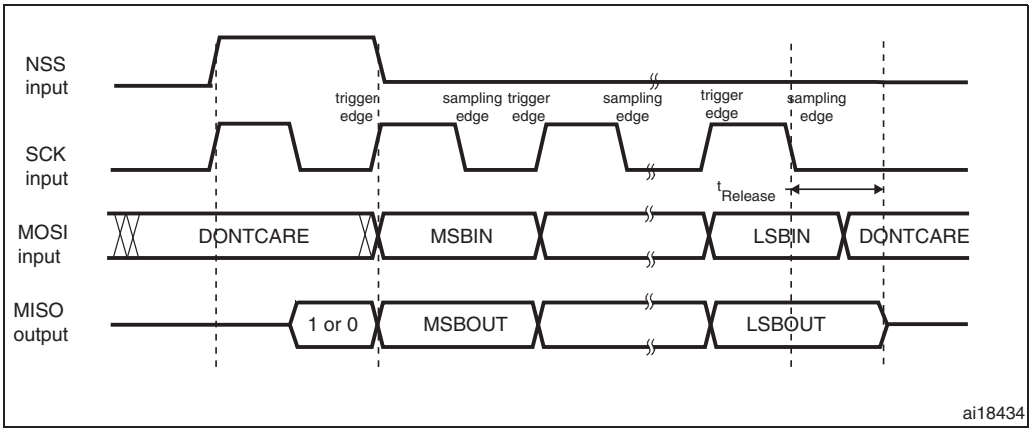
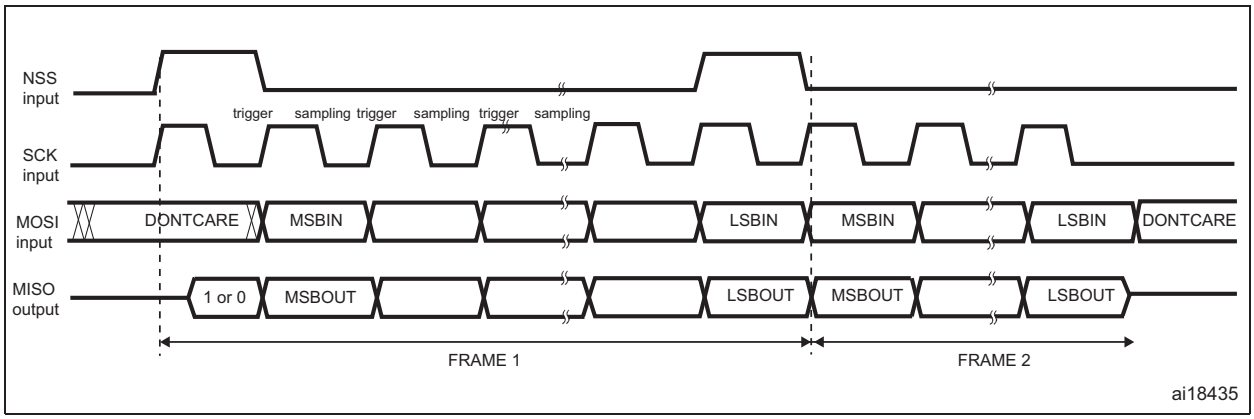


Рис. 250. Режим TI mode -Ведомый, непрерывная передача



28.3.3. Ведущий SPI

Тактовый сигнал выводится на ножку SCK.

Конфигурация

1. Битами `BR[2:0]` ставим частоту тактов обмена. В режиме TI этот шаг не нужен.
2. Ставим биты `CPOL` и `CPHA` вариантов тактирования. В режиме TI этот шаг не нужен.
3. Определяем длину фрейма данных битом `DFF`.
4. Определяем формат фрейма данных битом `LSBFIRST` регистра `SPI_CR1`. В режиме TI этот шаг не нужен.
5. Если ножка NSS нужна для ввода, то в аппаратном режиме подключаем NSS к высокому уровню на всё время передачи байта, в программном режиме ставим биты `SSM` и `SSI` в регистре `SPI_CR1`. Если ножка NSS нужна для вывода, то ставить надо только бит `SSOE`. В режиме TI этот шаг не нужен.
6. Битом `FRF` регистра `SPI_CR2` выбираем протокол TI.
7. Биты `MSTR` и `SPE` должны стоять (они остаются стоять только при подключении NSS к высокому уровню).

Сейчас MOSI это вывод, а MISO это ввод.

Последовательность передачи

Начинается записью байта в буфер Tx.

Во время передачи первого бита перегружается в регистр сдвига и выдаётся на ножку MOSI побитно нужным концом вперёд. После передачи данных из буфера Tx ставится флаг `TXE` в регистре `SPI_SR` и при стоящем бите `TXEIE` в регистре `SPI_CR2` выдаётся прерывание.

Последовательность приёма

По завершении передачи данных:

- Данные из регистра сдвига передаются в буфер Rx и ставится флаг `RXNE` в регистре `SPI_SR`
- При стоящем бите `RXNEIE` в регистре `SPI_CR2` выдаётся прерывание.

После чтения регистра `SPI_DR` снимается бит `RXNE` и SPI возвращается в приём.

Непрерывный поток передачи можно организовать записью очередного байта в буфер Tx поле начала передачи из регистра сдвига. При этом флаг TXE должен стоять.

NB: Для отключения ведомых SPI между передачами ножку NSS нужно конфигурировать как GPIO или использовать другой GPIO и переключать программно.

Протокол SPI TI ведущего

Бит FRF регистра SPI_CR2 делает ведущего SPI совместимым с этим протоколом.

Полярность и фаза тактов становится совместимой с протоколом TI независимо от значений в регистре SPI_CR1. Специфичное для протокола TI управление NSS через регистры SPI_CR1 и SPI_CR2 (вроде SSM, SSI, SSOE) весьма прозрачно.

Рис. 251. Режим TI - ведущий, одинарная передача

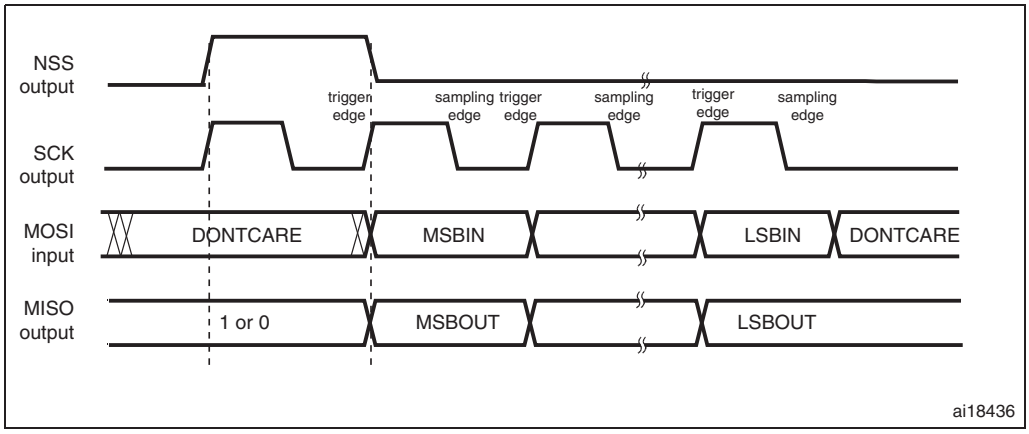
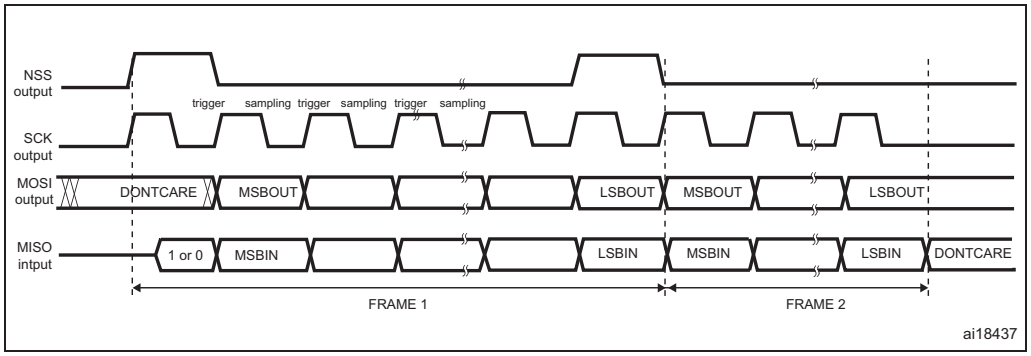


Рис. 252. Режим TI - ведущий, непрерывная передача



28.3.4. SPI в полудуплексе

Есть 2 конфигурации SPI в полудуплексе:

- 1 тактовый и 1 двунаправленный провод данных
- 1 тактовый и 1 провод данных (приём или передача)

1 тактовый и 1 двунаправленный провод данных (BIDIMODE=1)

Включается установкой бита BIDIMODE в регистре SPI_CR1. В этом режиме такты идут по SCK, а данные по MOSI ведущего или MISO ведомого. При стоящем бите BIDIOE регистра SPI_CR1 это линия вывода, иначе это ввод.

1 тактовый и 1 провод данных (BIDIMODE=0)

В этом режиме SPI работает только на ввод или на вывод.

- "Только передача" подобна полному дуплексу (BIDIMODE=0, RXONLY=0): данные передаются по передающей ножке (MOSI ведущего или MISO ведомого), а приёмную (MISO ведущего или MOSI ведомого) можно использовать как GPIO. В этом случае просто игнорируем буфер Rx.
- В "только приёме" отключаем вывод SPI установкой бита RXONLY регистра SPI_CR1. Теперь передающая ножка (MOSI ведущего или MISO ведомого) свободна для GPIO.

Запускаем "только ввод" конфигурируя и затем включая SPI:

- У ведущего ввод начинается немедленно и останавливается при очистке бита SPE и остановке текущего приёма. Читать флаг BSY нет нужды. Он всегда стоит при работающей операции SPI.
- У ведомого SPI продолжает принимать при низком NSS (или очистке бита SSI при программном управлении) и поступающих тактах SCK.

28.3.5. Приём и передача данных

Буферы Rx и Tx

При приёме данные из регистра сдвига пишутся в буфер Rx, при передаче записываемые в буфер Tx данные поступают в регистр сдвига. Чтение регистра SPI_DR выдаёт данное из Rx, а запись в SPI_DR пишет в буфер Tx.

Стартовая последовательность ведущего

- В полном дуплексе (BIDIMODE=0 и RXONLY=0)
  - Последовательность начинается записью данных в регистр SPI_DR (буфер Tx).
  - Данные переносятся из Tx в 8-бит регистр сдвига во время передачи первого бита и затем последовательно выдвигаются на ножку MOSI.
  - В то же время приёмные данные на ножке MISO последовательно вдвигаются в 8-бит регистр сдвига, а затем параллельно переносятся в регистр SPI_DR (буфер Rx).
- При однонаправленном приёме (BIDIMODE=0 и RXONLY=1)
  - Последовательность начинается установкой SPE=1
  - Активируется только приёмник и данные на ножке MISO последовательно вдвигаются в 8-бит регистр сдвига, а затем параллельно переносятся в регистр SPI_DR (буфер Rx).
- Двунаправленный режим, передача (BIDIMODE=1 и BIDIOE=1)
  - Последовательность начинается записью данных в регистр SPI_DR (буфер Tx).
  - Данные переносятся из Tx в 8-бит регистр сдвига во время передачи первого бита и затем последовательно выдвигаются на ножку MOSI.
  - Данные не принимаются.
- Двунаправленный режим, приём (BIDIMODE=1 и BIDIOE=0)
  - Последовательность начинается установкой SPE=1 и BIDIOE=0.
  - Приёмные данные на ножке MISO последовательно вдвигаются в 8-бит регистр сдвига, а затем параллельно переносятся в регистр SPI_DR (буфер Rx).
  - Передатчик не активируется и данные на ножку MOSI не поступают.

Стартовая последовательность ведомого

- Полный дуплекс (BIDIMODE=0 и RXONLY=0)
  - Последовательность начинается когда ведомый принимает тактовый сигнал и первый бит данных на своей ножке MOSI. Остальные 7 бит идут в регистр сдвига.
  - Одновременно, при передаче первого бита данные из буфера Tx пишутся в регистр сдвига и затем выдвигаются на ножку MISO. Писать передаваемые данные нужно до запуска передачи ведущим устройством SPI.
- При однонаправленном приёме (BIDIMODE=0 и RXONLY=1)
  - Последовательность начинается когда ведомый принимает тактовый сигнал и первый бит данных на своей ножке MOSI. Остальные 7 бит идут в регистр сдвига.
  - Передатчик не активируется и данные на ножку MOSI не поступают.
- Двунаправленный режим, передача (BIDIMODE=1 и BIDIOE=1)
  - Последовательность начинается когда ведомый принимает тактовый сигнал и первый бит из буфера Tx передаётся на ножку MISO.
  - Данные переносятся из Tx в 8-бит регистр сдвига во время передачи первого бита и затем последовательно выдвигаются на ножку MISO. Писать передаваемые данные нужно до запуска передачи ведущим устройством SPI.
  - Данные не принимаются.
- Двунаправленный режим, приём (BIDIMODE=1 и BIDIOE=0)
  - Последовательность начинается когда ведомый принимает тактовый сигнал и первый бит данных на своей ножке MISO.
  - Приёмные данные на ножке MISO последовательно вдвигаются в 8-бит регистр сдвига, а затем параллельно переносятся в регистр SPI_DR (буфер Rx).
  - Передатчик не активируется и данные на ножку MISO не поступают.

Обработка данных передачи и приёма

Флаг TXE (буфер Tx пуст) ставится после пересылки данных из буфера Tx в регистр сдвига. При стоящем бите TXEIE в регистре SPI_CR2 выдаётся прерывание. Чистится бит TXE записью в регистр SPI_DR.

NB: Запись в Tx при снятом флаге TXE заменяет хранящиеся там данные.

Флаг RXNE (буфер Rx не пуст) ставится по последнему фронту такта считывания при передаче данных из регистра сдвига в буфер Rx. Данные можно читать из SPI_DR. При стоящем бите RXNEIE в регистре SPI_CR2 выдаётся прерывание. Чистится бит RXNE чтением регистра SPI_DR.

В некоторых конфигурациях для ожидания конца передачи последнего данного можно использовать флаг BSY.

Полный дуплекс BIDIMODE=0 и RXONLY=0.

Программная процедура:

1. Включаем SPI установкой бита SPE.
2. Пишем первое передаваемое данное в регистр SPI_DR (флаг TXE снимается).
3. Ждём TXE=1 и пишем второе передаваемое данное. Теперь ждём RXNE=1 и читаем регистр SPI_DR с первым принимаемым данным (флаг RXNE снимается). Повторяем передачу/приём до получения n–1 данных.
4. Ждём RXNE=1 и читаем последнее данное.
5. Ждем TXE=1 и затем BSY=0 перед отключением SPI.

Всё это можно сделать с помощью программ прерывания по передним фронтам флагов RXNE и TXE.

Рис. 253. Флаги TXE/RXNE/BSY ведущего в полном дуплексе (BIDIMODE=0 и RXONLY=0) при непрерывной передаче

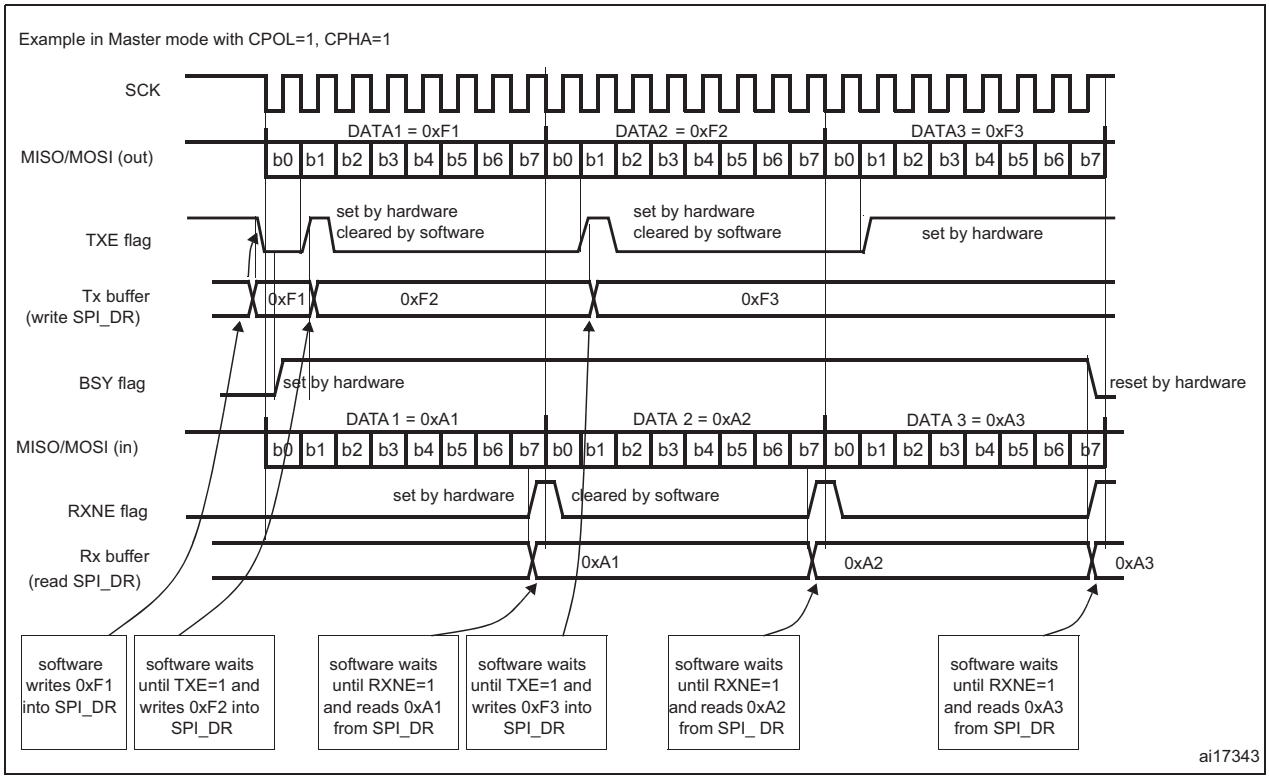
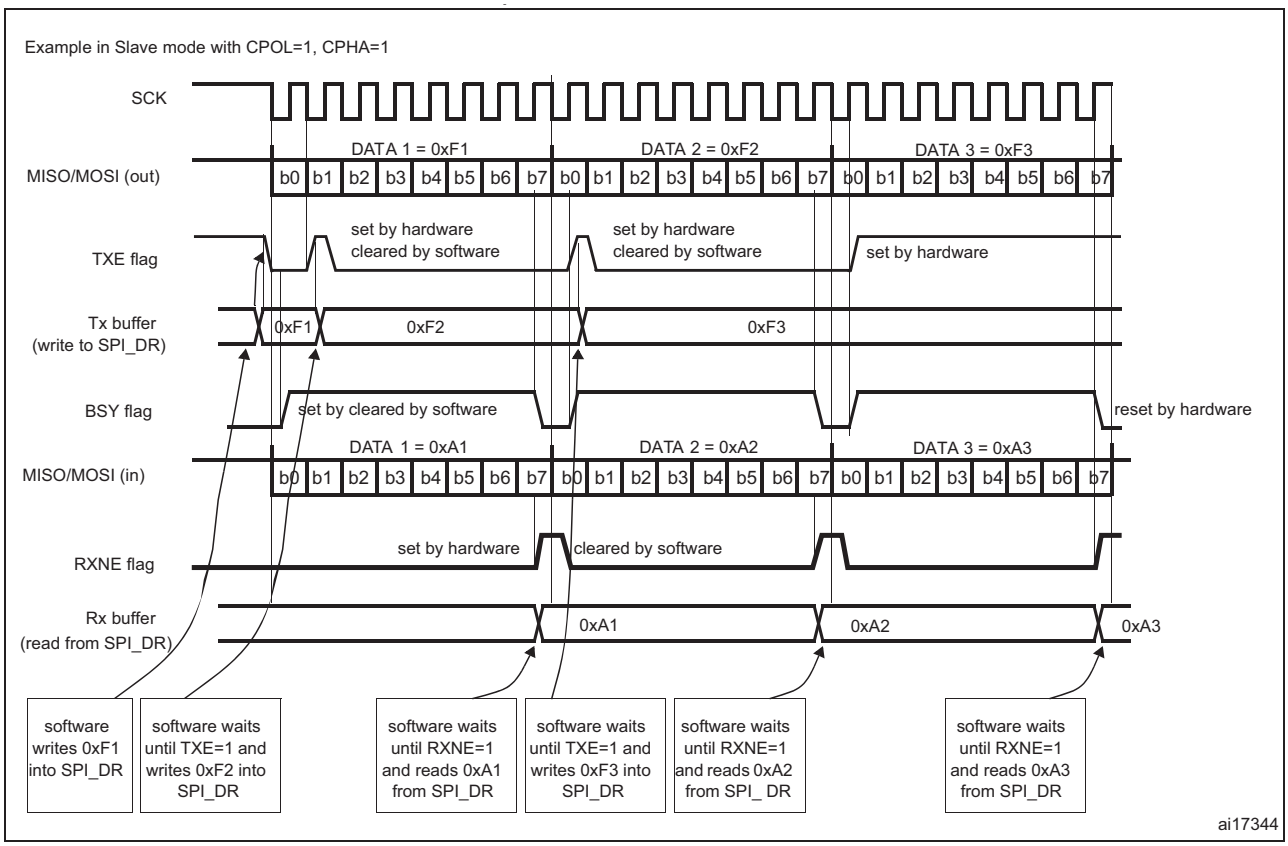


Рис. 254. Флаги TXE/RXNE/BSY ведомого в полном дуплексе (BIDIMODE=0 и RXONLY=0) при постоянной передаче



Процедура Только передачи (BIDIMODE=0 RXONLY=0)

В этом режиме процедуру можно сократить и использовать бит BSY для ожидания конца передачи.

1. Включаем SPI установкой бита SPE.
2. Пишем первое посылаемое данное в регистр SPI_DR (битами TXE чистится).
3. Ждём TXE=1 и пишем следующее пересылаемое данное. Повторяем этот шаг для всех данных.
4. После записи последнего данного ждём TXE=1, и затем BSY=0 для определения конца передачи последнего данного.

Всё это можно сделать с помощью программ прерывания по переднему фронту флага TXE.

Во время прерывных передач между записью в SPI_DR и установкой бита BSY появляется задержка в 2 такта APB. Следовательно после записи последнего данного дожидаться установки бита TXE и затем снятия бита BSY нужно обязательно.

После передачи двух данных в режиме только передачи встаёт флаг OVR регистра SPI_SR, так как читать данные никто не намерен.

Рис. 255. Флаги TXE/BSY ведущего/только передача (BIDIMODE=0 и RXONLY=0) при постоянной передаче

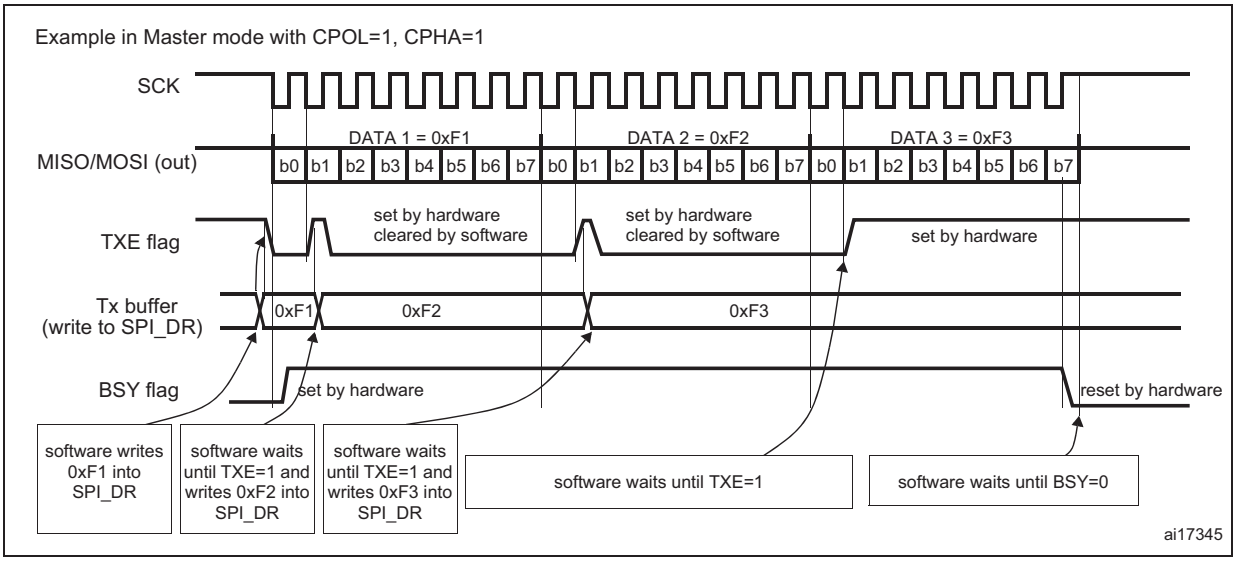
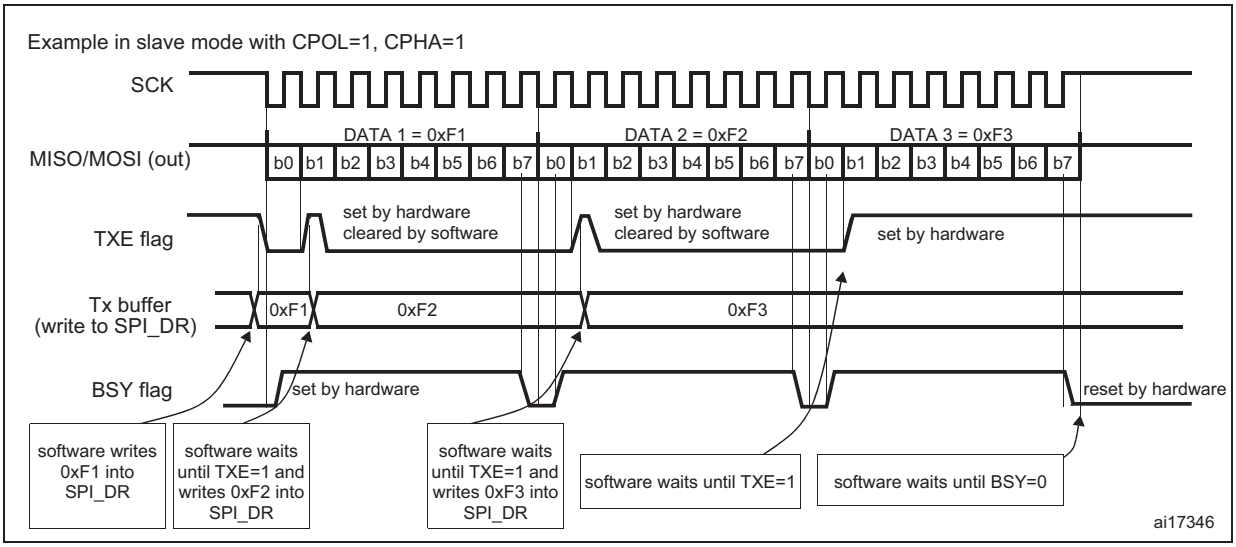


Рис. 256. Флаги TXE/BSY ведомого/только передача (BIDIMODE=0 and RXONLY=0) при постоянной передаче



Процедура двунаправленной передачи (BIDIMODE=1 и BIDIOE=1)

Она подобна процедуре режима Только передачи, но перед включением SPI надо поставить биты `BIDIMODE` и `BIDIOE` регистра `SPI_CR2`.

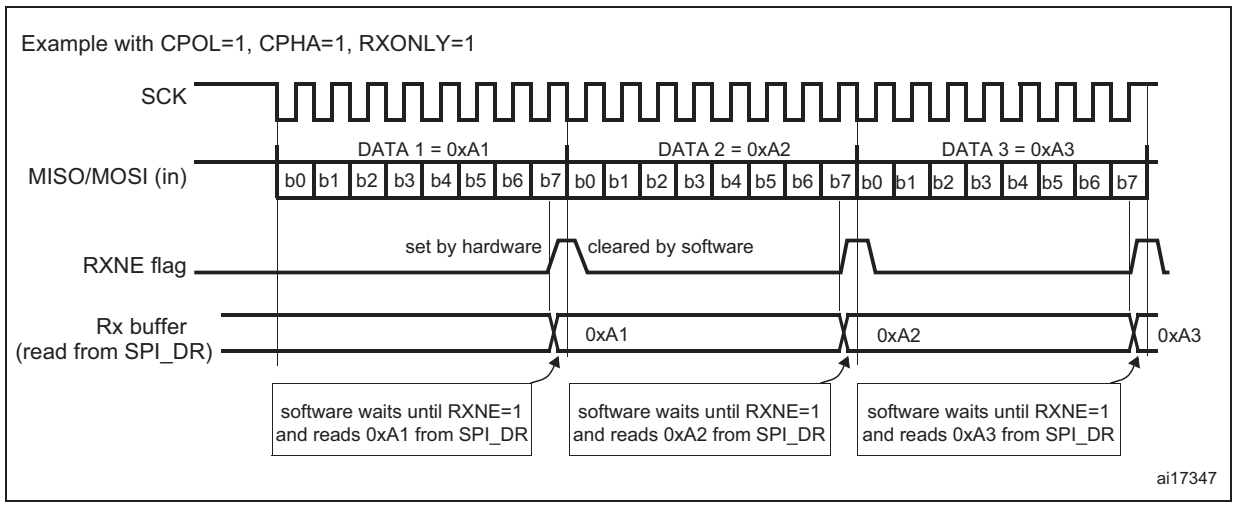
Процедура однонаправленного приёма (BIDIMODE=0 и RXONLY=1)

1. Ставим бит `RXONLY` в регистре `SPI_CR1`.
2. Включаем SPI установкой бита `SPE`:
   a) У ведущего немедленно активируется выдача тактов на SCK и приём последовательных данных вплоть до выключения SPI (`SPE`=0).
   b) У ведомого данные принимаются при низком NSS и появлении тактов SCK.
3. Ждём `RXNE`=1 и читаем регистр `SPI_DR` (бит `RXNE` снимается). Повторяем этот пункт.

Всё это можно сделать с помощью программ прерывания по переднему фронту флага `RXNE`.

После завершения всех передач SPI надо выключать.

Рис. 257. Флаг RXNE только приём (BIDIRMODE=0 и RXONLY=1) при постоянной передаче



Процедура двунаправленного приёма (BIDIMODE=1 и BIDIOE=0)

Она подобна процедуре режима Только приёма, но перед включением SPI надо поставить бит BIDIMODE и снять BIDIOE регистра SPI_CR2.

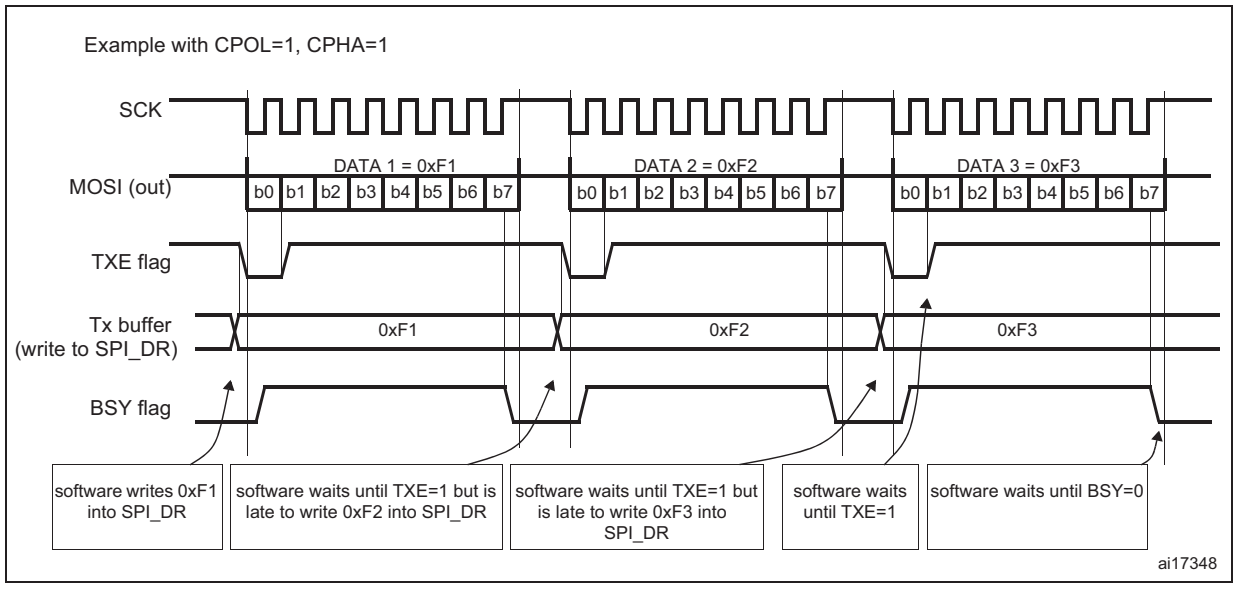
Непрерывные и прерывные передачи

Если в режиме ведущего программа успевает обнаружить каждый передний TXE (или прерывание TXE) и записать регистр SPI_DR до завершения текущей передачи, то это непрерывная передача: подача тактов SPI не прерывается и бит BSY между передачами данных не снимается. У прерывных передач бит BSY успевает сниматься.

В только приёме ведущего (RXONLY=1), связь непрерывная и BSY читается как 1.

У ведомого непрерывность определяется ведущим, но бит BSY всегда снимается хоть на один такт SPI.

Рис. 258. Флаги TXE/BSY передача (BIDIRMODE=0 and RXONLY=0) при прерывной передаче



28.3.6. Вычисление CRC

Для приёмных и передаваемых данных используются два отдельных калькулятора CRC. Он вычисляется по фронту считывания, в зависимости от битов CPHA и CPOL регистра SPI_CR1.

NB: Для разных данных используются два вида CRC: 8-бит (CR8) 16-бит (CRC16).

Вычисление CRC разрешается установкой бита CRCEN регистра SPI_CR1. Этим сбрасываются регистры CRC (SPI_RXCRCR и SPI_TXCRCR). При программном управлении в полном дуплексе и только передаче сразу после записи последнего данного в регистр SPI_DR надо записать бит CRCNEXT. Тогда после этой передачи будет передан регистр SPI_TXCRCR.

В только приёме и программном управлении бит CRCNEXT надо писать после второго принятого данного и CRC будет проверен.

По концу передачи CRC результат вычисления определяет флаг CRCERR в регистре SPI_SR.

При наличии данного в буфере TX значение CRC передаётся после передачи последнего байта. При передаче CRC калькулятор CRC выключается и значение регистра не изменяется.

Процедура такая:

1. Пишем значения CPOL, CPHA, LSBFIRST, BR, SSM, SSI и MSTR.
2. Пишем полином в регистр SPI_CRCPR.
3. Включаем вычисление CRC установкой бита CRCEN в регистре SPI_CR1. При этом чистятся регистры SPI_RXCRCR и SPI_TXCRCR.
4. Включаем SPI установкой бита SPE в регистре SPI_CR1.
5. Запускаем связь и поддерживаем её до предпоследнего байта.
   - При программном управлении в полном дуплексе или только передаче сразу после записи последнего байта в регистр SPI_DR, надо записать бит CRCNEXT. Тогда после этой передачи будет передан регистр SPI_TXCRCR.
   - В только приёме бит CRCNEXT надо ставить сразу после приёма предпоследнего данного, чтобы подготовить переход SPI в фазу CRC по концу приёма последнего данного. Вычисление CRC во время его передачи стопорится.
6. После передачи последнего байта или полуслова SPI уходит в фазу передачи и проверки CRC. В полном дуплексе или только приёме принятый CRC сравнивается с регистром SPI_RXCRCR. При несовпадении ставится флаг CRCERR в регистре SPI_SR и при стоящем бите ERRIE в регистре SPI_CR2 выдаётся прерывание.

В режиме ведомого включайте вычисление CRC только при стабильном такте (в устойчивом состоянии). Иначе можно вычислить не так. В действительности CRC чувствителен к входным тактам ведомого сразу после установки CRCEN, независимо от бита SPE.

Будьте внимательны с передачей CRC на высоких частотах. Число тактов CPU в фазе передачи CRC должно быть минимально возможным, так что в это время вызов функций запрещён, чтобы не терять последнее данное. В действительности бит CRCNEXT надо писать перед концом передачи/приёма последнего данного.

На высоких частотах передачи полезно использовать DMA.

У ведомых устройств в аппаратном режиме NSS, ножку NSS нужно держать низкой между фазами данных и CRC.

При разрешении функции CRC ведомых устройств, он вычисляется даже при высокой ножке NSS.

Во время переключения выбора NSS значения CRC нужно сбрасывать на ведущей и ведомой стороне.

Процедура очистки CRC такова:

1. Выключаем SPI (SPE = 0)
2. Снимаем бит CRCEN
3. Ставим бит CRCEN
4. Включаем SPI (SPE = 1)

28.3.7. Флаги состояния

Есть четыре флага состояния:

Флаг пустого буфера Tx (TXE)

Ставится аппаратно. Снимается записью в регистр SPI_DR.

Флаг занятого буфера Rx (RXNE)

Ставится аппаратно. Снимается чтением регистра SPI_DR.

Флаг BUSY

Флаг BSY ставится и снимается аппаратно при занятом и свободном уровне связи SPI соответственно. Исключение составляет двунаправленный приём ведущего (MSTR=1 и BDM=1 и BDOE=0), когда флаг BSY остаётся низким во время приёма. Полезен для обнаружения конца

передачи перед остановкой SPI (или выключения тактов устройства). Так не нарушится последняя передача. Для этого надо соблюсти описанную ниже процедуру.

Также флаг BSY полезен для исключения коллизий записи в системе с несколькими ведомыми.

Ставится в начале передачи кроме режима с (MSTR=1, BDM=1 и BDOE=0).

Снимается:

- в конце передачи (кроме непрерывных передач ведущего)
- при выключении SPI
- при сбое режима ведущего (MODF=1)

У прерывных передач флаг BSY снимается между всеми передачами.

При непрерывных передачах:

- у ведущего флаг BSY остаётся высоким во время всех передач
- у ведомого флаг BSY снимается на один такт между передачами

NB: Не используйте флаг BSY для обработки конца передачи, флаги TXE и RXNE гораздо лучше.

28.3.8. Отключение SPI

Для этого снимается бит SPE.

В некоторых конфигурациях выключение SPI с незавершённой передачей может нарушить её и исказить бит BSY. Так что нужно соблюсти процедуры:

В полном дуплексе ведущего или ведомого (BIDIMODE=0, RXONLY=0)

1. Ждём RXNE=1 для последнего данного
2. Ждём TXE=1
3. Ждём BSY=0
4. Выключаем SPI (SPE=0)

Однонаправленная только передача ведущего или ведомого (BIDIMODE=0, RXONLY=0) или двунаправленная передача (BIDIMODE=1, BIDIOE=1)

После записи последнего данного в регистр SPI_DR:

1. Ждём TXE=1
2. Ждём BSY=0
3. Выключаем SPI (SPE=0)

Однонаправленный только приём ведущего (MSTR=1, BIDIMODE=0, RXONLY=1) или двунаправленный приём (MSTR=1, BIDIMODE=1, BIDIOE=0)

Тут надо убедиться, что SPI не начал новую передачу:

1. Ждём предпоследнего появления RXNE=1 (n–1)
2. Программным циклом ждём 1 такт SPI перед его выключением (SPE=0)
3. Теперь ждём RXNE=1 перед остановом

NB: У ведущего при двунаправленном приёме (MSTR=1 and BDM=1 and BDOE=0) флаг BSY остаётся высоким во время всех передач.

Ведомый только приём (MSTR=0, BIDIMODE=0, RXONLY=1) или двунаправленный приём (MSTR=0, BIDIMODE=1, BIDOE=0)

1. Выключать SPI (SPE=1) можно когда угодно: сначала закончится текущая передача
2. Перед остановом дождитесь BSY = 0.

28.3.9. SPI с DMA

Доступ с DMA включается отдельными битами регистра SPI_CR2 для запросов Tx и Rx буферов:

- При передаче запрос DMA выдаётся по установке TXE. Запись DMA в SPI_DR чистит его.
- При приёме запрос DMA выдаётся по установке RXNE. Чтение DMA из SPI_DR чистит его.

При работе только в одном направлении можно включать только один канал DMA (Tx или Rx).

При работе SPI только на передачу флаг OVR стоит, поскольку принимаемые данные не читаются.

При приёме, после передачи DMA всех данных (стоит флаг TCIF регистра DMA_ISR), конец работы SPI отслеживают по флагу BSY. Нужно дождаться сначала TXE=1, затем BSY=0.

NB: Во время прерывных операций между записью регистра SPI_DR и флагом BSY появляется задержка в 2 такта APB. Так что после записи последнего данного нужно дождаться сначала TXE=1, затем BSY=0.

Рис. 259. Передача с DMA

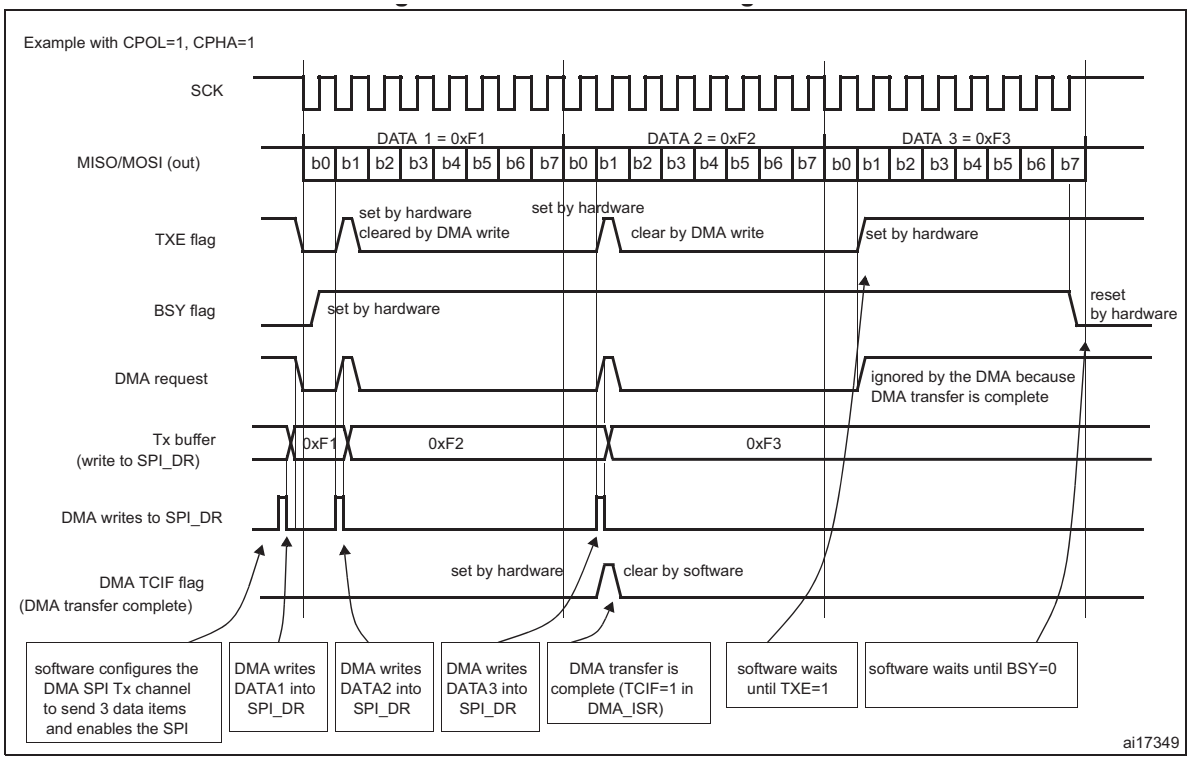
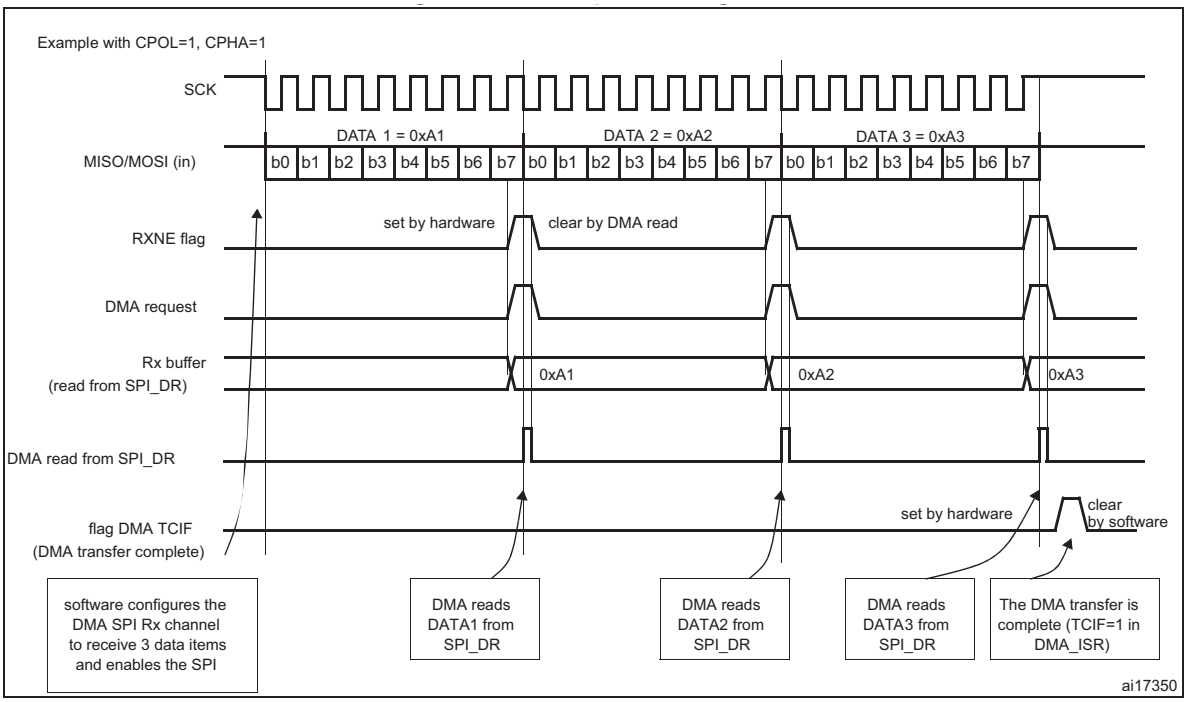


Рис. 260. Приём с DMA



DMA с CRC

При DMA передачах с включённым CRC бит CRCNEXT ставить не надо, всё происходит автоматически, надо лишь прочитать полученный CRC из регистра SPI_DR чтобы снять флаг RXNE.

При ошибке передачи с CRC ставится флаг CRCERR в регистре SPI_SR.

28.3.10.Флаги ошибок

Сбой режима ведущего (MODF)

Бит MODF автоматически ставится если ножка NSS ведущего (при аппаратном управлении) силком включена в низкий уровень, или бит SSI стоит низким (при программном управлении). Сбой работает так:

- Бит MODF ставится и при стоящем бите ERRIE выдаётся прерывание.

- Чистится бит SPE, чем блокируются выходы и отключается интерфейс SPI.
- Бит MSTR снимается, теперь он ведомый.

Программно бит MODF снимается так:

1. Читаем или пишем регистр SPI_SR при стоящем бите MODF.
2. Пишем регистр SPI_CR1.

Во избежание конфликта ведомых в системе с несколькими MCU во время очистки бита MODF ножка NSS должна удерживаться высокой. После этой очистки биты SPE и MSTR можно ставить в исходное состояние, при стоящем MODF аппаратура не позволяет ставить биты SPE и MSTR.

У ведомых бит MODF не ставится. Но в системе с несколькими ведущими устройство может попасть в ведомые со стоящим MODF. Так обнаруживается конфликт ведущих.

Переполнение

Возникает при посылке ведущим данных, когда ведомый ещё не очистил бит RXNE предыдущей передачи. При переполнении:

- ставится бит OVR и при стоящем бите ERRIE выдаётся прерывание.

В этом случае новый байт не заменяет старого содержимого в регистре SPI_DR и все последующие байты теряются.

Снимается бит OVR чтением из регистра SPI_DR с последующим чтением из регистра SPI_SR.

Ошибка CRC

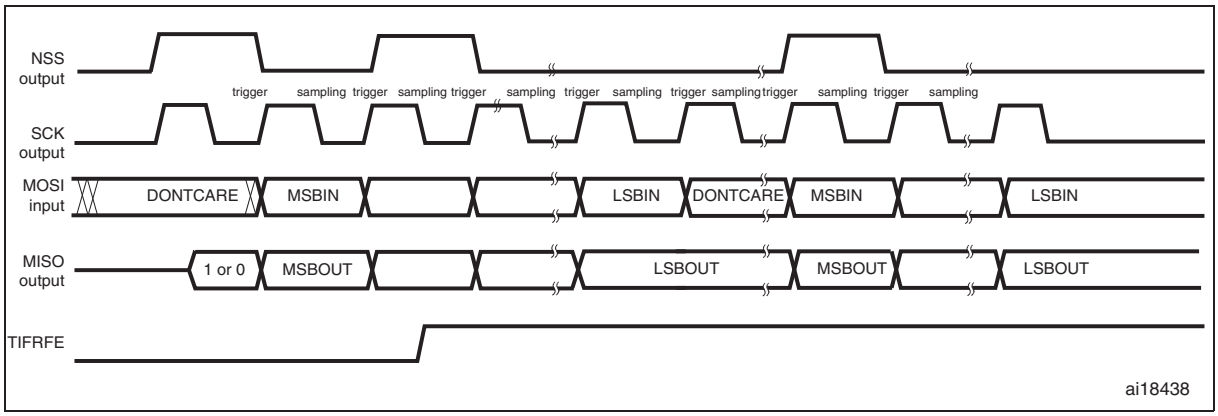
Флаг CRCERR в регистре SPI_SR ставится если принятое значение CRC не совпадает с содержимым регистра SPI_RXCRCR при стоящем бите CRCEN в регистре SPI_CR1.

Ошибка формата фрейма в режиме TI

При появлении ошибки ставится флаг FRE в регистре SPI_SR. SPI не выключается, импульс NSS игнорируется и SPI ждёт следующего импульса NSS для начала новой передачи. В данных могут потеряться два байта.

Флаг FRE снимается чтением регистра SPI_SR. При стоящем бите ERRIE выдаётся прерывание ошибки NSS. В этом случае SPI надо выключить, так как данные разрушаются и нужна новая инициализация связи ведущим после повторного включения ведомого.

Рис. 261. Ошибка формата фрейма TI



28.3.11.Прерывания SPI

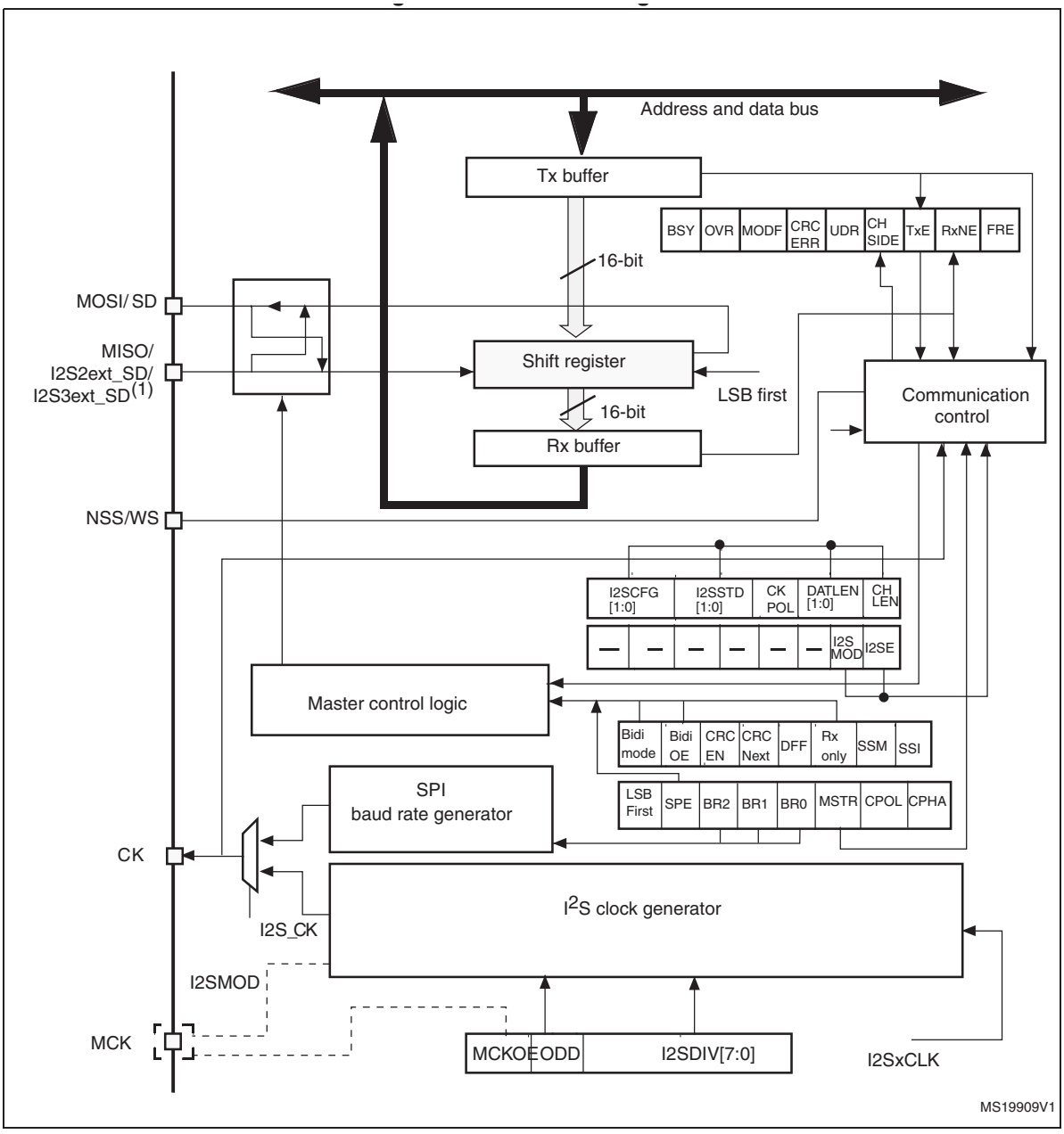
Таблица 126. Запросы прерываний SPI

Прерывание	Флаг события	Бит разрешения
Передающий буфер пуст	TXE	TXEIE
Приёмный буфер не пуст	RXNE	RXNEIE
Сбой Ведущего	MODF	ERRIE
Переполнение	OVR	
Ошибка CRC	CRCERR	

28.4. Функциональное описание I²S

28.4.1. Общее описание I²S

Рис. 262. Блок-схема I²S



1. I2S2ext_SD и I2S3ext_SD это расширение ножек SD для режима полного дуплекса I²S.

Режим звукового интерфейса I²S блока SPI включается установкой бита `I2SMOD` в регистре `SPI_I2SCFGR`. Используются почти те же ножки, флаги и прерывания, что и в SPI.

У I²S три общие ножки с SPI:

- SD: Последовательные данные (или MOSI) для приёма или передачи двух мультиплексированных по времени каналов данных (только полу-дуплекс).
- WS: Выбор слова (или NSS) это вывод данных для ведущего или ввод данных для ведомого.
- CK: Такты линии (или SCK) это выход тактов для ведущего или вход тактов для ведомого.
- I2S2ext_SD и I2S3ext_SD (на ножке MISO): для режима полного дуплекса I²S.

Дополнительная ножка используется когда внешнему устройству нужны основные такты:

- MCK: Ведущие такты (отдельная), используется при ведущем I²S (и при стоящем бите `MCKOE` в регистре `SPI_I2SPR`) для вывода дополнительных тактов частотой 256 × F_S, где F_S это частота дискретизации звука.

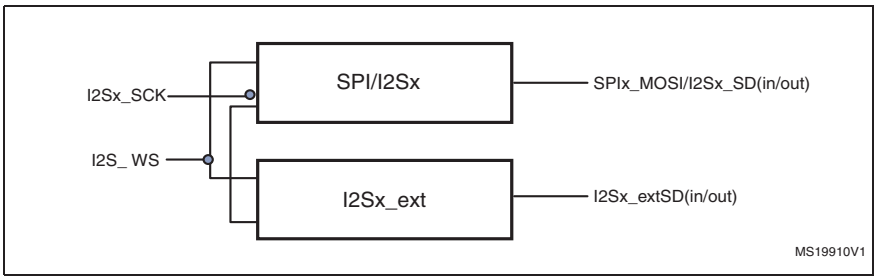
В режиме ведущего I²S сам генерирует такты связи и ведущие такты от одного генератора. Также есть два дополнительных регистра. Один конфигурации тактов (SPI_I2SPR) и другой конфигурации I²S (SPI_I2SCFGR) (стандарт звука, ведущий/ведомый, формат данных, фрейм пакета, полярность тактов и пр.). Регистры SPI_CR1 и CRC в режиме I²S не используется. Точно также не используется бит SSOE регистра SPI_CR2 и биты MODF и CRCERR регистра SPI_SR.

Для передачи данных в 16-бит режиме I²S использует регистр данных SPI (SPI_DR).

28.4.2. Полный дуплекс I²S

Для поддержки полного дуплекса кроме I2S2 и I2S3 есть два расширения I²S (I2S2_ext, I2S3_ext). Первый I²S полного дуплекса основан на I2S2 и I2S2_ext, второй на I2S3 и I2S3_ext. Они используются только в полном дуплексе.

Рис. 263. Блок-схема полного дуплекса I²S



1. Здесь x может быть 2 или 3.

I2Sx могут работать в режиме ведущего:

- В полудуплексе выводить SCK и WS может только I2Sx
- В полном дуплексе подавать SCK и WS на I2S2_ext и I2S3_ext может только I2Sx.

28.4.3. Поддерживаемые звуковые протоколы

Для передачи звуковых данных двух мультиплексированных каналов (левого и правого) есть три линии шины и только один 16-бит регистр приёма/передачи данных. Так что писать или читать данные канала нужно сверяясь с битом CHSIDE регистра SPI_SR. Левый канал всегда передаётся первым (в протоколе PCM бит CHSIDE смысла не имеет).

Данные посылаются в форматах:

- 16-бит данные в 16-бит фрейме
- 16-бит данные в 32-bit фрейме
- 24-бит данные в 32-bit фрейме
- 32-бит данные в 32-bit фрейме

При 16-бит данных в 32-бит пакете старшие 16 бит значимые, остальные 16 бит нулевые и не требуют действий программы или запроса DMA (только одно чтение/запись).

Для 24-бит и 32-бит фреймов данных нужны два чтения/записи SPI_DR или операции DMA. Незначимые 8 бит в 24-бит фрейме аппаратно расширяются нулями до 32 бит.

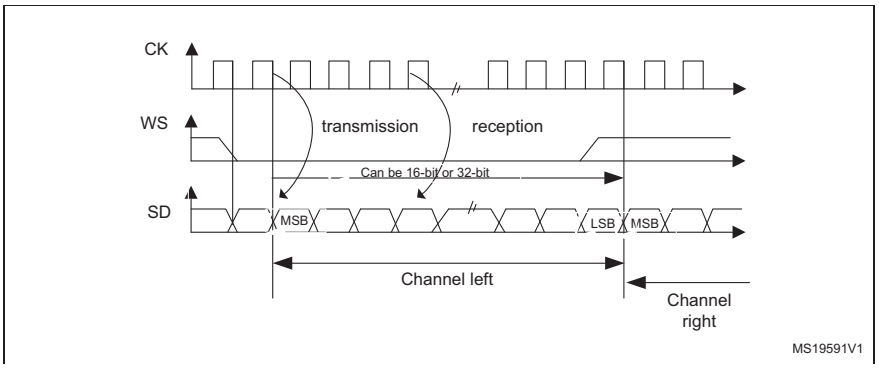
Во всех форматах данных и связи первым передаётся самый значащий бит.

Биты I2SSTD[1:0] и PCMSYNC регистра SPI_I2SCFGR определяют один из четырёх поддерживаемых I²S звуковых стандартов:

Стандарт I²S Philips

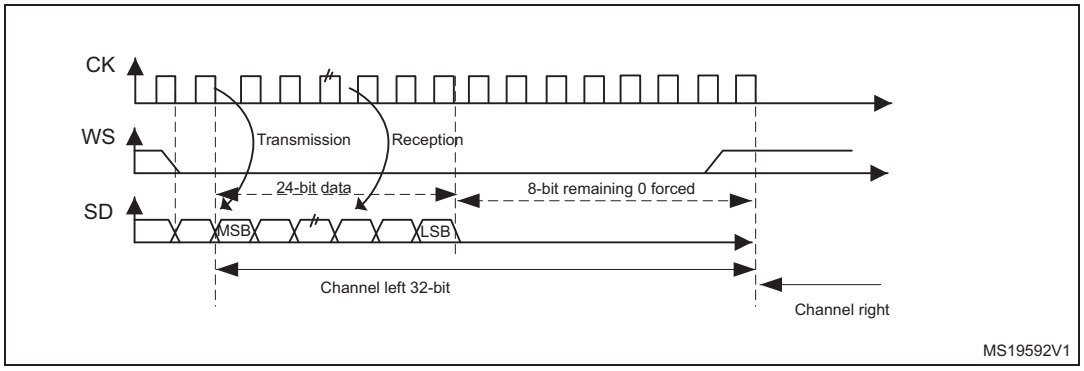
Здесь передаваемый канал указывается сигналом WS, он активируется за один такт CK перед подачей первого бита данных.

Рис. 264. Стандарт I²S Philips (16/32-бит полной точности, CPOL = 0)



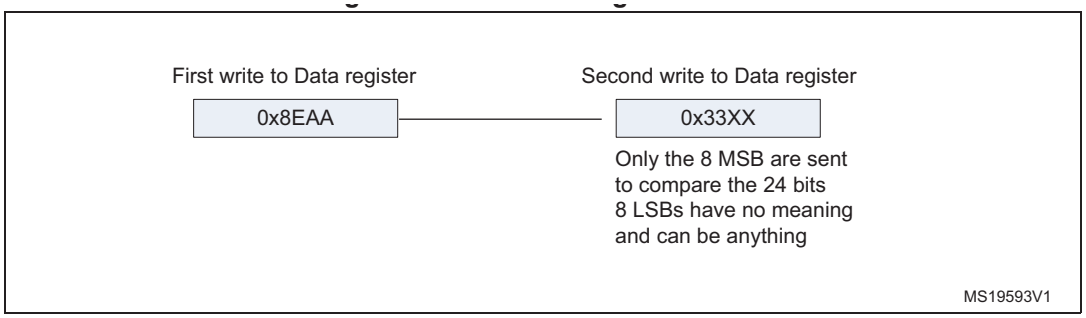
Данные фиксируются передатчиком по заднему фронту CK, а приёмником по переднему фронту. Сигнал WS считывается по заднему фронту CK.

Рис. 265. Стандарт I²S Philips (24-бит фрейм с CPOL = 0)



Этот режим требует двух операций чтения/записи `SPI_DR`.

• Посылаем 0x8EAA33 (24 бита):



• Принимаем 0x8EAA33:

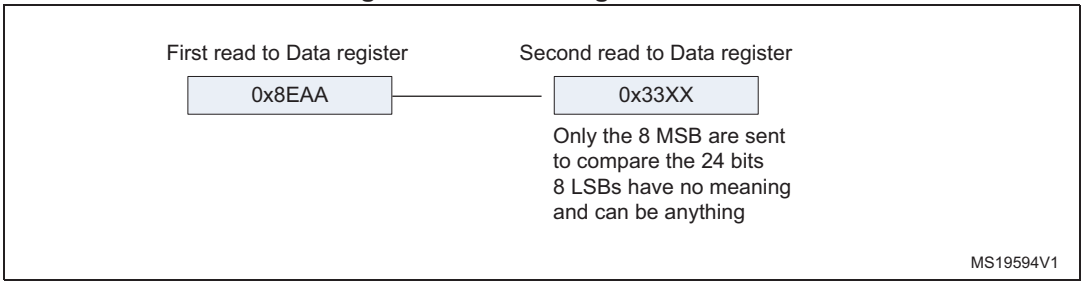
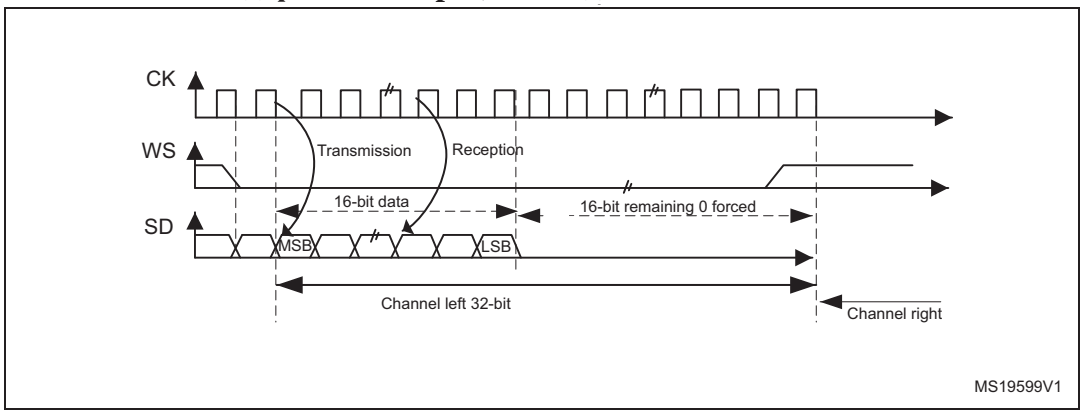
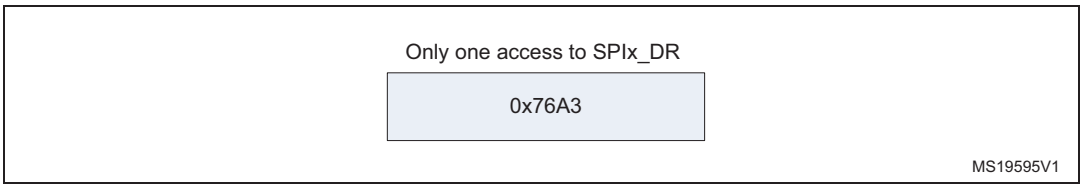


Рис. 268. Стандарт I²S Philips (16-бит данных в 32-бит пакете с CPOL = 0)



При конфигурации I²S на 16-данные в 32-бит фрейме канала требуется только одно обращение к регистру SPI_DR. Остальные 16 бит аппаратно обнуляются.

Передаём расширенное до 32 бит число 0x76A3 (0x76A30000):



При передаче, после записи старшей части в SPI_DR, ставится флаг TXE и может вызываться прерывание для записи нового передаваемого значения. Число 0x0000 пересылается аппаратно, не затрагивая SPI_DR.

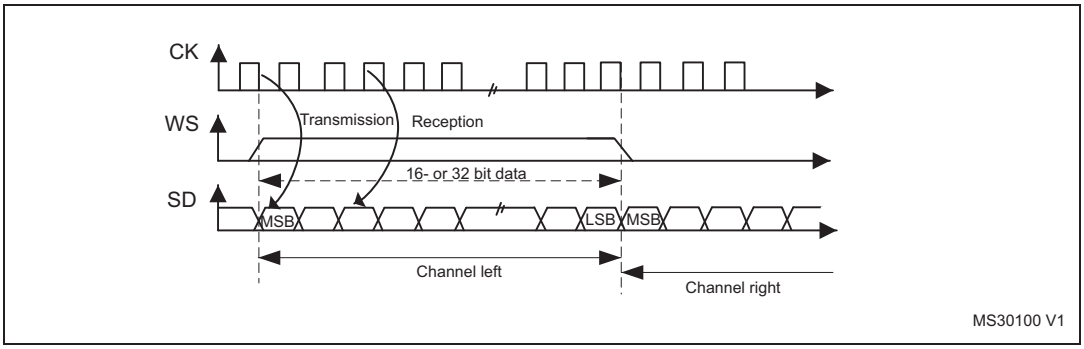
При приёме флаг RXNE ставится при получении старшего 16-бит полуслова.

Так для двух обращений к SPI_DR отводится больше времени.

Стандарт левого выравнивания

Здесь сигнал WS выдаётся одновременно с первым (старшим) битом данных.

Рис. 270. Левое выравнивание 16- или 32-бит полной точности с CPOL = 0



Данные фиксируются передатчиком по заднему фронту CK, а приёмником по переднему фронту.

Рис. 271. Левое выравнивание 24-бит фрейма с CPOL = 0

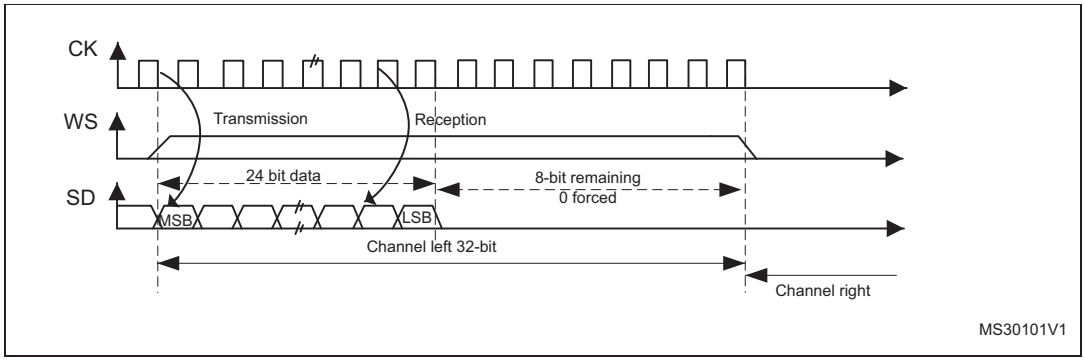
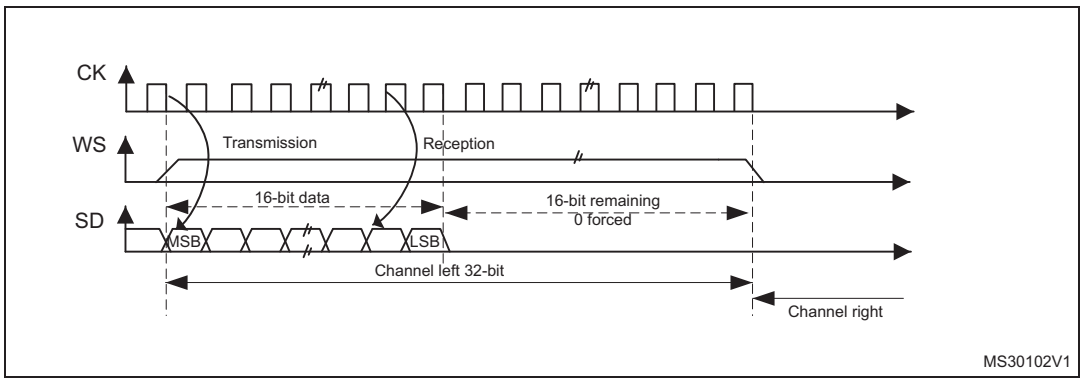


Рис. 272. Левое выравнивание 16-бит данных в 32-бит пакете с CPOL = 0



Стандарт правого выравнивания

Он подобен левому выравниванию (форматы 16-бит и 32-бит фреймов полной точности совпадают).

Рис. 273. Правое выравнивание 16- или 32-бит полной точности с CPOL = 0

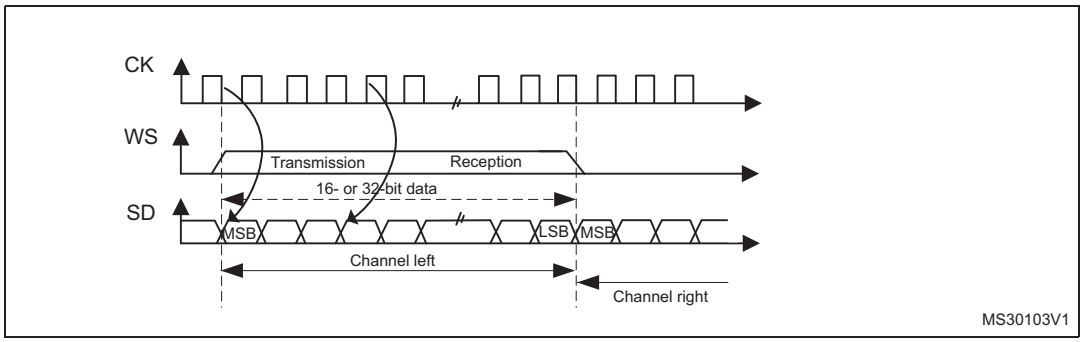
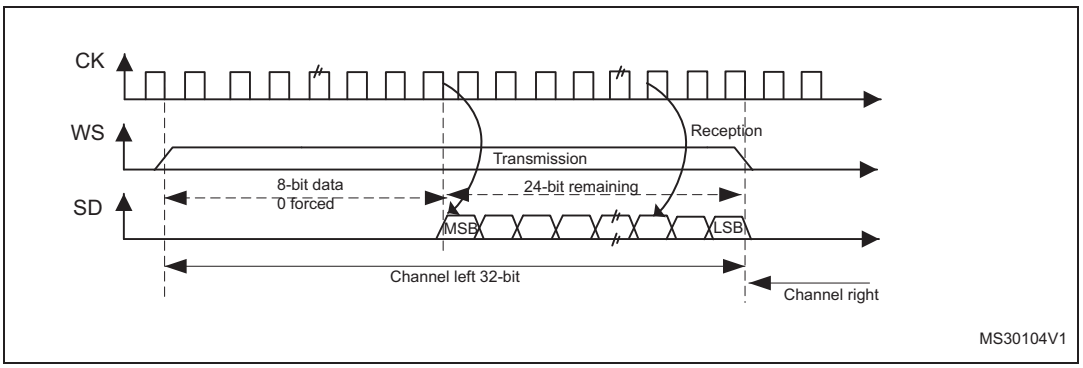
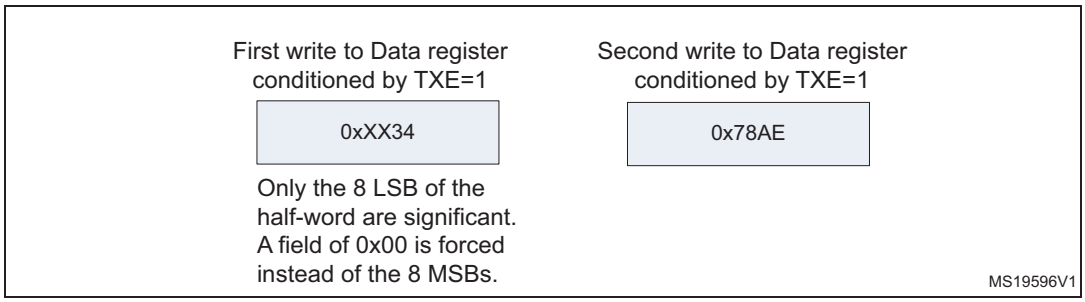


Рис. 274. Правое выравнивание 24-бит фрейма с CPOL = 0



- Передаём число 0x3478AE за две записи в регистр SPI_DR:

Рис. 275. Операции передачи 0x3478AE



Принимаем 0x3478AE за два чтения SPI_DR по событию RXNE:

Рис. 276. Операции приёма 0x3478AE

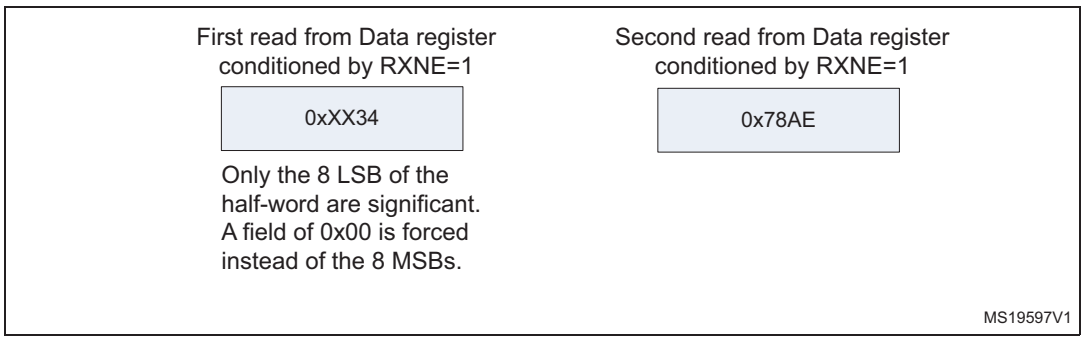
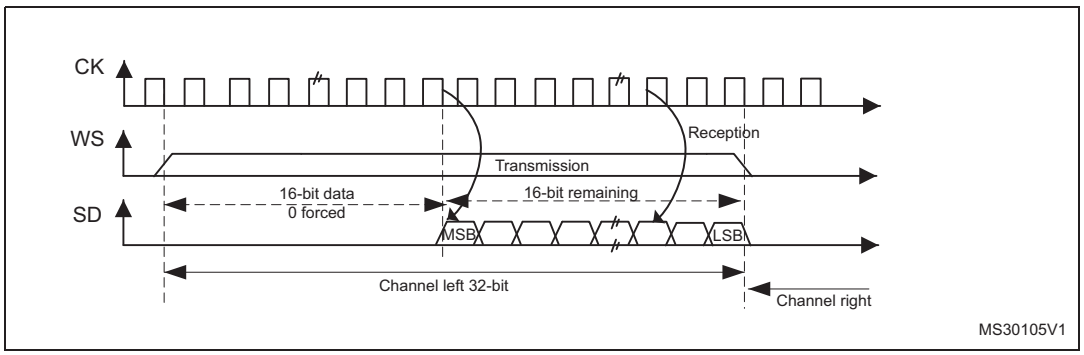


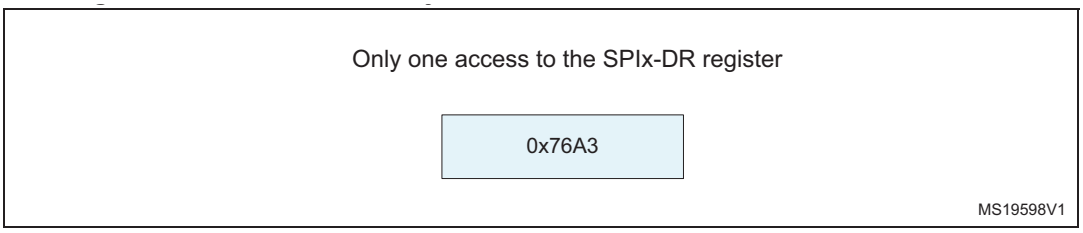
Рис. 277. Правое выравнивание 16-бит в 32-бит фрейме с CPOL = 0



При конфигурации I^{2}S на 16-данные в 32-бит фрейме канала требуется только одно обращение к регистру SPI_DR. Остальные 16 бит аппаратно обнуляются.

Передаём расширенное до 32 бит число 0x76A3 (0x0000 76A3):

Рис. 278. Пример правого выравнивания 16 бит в 32 бит фрейме



При передаче, по установке TXE пишутся передаваемые данные (здесь 0x76A3). Поле 0x0000 передаётся первым. Бит TXE снова ставится после передачи значимых данных (0x76A3) по SD.

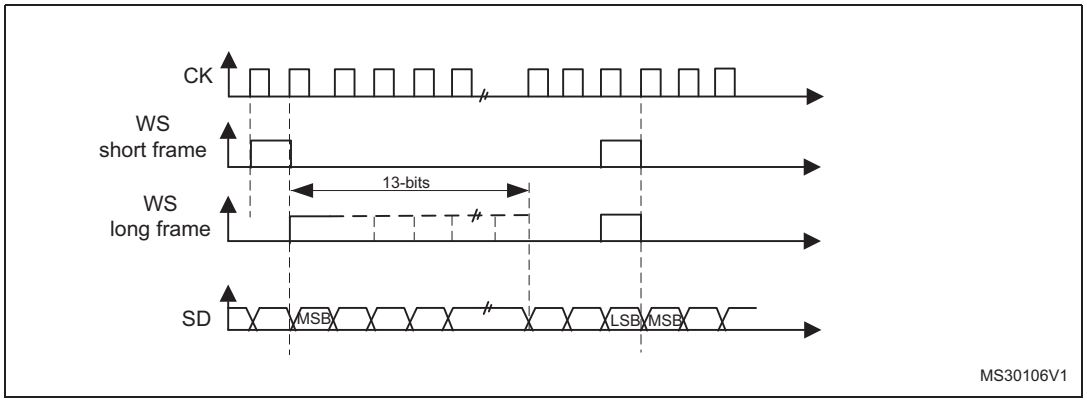
При приёме, RXNE выставляется после получения значимого полуслова (на поле 0x0000 плюём).

Так выделяется больше времени на промежуток между обращениями к SPI_DR.

Стандарт PCM

Здесь информация о канале (левый/правый) не нужна. Два режима PCM (короткий и длинный фрейм) определяются битом PCMSYNC в регистре SPI_I2SCFGR.

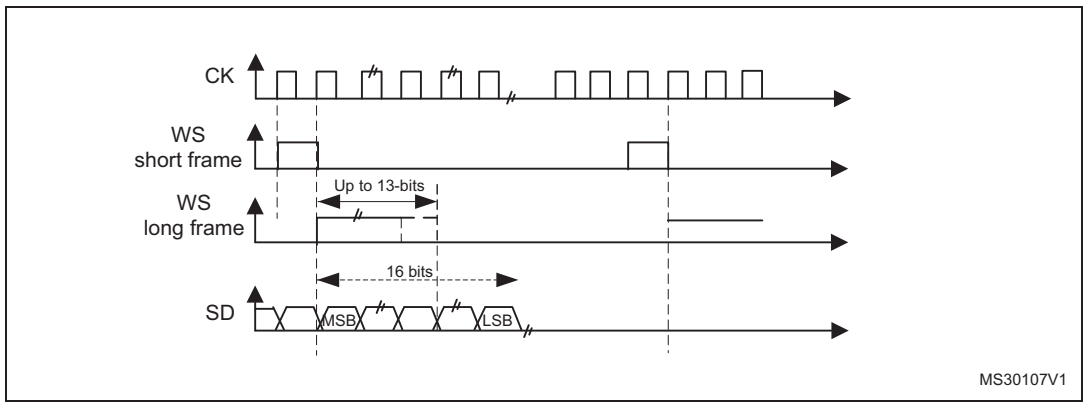
Рис. 279. Стандартные сигналы PCM (16-бит)



При длинной синхронизации фрейма сигнал WS имеет длину 13 бит в режиме ведущего.

При короткой синхронизации фрейма сигнал WS имеет длину только 1 такт.

Рис. 280. Стандартные сигналы PCM (16-бит данных в 32-бит фрейме)



NB: Для обоих режимов (ведущий и ведомый) и обоих видов синхронизации битов (короткий и длинный) надо указывать число битов между двумя последовательными участками данных (биты DATLEN и CHLEN в регистре SPI_I2SCFGR register).

28.4.4. Тактовый генератор

Битрейт I^2S определяет поток данных I^2S и частоту тактов I^2S.

Битрейт I^2S = число бит на канал × число каналов × частота дискретизации звука

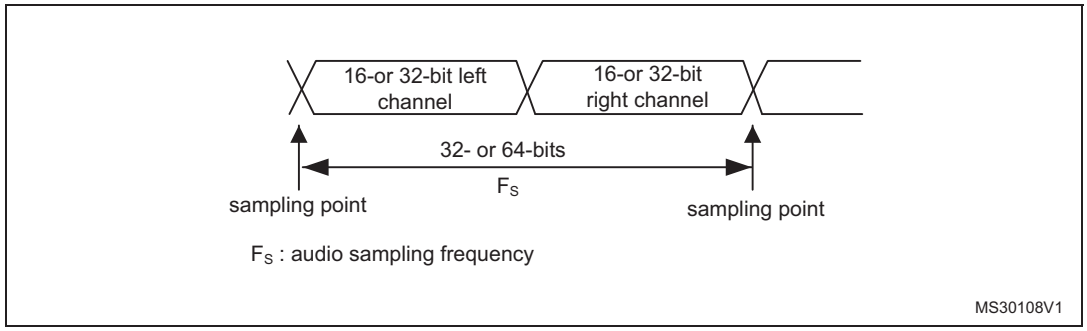
Для 16-бит звука, двух каналов битрейт I^2S вычисляется так:

Битрейт $I^2S = 16 \times 2 \times F_S$

Для 32-бит пакета он будет:

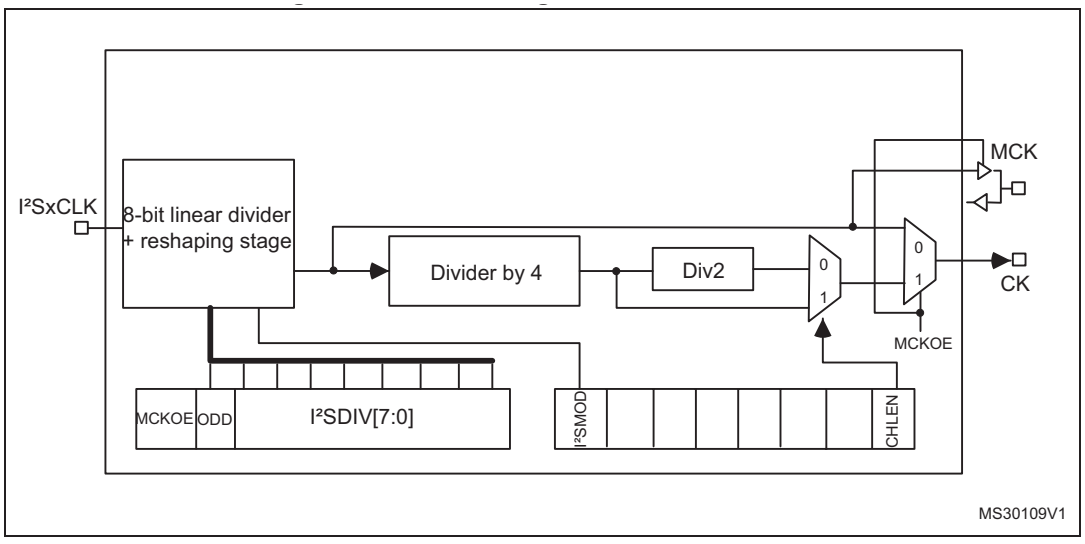
Битрейт $I^2S = 32 \times 2 \times F_S$

Рис. 281. Определение частоты дискретизации звука



Для связи с частотой дискретизации звука в режиме ведущего надо установить линейный делитель.

Рис. 282. Архитектура тактового генератора I^2S



1. Здесь x может быть 2 или 3.

I2SxCLK это системные такты (подаются от HSI, HSE или PLL, питающих такты AHB). В сетевых устройствах источником I2SxCLK могут быть SYSCLK или PLL3 VCO (2 × PLL3CLK). Это определяется битами I2S2SRC и I2S3SRC регистра RCC_CFGR2.

Частота дискретизации звука может быть 96 kHz, 48 kHz, 44.1 kHz, 32 kHz, 22.05 kHz, 16 kHz, 11.025 kHz или 8 kHz (или внутри этого диапазона). Линейный делитель определяется по следующим формулам:

При выдаче ведущих тактов (стоит бит MCKOE в регистре SPI_I2SPR):

F_S = I2SxCLK / [(16*2)*((2*I2SDIV)+ODD)*8)] при 16-бит фрейме канала

F_S = I2SxCLK / [(32*2)*((2*I2SDIV)+ODD)*4)] при 32-бит фрейме канала

При отсутствии ведущих тактов (бит MCKOE чист):

F_S = I2SxCLK / [(16*2)*((2*I2SDIV)+ODD))] при 16-бит фрейме канала

F_S = I2SxCLK / [(32*2)*((2*I2SDIV)+ODD))] при 32-бит фрейме канала

NB: Есть и другие конфигурации с оптимальной точностью тактов.

Таблица 127. Точность звуковой частоты (для PLLM VCO = 1 MHz или 2 MHz)(1)

Главные такты	Целевая f_S (Hz)	Формат данных	PLLI2SN	PLLI2SR	I2SDIV	I2SODD	Реальная f_S (Hz)	Ошибка
Выкл.	8000	16-bit	192	2	187	1	8000	0 %
		32-bit	192	3	62	1	8000	0 %
	16000	16-bit	192	3	62	1	16000	0 %
		32-bit	256	2	62	1	16000	0 %
	32000	16-bit	256	2	62	1	32000	0 %
		32-bit	256	5	12	1	32000	0 %
	48000	16-bit	192	5	12	1	48000	0 %
		32-bit	384	5	12	1	48000	0 %
	96000	16-bit	384	5	12	1	96000	0 %
		32-bit	424	3	11	1	96014.49219	0 %
	22050	16-bit	290	3	68	1	22049.87695	0 %
		32-bit	302	2	53	1	22050.23438	0 %
	44100	16-bit	302	2	53	1	44100.46875	0 %
		32-bit	429	4	19	0	44099.50781	0 %
	192000	16-bit	424	3	11	1	192028.9844	0 %
		32-bit	258	3	3	1	191964.2813	0 %
Вкл.	8000	всё равно	256	5	12	1	8000	0 %
	16000	всё равно	213	2	13	0	16000.60059	0 %
	32000	всё равно	213	2	6	1	32001.20117	0 %
	48000	всё равно	258	3	3	1	47991.07031	0 %
	96000	всё равно	344	2	3	1	95982.14063	0 %
	22050	всё равно	429	4	9	1	22049.75391	0 %
	44100	всё равно	271	2	6	0	44108.07422	0 %

(1) Это только примерные значения.

28.4.5. Ведущий I²S

I²S можно поставить:

- Ведущим на передачу или приём (полудуплекс с использованием I2Sx)
- Ведущим на передачу и приём (полный дуплекс с использованием I2Sx и I2Sx_ext).

У ведущего I²S на ножку CK выдаются такты связи, равно как и сигнал выбора слова (WS), выдача ведущих тактов (MCK) разрешается битом MCKOE регистра SPI_I2SPR.

Процедура

1. Битами I2SDIV[7:0] регистра SPI_I2SPR определяем частоту тактов связи. Также определяем бит ODD регистра SPI_I2SPR.

2. Битом CKPOL определяем уровень стабильности тактов связи. Если для внешнего звукового компонента DAC/ADC нужны такты MCK, то разрешаем их битом MCKOE регистра SPI_I2SPR (значения I2SDIV и ODD зависят от наличия MCK).

3. Битом I2SMOD регистра SPI_I2SCFGR активируем функциональность I^2S, битами I2SSTD[1:0] PCMSYNC выбираем стандарт I^2S, длину данных битами DATLEN[1:0] и число битов на канал битом CHLEN. Включаем режим ведущего I^2S и направление битами I2SCFG[1:0] регистра SPI_I2SCFGR.

4. По нужде выбираем потенциальные источники прерываний и возможности DMA в регистре SPI_CR2.

5. Ставим бит I2SE регистра SPI_I2SCFGR.

WS и CK в выходном режиме. Вывод MCK разрешается битом MCKOE регистра SPI_I2SPR.

Последовательность передачи

Начинается записью первого полуслова левого канала в буфер Tx.

Передача буфера Tx в регистр сдвига ставит флаг TXE, умоляя записать в Tx полуслово правого канала. Флаг CHSIDE указывает какой канал надо передавать следующим. Он имеет смысл при высоком флаге TXE, так как изменяется во время установки TXE.

Полный фрейм это передача данных левого и затем правого канала, по отдельности один канал передать нельзя.

Полуслово данных параллельно пишется в 16-бит регистр сдвига, откуда выдвигается на ножку MOSI/SD, начиная со старшего бита. Флаг TXE ставится каждой передачей из Tx в регистр сдвига, а ещё может выдаваться прерывание, если разрешено битом TXEIE в регистре SPI_CR2.

Для непрерывной передачи, следующее данное надо писать в регистр SPI_DR до конца текущей передачи.

Для выключения I^2S очисткой бита I2SE, надо сначала дождаться TXE = 1 и BSY = 0.

Последовательность приёма

Работа подобна передаче, кроме пункта 3, где битами I2SCFG[1:0] надо поставить приём ведущего.

Независимо от длины данных и канала данные принимаются 16-бит пакетами, то есть приём отсчёта канала может занимать до двух чтений буфера Rx. После каждого приёма в буфер Rx ставится флаг RXNE и выдаётся прерывание, разрешённое битом RXNEIE регистра SPI_CR2.

Бит RXNE чистится чтением регистра SPI_DR. CHSIDE обновляется после каждого приёма, он чувствителен к сигналу WS от ячейки I^2S.

В случае приёма данного при ещё не прочитанном предыдущем, ставится флаг OVR. Прерывание переполнения разрешается битом ERRIE регистра SPI_CR2.

Для выключения I^2S нужно правильно остановить текущие передачи в соответствии с установками. В случае:

- 16-бит данных 32-бит канала (DATLEN=00 и CHLEN=1) с правым выравниванием (I2SSTD=10)
  a) Ждём предпоследнего RXNE=1(n–1)
  b) Ждём 17 тактов I^2S (в программном цикле)
  c) Выключаем I^2S (I2SE = 0)
- 16-бит данных 32-бит канала (DATLEN=00 и CHLEN=1) с левым выравниванием, I^2S или PCM (I2SSTD = 00, I2SSTD = 01 или I2SSTD = 11, соответственно)
  a) Ждём последнего RXNE
  b) Ждём 1 такт I^2S (в программном цикле)
  c) Выключаем I^2S (I2SE = 0)
- Для остальных комбинаций DATLEN и CHLEN, независимо от битов I2SSTD:
  a) Ждём предпоследнего RXNE=1(n–1)
  b) Ждём 1 такт I^2S (в программном цикле)
  c) Выключаем I^2S (I2SE = 0)

NB: Во время передач флаг BSY остаётся низким.

28.4.6. Ведомый I²S

I²S можно поставить:

- Ведомым на передачу или приём (полудуплекс с использованием I2Sx)
- Ведомым на передачу и приём (полный дуплекс с использованием I2Sx и I2Sx_ext).

Здесь такты на интерфейс I²S не выдаются, они и сигнал WS это входы.

Конфигурируем:

1. Битом I2SMOD регистра SPI_I2SCFGR активируем функциональность I²S, битами I2SSTD[1:0] выбираем стандарт I²S, длину данных битами DATLEN[1:0] и число битов на канал битом CHLEN. Включаем режим ведомого I²S и направление битами I2SCFG[1:0] регистра SPI_I2SCFGR.
2. По нужде выбираем потенциальные источники прерываний и возможности DMA в регистре SPI_CR2.
3. Ставим бит I2SE регистра SPI_I2SCFGR.

Последовательность передачи

Начинается при появлении тактов от ведущего и сигнала NSS_WS запроса передачи. Ведомый должен быть уже включён и регистр данных записан.

В режимах I²S, левого и правого выравнивания первым пишется левый канал. При старте данное из буфера Tx пишется в регистр сдвига. Ставится флаг TXE, чтобы в регистр данных мы записали данное правого канала.

Флаг CHSIDE показывает, какой канал надо передавать. Он зависит от сигнала WS от внешнего ведущего. То есть ведомый должен быть готов к передаче первого данного левого канала до подачи тактов. Сигнал WS выставляется для первого левого канала.

NB: Бит I2SE надо писать не менее чем за два такта PCLK до вывода первого такта на линию CK.

Первое полуслово данных пишется в регистр сдвига вовремя передачи первого бита и затем выдвигается на ножку MOSI/SD начиная со старшего бита. Флаг TXE ставится после каждой передачи из Tx в регистр сдвига и выдаётся прерывание, если разрешено битом TXEIE в регистре SPI_CR2.

Стоящий флаг TXE надо проверять перед попыткой записи в буфер Tx.

Для непрерывных передач регистр SPI_DR надо писать до завершения текущей передачи. Если до первого фронта тактов следующей передачи регистр SPI_DR ещё не записан, то ставится флаг UDR ошибки исчерпания. Прерывание вызывается при стоящем бите ERRIE регистра SPI_CR2. В этом случае надо обязательно выключать I²S для рестарта передачи начиная с левого канала.

I²S выключается очисткой бита I2SE с обязательным ожиданием TXE = 1 и BSY = 0.

Последовательность приёма

Работа подобна передаче, кроме пункта 1, где битами I2SCFG[1:0] регистра SPI_I2SCFGR надо поставить приём ведомого.

Независимо от длины данных и канала данные принимаются 16-бит пакетами, то есть приём отсчёта канала может занимать до двух чтений буфера Rx. После каждого приёма в буфер Rx ставится флаг RXNE и выдаётся прерывание, разрешённое битом RXNEIE регистра SPI_CR2.

Бит RXNE чистится чтением регистра SPI_DR. CHSIDE обновляется после каждого приёма, он чувствителен к внешнему сигналу WS.

В случае приёма данного при ещё не прочитанном предыдущем, ставится флаг OVR. Прерывание переполнения разрешается битом ERRIE регистра SPI_CR2.

Выключается приём I²S очисткой бита I2SE сразу после появления последнего RXNE = 1.

28.4.7. Флаги состояния

Их три:

Занят (BSY)

Флаг BSY ставится и снимается аппаратно, указывая состояние уровня связи I^2S.

В режиме приёма ведущего (I2SCFG = 11) флаг BSY удерживается низким.

Смотреть на флаг BSY надо перед выключением I^2S.

Очищается флаг BSY:

- при завершении передачи (кроме непрерывной передачи ведущего)
- при выключении I^2S

При непрерывных передачах:

- В режиме передачи ведущего флаг BSY удерживается высоким
- В режиме ведомого флаг BSY снимается на 1 такт I^2S между всеми передачами

NB: Не используйте флаг BSY для определения конца передачи, TXE и RXNE лучше.

Буфер Tx пуст (TXE)

Флаг TXE сбрасывается при записи в буфер Tx и у выключенного I^2S (бит I2SE снят).

Буфер RX не пуст (RXNE)

В буфере RX есть данное. Снимается при чтении регистра SPI_DR.

Флаг канала (CHSIDE)

При передаче флаг CHSIDE переключается во время установки флага TXE. Показывает номер передаваемого канала (левый/правый). В случае ошибки исчерпания от теряет смысл и I^2S надо выключать и включать для восстановления передач, начиная с левого канала.

При приёме он переключается при записи полученного данного в регистр SPI_DR. Показывает номер полученного канала. В случае ошибки (вроде OVR) от теряет смысл и I^2S надо выключать и включать для восстановления передач, начиная с левого канала.

В режиме PCM он не нужен.

Флаги ошибки OVR и UDR в регистре SPI_SR ставят бит ERRIE регистра SPI_CR2 и выдают прерывание. Прерывание снимается чтением регистра SPI_SR после очистки источника прерывания.

28.4.8. Флаги ошибок

Есть две ошибки ячейки I^2S.

Исчерпание (UDR)

При передаче ведомого он ставится по первому такту передачи при ещё не записанном регистре SPI_DR. Возможен при стоящем бите I2SMOD в регистре SPI_I2SCFGR. Прерывание возможно по стоящему биту ERRIE регистр SPI_CR2.

Бит UDR снимается чтением регистра SPI_SR.

Переполнение (OVR)

Ставится при получении данного с ещё не прочитанным регистром SPI_DR. Новое и все последующие данные теряются. Прерывание возможно по стоящему биту ERRIE регистр SPI_CR2.

Снимается бит OVR чтением регистра SPI_DR с последующим чтением регистра SPI_SR.

Ошибка фрейма (FRE)

Ставится в режиме ведомого, когда ведущий меняет линию WS в неподходящий момент. Исправляют ситуацию так:

1. Выключаем I^2S
2. Включаем его при появлении нужного уровня на линии WS.

Прерывания по биту FRE разрешают битом ERRIE. Снимается чтением регистра состояния.

28.4.9. Прерывания I²S

Прерывание	Флаг события	Бит разрешения
Передающий буфер пуст	TXE	TXEIE
Приёмный буфер не пуст	RXNE	RXNEIE
Переполнение	OVR	ERRIE
Исчерпание	UDR	
Ошибка фрейма	FRE	ERRIE

28.4.10. I²S с DMA

Здесь DMA работает так же как и в SPI. Вот только контроля CRC нет.

28.5. Регистры SPI и I²S

Регистры доступны полусловами (16 бит) и словами (32 бит).

28.5.1. Регистр 1 управления SPI (SPI_CR1) (в I²S не используется)

Смещение адреса: 0x00

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BIDI MODE	BIDI OE	CRC EN	CRC NEXT	DFF	RX ONLY	SSM	SSI	*LSB FIRST*	SPE	BR [2:0]			MSTR	CPOL	CPHA
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Бит 15 **BIDIMODE:** Двунаправленный режим данных

0: Однонаправленные данные по 2 линиям

1: Двунаправленные данные по 1 линии

NB: В режиме I²S не используется

— Бит 14 **BIDIOE:** Разрешение вывода двунаправленного режима

Вместе с битом BIDIMODE определяет направление двунаправленной передачи

0: Выключен (только приём)

1: Включён (только передача)

NB: В режиме I²S не используется.

У ведущего используется ножка MOSI, у ведомого - MISO.

— Бит 13 **CRCEN:** Разрешение вычисления CRC

0: Нельзя

1: Нужно

NB: В режиме I²S не используется. Пишется при выключенном SPI (SPE = '0').

— Бит 12 **CRCNEXT:** CRC следующий

0: Фаза данных

1: Следующая фаза CRC

NB: В режимах полного дуплекса и только передачи CRCNEXT надо ставить сразу после записи регистра SPI_DR.

В режиме только приёма CRCNEXT должен стоять после приёма предпоследнего данного.

При передачах с DMA должен быть чистым. В режиме I²S не используется.

— Бит 11 **DFF:** Формат фрейма данных

0: 8-бит фрейм данных

1: 16-бит фрейм данных

NB: В режиме I²S не используется. Пишется при выключенном SPI (SPE = '0').

— Бит 10 **RXONLY:** Только приём

Вместе с битом BIDIMODE определяет направление передачи в однонаправленном режиме по 2 линиям. Полезен в системе с несколькими ведомыми при недоступности одного, вывод от доступного не нарушается.

0: Полный дуплекс (приём и передача)

1: Вывод запрещён (только приём)

NB: В режиме I²S не используется.

— Бит 9 **SSM:** Программное управление ведомым

При стоящем бите SSM ввод от ножки NSS подменяется значением бита SSI.

0: Выключено

1: Включено

NB: В режиме I^2S не используется.

— Бит 8 **SSI:** Внутренний выбор ведомого

Действует при стоящем бите SSM. Вместо ножки NSS на выбор режима выдаётся этот бит.

NB: В режиме I^2S не используется.

— Бит 7 **LSBFIRST:** Формат фрейма

0: Старший бит передаётся первым

1: Младший бит передаётся первым

NB: В режиме I^2S не используется. Во время передачи изменять нельзя.

— Бит 6 **SPE:** Включение SPI

0: Выключен

1: Включён

NB: В режиме I^2S не используется.

— Биты 5:3 **BR[2:0]:** Скорость передачи

000: $f_{PCLK}/2$

001: $f_{PCLK}/4$

010: $f_{PCLK}/8$

011: $f_{PCLK}/16$

100: $f_{PCLK}/32$

101: $f_{PCLK}/64$

110: $f_{PCLK}/128$

111: $f_{PCLK}/256$

NB: В режиме I^2S не используется. Во время передачи изменять нельзя.

— Бит 2 **MSTR:** Выбор ведущего

0: Ведомый

1: Ведущий

NB: В режиме I^2S не используется. Во время передачи изменять нельзя.

— Бит 1 **CPOL:** Полярность тактов

0: CK равен 0 в простое

1: CK равен 1 в простое

NB: В режиме I^2S не используется. Во время передачи изменять нельзя.

— Бит 0 **CPHA:** Фаза тактов

0: Фронт считывания в первом такте

1: Фронт считывания во втором такте

NB: В режиме I^2S не используется. Во время передачи изменять нельзя.

28.5.2. Регистр 2 управления SPI (SPI_CR2)

Смещение адреса: 0x04

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TXEIE	RXNEIE	ERRIE	FRF	Res.	*SSOE*	TXDMAEN	RXDMAEN
								rw	rw	rw	rw		rw	rw	rw

— Биты 15:8 Резерв, не трогать.

— Бит 7 **TXEIE:** Разрешение прерывания пустого буфера Tx

0: Прерывание TXE запрещено

1: Прерывание TXE разрешено.

— Бит 6 **RXNEIE:** Разрешение прерывания непустого буфера RX

0: Прерывание RXNE запрещено

1: Прерывание RXNE разрешено.

— Бит 5 **ERRIE:** Разрешение прерывания ошибки

Прерывание по ошибке (CRCERR, OVR, MODF в режиме SPI и UDR, OVR в режиме I^2S).

0: Прерывание ошибки запрещено

1: Прерывание ошибки разрешено.

— Бит 4 **FRF:** Формат фрейма

0: SPI Motorola

1: SPI TI

— Бит 3 Резерв, не трогать.

— Бит 2 **SSOE:** Разрешение вывода SS

0: В режиме ведущего вывод SS выключен, ячейка может работать режиме нескольких ведущих

1: В режиме ведущего вывод SS включен, ячейка не может работать режиме нескольких ведущих

NB: В режимах I^2S и SPI TI не используется.

— Бит 1 **TXDMAEN:** Разрешение DMA буфера Tx

При стоящем бите запрос DMA выдаётся при установке флага TXE.

0: Нельзя

1: Можно

— Бит 0 **RXDMAEN:** Разрешение DMA буфера Rx

При стоящем бите запрос DMA выдаётся при установке флага RXNE.

0: Нельзя

1: Можно

28.5.3. Регистр состояния SPI (SPI_SR)

Смещение адреса: 0x08

По сбросу: 0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							FRE	BSY	OVR	MODF	CRC ERR	UDR	CHSIDE	TXE	RXNE
							r	r	r	r	rc_w0	r	r	r	r

— Биты 15:9 Резерв, не трогать.

— Бит 8 **FRE:** Ошибка формата фрейма

Ставится аппаратно, снимается чтением регистра SPIx_SR.

0: Нету

1: Была

NB: Используется в режиме ведомого SPI TI и I2S

— Бит 7 **BSY:** Занят

Ставится и снимается аппаратно.

0: SPI (или I^2S) свободен

1: SPI (или I^2S) занят связью или буфер Tx не пуст

— Бит 6 **OVR:** Переполнение

Ставится аппаратно, снимается программной последовательностью

0: Не было

1: Было

— Бит 5 **MODF:** Сбой режима

Ставится аппаратно, снимается программной последовательностью

0: Не было

1: Было

NB: В режиме I^2S не используется.

— Бит 4 **CRCERR:** Ошибка CRC

Ставится аппаратно, снимается записью 0

0: Значение CRC совпало с SPI_RXCRCR

1: Значение CRC не совпало с SPI_RXCRCR

NB: В режиме I^2S не используется.

— Бит 3 **UDR:** Исчерпание

Ставится аппаратно, снимается программной последовательностью

0: Не было

1: Было

NB: В режиме SPI не используется.

— Бит 2 **CHSIDE**: Номер канала

0: Надо передать или принят левый канал

1: Надо передать или принят правый канал

NB: В режиме SPI не используется. В PCM не имеет смысла.

— Бит 1 **TXE:** Буфер Tx пуст

0: Не пуст

1: Пуст

— Бит 0 **RXNE:** Буфер Rx не пуст

0: Пуст

1: Не пуст

28.5.4. Регистр данных SPI (SPI_DR)

Смещение адреса: 0x0C

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:0 **DR[15:0]:** Регистр данных приёма или передачи

Это 2 регистра, Tx для записи и Rx для чтения, расположенных по одному адресу.

NB: Примечания для SPI:

Формат данных (8- или 16-бит) надо выбирать битом DFF регистра SPI_CR1 до включения SPI.

При 8-бит фрейме данных используется только младший байт (SPI_DR[7:0]), при приёме старший байт (SPI_DR[15:8]) аппаратно обнуляется.

При 16-бит фрейме данных используется весь регистр.

28.5.5. Регистр полинома CRC SPI (SPI_CRCPR) (в I²S не используется)

Смещение адреса: 0x10

По сбросу: 0x0007

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRCPOLY[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 15:0 **CRCPOLY[15:0]:** Полином вычисления CRC

По сбросу содержит число 0007h. Можно и другой.

NB: В режиме I²S не используется.

28.5.6. Регистр RX CRC SPI (SPI_RXCRCR) (в I²S не используется)

Смещение адреса: 0x14

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXCRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

— Биты 15:0 **RXCRC[15:0]:** Rx CRC

При включённом вычислении CRC биты RxCRC[15:0] содержат CRC последовательно принятых байтов. Сбрасывается записью 1 в бит CRCEN регистра SPI_CR1.

В вычислениях участвуют младшие 8 бит (стандарт CRC8) или весь регистр (стандарт CRC16) в зависимости от бита DFF регистра SPI_CR1

NB: При стоящем флаге BSY читается ерунда. В режиме I²S не используется.

28.5.7. Регистр TX CRC SPI (SPI_TXCRCR) (в I²S не используется)

Смещение адреса: 0x18

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXCRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

— Биты 15:0 **TXCRC[15:0]:** Rx CRC

При включённом вычислении CRC биты TxCRC[15:0] содержат CRC последовательно переданных данных. Сбрасывается записью 1 в бит CRCEN регистра SPI_CR1.

В вычислениях участвуют младшие 8 бит (стандарт CRC8) или весь регистр (стандарт CRC16) в зависимости от бита DFF регистра SPI_CR1

NB: При стоящем флаге BSY читается ерунда. В режиме I²S не используется.

28.5.8. Регистр конфигурации SPI_I²S (SPI_I2SCFGR)

Смещение адреса: 0x1C

По сбросу: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				I2SMOD	I2SE	I2SCFG		PCMSYNC	Res.	I2SSTD		CKPOL	DATLEN		CHLEN
				rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw

— Биты 15:12 Резерв, не трогать.

— Бит 11 **I2SMOD**: Режим I²S

0: Режим SPI

1: Режим I²S

NB: Писать можно при выключенных SPI и I²S

— Бит 10 **I2SE**: Включение I²S

0:Выключен

1: Включён

NB: В режиме SPI не используется.

— Биты 9:8 **I2SCFG**: Конфигурация I²S

00: Ведомый - передача

01: Ведомый - приём

10: Ведущий - передача

11: Ведущий - приём

NB: В режиме SPI не используется. Писать можно при выключенном I²S.

— Бит 7 **PCMSYNC**: Синхронизация фрейма PCM

0: Короткая

1: Длинная

NB: В режиме SPI не используется. Имеет значение только при I2SSTD = 11 (стандарт PCM).

— Бит 6 Резерв, не трогать.

— Биты 5:4 **I2SSTD**: Стандарт I²S

00: Стандарт I²S Philips.

01: Левое выравнивание

10: Правое выравнивание

11: Стандарт PCM

NB: В режиме SPI не используется. Писать можно при выключенном I²S.

— Бит 3 **CKPOL**: Полярность стабильного такта I²S

0: Низкий

1: Высокий

NB: В режиме SPI не используется. Писать можно при выключенном I²S.

— Биты 2:1 **DATLEN**: Длина передаваемых данных

00: 16-бит

01: 24-бита

10: 32-бита

11: Нельзя

NB: В режиме SPI не используется. Писать можно при выключенном I²S.

— Бит 0 **CHLEN**: Длина канала (число бит канала)

0: 16-бит

1: 32-бита

Имеет смысл только при DATLEN = 00, иначе аппаратно равна 32-бита, независимо от этого значения.

NB: В режиме SPI не используется. Писать можно при выключенном I²S.

28.5.9. Регистр конфигурации SPI_I²S (SPI_I2SCFGR)

Смещение адреса: 0x20

По сбросу: 0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						MCKOE	ODD	I2SDIV							
						rw	rw	rw							

— Биты 15:10 Резерв, не трогать.

— Бит 9 **MCKOE**: Разрешение вывода ведущих тактов

0: Нельзя

1: Нужно

NB: Писать можно при выключенном I²S. Используется только в ведущем I²S.

— Бит 8 **ODD**: Нечётный предделитель

0: предделитель = I2SDIV *2

1: предделитель = (I2SDIV * 2)+1

NB: Писать можно при выключенном I²S. Используется только в ведущем I²S.

— Биты 7:0 **I2SDIV**: Линейный предделитель I²S

Значения I2SDIV [7:0] = 0 и I2SDIV [7:0] = 1 запрещены.

NB: Писать можно при выключенном I²S. Используется только в ведущем I²S.

28.5.10.Карта регистров SPI

Таблица 129. Карта регистров SPI

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	SPI_CR1	Reserved																BIDIMODE	BIDIOE	CRCEN	CRCNEXT	DFF	RXONLY	SSM	SSI	*LSBFIRST*	SPE	BR [2:0]			MSTR	CPOL	CPHA
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	SPI_CR2	Reserved																								TXEIE	RXNEIE	ERRIE	FRF	Reserved	*SSOE*	TXDMAEN	RXDMAEN
	Reset value																									0	0	0	0		0	0	0
0x08	SPI_SR	Reserved																							FRE	BSY	OVR	MODF	CRCERR	UDR	CHSIDE	TXE	RXNE
	Reset value																								0	0	0	0	0	0	0	1	0
0x0C	SPI_DR	Reserved																DR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	SPI_CRCPR	Reserved																CRCPOLY[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0x14	SPI_RXCRCR	Reserved																RxCRC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	SPI_TXCRCR	Reserved																TxCRC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	SPI_I2SCFGR	Reserved																				I2SMOD	I2SE	I2SCFG		PCMSYNC	Reserved	I2SSTD		CKPOL	DATLEN		CHLEN
	Reset value																					0	0	0	0	0		0	0	0	0	0	0
0x20	SPI_I2SPR	Reserved																						MCKOE	ODD	I2SDIV							
	Reset value																							0	0	0	0	0	0	0	0	1	0

29. Последовательный интерфейс звука (SAI)

Только для STM32F42xxx и STM32F43xxx.

29.1. Введение

Он очень, очень широко совместимый. Тут есть два совершенно независимых звуковых блока. У каждого до 4 ножек (SD, SCK, FS, MCLK). При синхронной работе некоторые ножки могут быть общими. Ножка MCLK может работать на вход и на выход.

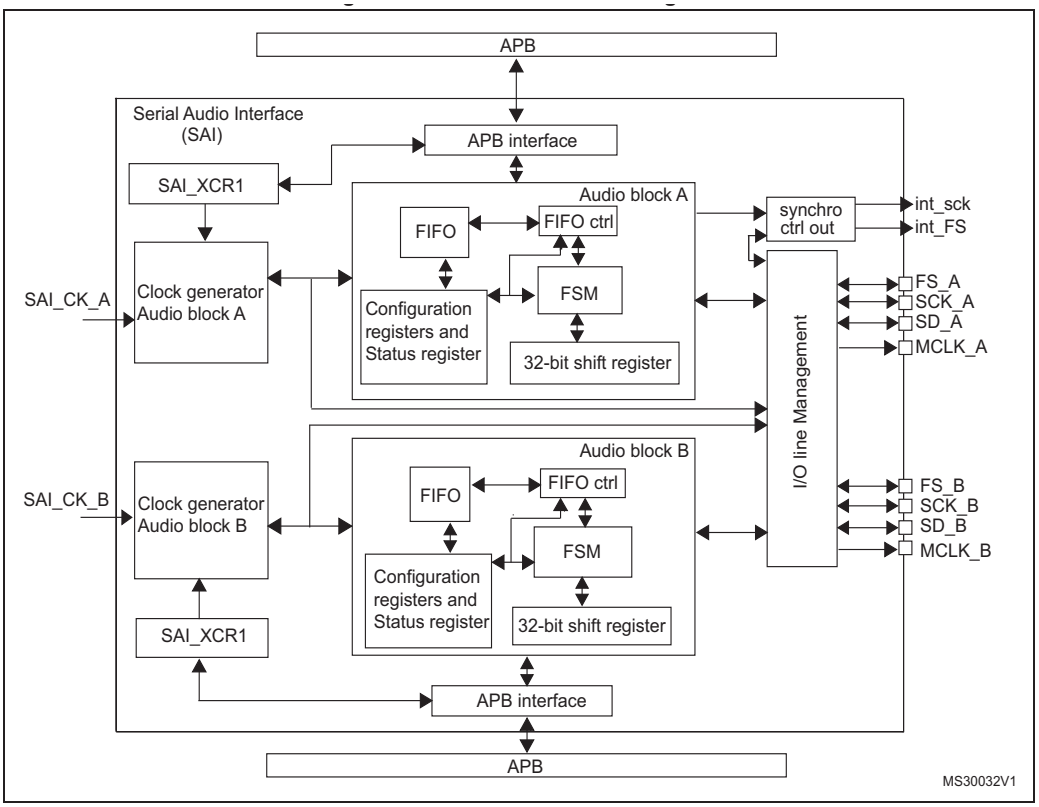
SAI может быть ведущим или ведомым, каждый блок - приёмником или передатчиком и работать синхронно с другим.

29.2. Основные свойства SAI

- Два независимых звуковых блока со своими FIFO.
- Интегрированные 8-словные FIFO у каждого блока.
- Синхронный или асинхронный режим между блоками.
- Независимая конфигурация ведущего или ведомого у каждого блока.
- Независимая частота выборки блоков в режиме двух ведущих.
- Размер данных: 8-, 10-, 16-, 20-, 24-, 32-бита.
- Очень гибкое устройство, есть протоколы: I2S, LSB или MSB, PCM/DSP, TDM, AC'97 и др.
- До 16 слотов переменного размера с возможностью выбора активного во фрейме.
- Задаваемое число битов во фрейме.
- Настройка синхронизации фрейма (смещение, длина, уровень).
- Настройка позиции первого активного бита во фрейме.
- Передача первым старшего или младшего бита.
- Немой режим.
- Моно/стерео фреймы.
- Задание фронта строба связи (SCK).
- Флаги ошибок с прерываниями:
  - Переполнение и Исчерпание,
  - Ранняя синхронизация фрейма у ведомого,
  - Поздняя синхронизация фрейма у ведомого,
  - Кодек не готов для приёма AC'97.
- Источники прерываний:
  - Ошибки,
  - Запросы FIFO.
- Интерфейс DMA с 2 каналами для доступа к FIFO каждого блока.

29.3. Блок-схема

Рис. 283. Блок-схема



SAI состоит из двух независимых блоков со своими тактовыми генераторами. Блок это 32-бит регистр сдвига со своей машиной состояний. Данные хранятся в своём FIFO с доступом от CPU и DMA. Блоки могут синхронизироваться между собой.

При синхронизации блоков контроллер линий I/O освобождает ножку FS, SCK и иногда MCLK.

Машину состояний можно настроить на множество звуковых протоколов.

Блок может быть ведущим или ведомым, приёмником или передатчиком. Такты битов SCK и синхросигнал фрейма выдаёт ведущий. В протоколе AC'97 сигнал FS всегда будет на выходе даже у ведомого.

29.4. Основные режимы SAI

Режим ведущий/ведомый задаётся битом MODE[0] в регистре SAI_xCR1 блока.

У ведущего блока:

- Такты бита идут на свою ножку SCK_x.
- Теперь она выходная.

У ведомого блока:

- Он включается раньше ведущего.
- В асинхронном режиме ножка SCK это вход.
- Ножка SCK блока, объявленного синхронным, освобождается и внутренне подключается к ножке другого синхронизируемого блока.

Режим приёмник/передатчик задаётся битом MODE[1] в регистре SAI_xCR1 блока. Ножка SD станет входом или выходом соответственно.

Можно объявить два ведущих асинхронных блока с разными частотами MCLK и SCK.

Блоки включаются битом SAIxEN в своём регистре SAI_xCR1. После этого они реагируют на линии тактов, данных и синхронизации у ведомого.

В режиме TX ведущего включённый блок сразу же выдаёт такт бита для ведомых, даже если в FIFO нет данных. Но выдача сигнала FS обусловлена наличием данных в FIFO. Полученное в FIFO первое данное сразу же передаётся ведомым. Если из FIFO передавать нечего, то в фрейме передаются нули 0 и ставится флаг исчерпания.

В режиме ведомого фрейм начинается при включении блока и появлении старта фрейма.

В режиме TX ведомого исчерпание невозможно, в этом случае обязательная последовательность такова:

1. Пишем в SAI_xDR (программой или DMA).
2. Ждём флага порога FIFO (FLH) отличным от 000b (FIFO пуст).
3. Включаем блок на передачу ведомого.

29.5. Режим синхронизации SAI

Внутренняя синхронизация

Блок, объявленный синхронным, увидит свои ножки SCK_x, FS_x и MCLK_x свободными, у асинхронного ведущего блока они просматриваются.

Обычно синхронный режим используется для полного дуплекса SAI. Один может быть ведущим, другой ведомым или оба ведомые; один асинхронный (бит SYNCEN[1:0] = 00 в SAI_xCR1) другой синхронный (бит SYNCEN[1:0] = 01 в SAI_xCR1).

NB: Частота APB PCLK должна быть больше или равна двукратному битрейту.

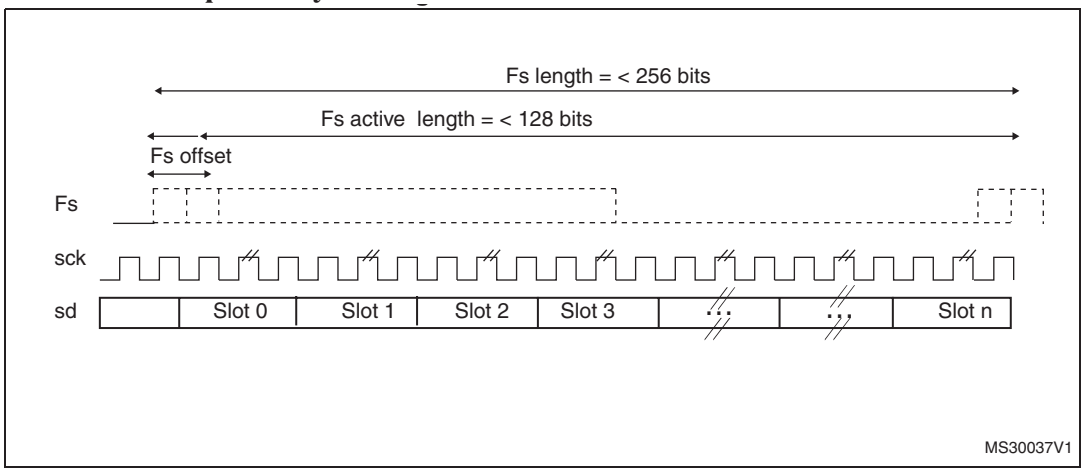
29.6. Размер данных звука

Биты DS[2:0] регистра SAI_xCR1 задают длину данных 8-, 10-, 16-, 20-, 24- или 32 бита. Бит LSBFIRST в SAI_xCR1 говорит какой бит передавать первым (MSB или LSB).

29.7. Синхронизация фрейма

Сигнал синхронизации фрейма FS это Старт Фрейма звука. В регистре SAI_xFRCR можно настроить его форму для самых разных протоколов.

Рис. 284. Фрейм звука



В режиме AC’97 (бит PRTCFG[1:0] = 10 в SAI_xCR1) форма FS задаётся силком и регистр SAI_xFRCR игнорируется.

Настраивать надо оба блока, они слишком независимые.

29.7.1. Длина фрейма

- Ведущий: Биты FRL[7:0] регистра SAI_xFRCR задают длину фрейма до 256 тактов бита. Если длина фрейма больше числа объявленных слотов во фрейме, то остаток битов передаётся нулями или линия SD уходит в HI-z (высокоимпедансное, третье) состояние в зависимости от бита TRIS в регистре SAI_xCR2. При приёме остаток битов игнорируется.
- Ведомый: Длина фрейма в основном говорит ведомому число тактов бита во фрейме, передаваемых ведущим. Служит для обнаружения раннего или позднего появления сигнала синхронизации фрейма во время работающей передачи. Тут уж выдаётся ошибка.

Число битов во фрейме равно FRL[7:0] + 1.

Минимальное число 8. Тут размер данных равен 8 битам и битами NBSLOT[3:0] в регистре SAI_xSLOTR определён только один слот (NBSLOT[3:0] = 0000 для слота 0).

У ведущего:

- Если бит NODIV в регистре SAI_xCR1 снят, то длина фрейма должна быть равна степени 2 от 8 до 256. При этом фрейм содержит целое число импульсов MCLK на такт бита, нужных для DAC/ADC декодеров. Если значение в FRL[7:0] не соответствует этому правилу, то ставится флаг WCKCFG и при разрешении битом WCKCFGIE в регистре SAI_xIM выдаётся прерывание. SAI автоматически отключается.
- Если бит NODIV в регистре SAI_xCR1 стоит, то биты FRL[7:0] могут принимать любое значение, поскольку входные такты блока должны быть равны тактам битов. Такты MCLK_x автоматически выключаются.

У ведомого ограничений на биты FRL[7:0] в регистре SAI_xFRCR нет.

29.7.2. Полярность синхронизации фрейма

Бит FSPOL регистра SAI_xFRCR задаёт активную полярность ножки FS при старте фрейма. Запуск по фронту.

В режиме ведомого блок для начала приёма или передачи ждёт допустимого фрейма, синхронизированного с сигналом Старта фрейма. И если не обнаружена ошибка раннего старта фрейма, начинает работу.

У ведущего синхронизация фрейма постоянно посылается по завершению фрейма, пока не снимется SAIxEN в SAI_xCR1. Если по концу предыдущего фрейма данных в FIFO нет, то возникает исчерпание, но поток не прерывается.

29.7.3. Длина активного уровня синхронизации фрейма

Биты FSALL[6:0] в регистре SAI_xFRCR задают длительность сигнала FS от 1 до 128 тактов бита SCK. Она может быть половину длины фрейма в I2S, LSB или MSB-выравненных режимах, один бит в режимах PCM/DSP или TDM и 16-бит в AC’97.

29.7.4. Смещение синхронизации фрейма

Бит FSOFF регистра SAI_xFRCR позволяет сместить сигнал FS на время передачи первого или последнего бита фрейма.

29.7.5. Роль сигнала FS

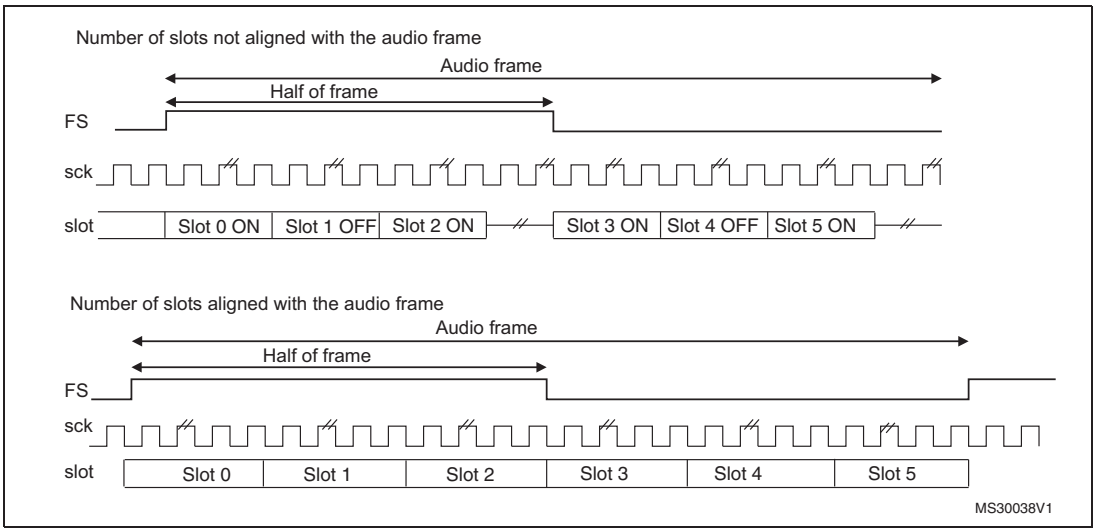
Бит FSDEF в SAI_xFRCR говорит, что сигнал FS может означать:

0: Старт фрейма, как в протоколах PCM/DSP, TDM, AC'97,

1: Старт фрейма и канал внутри фрейма, как в протоколах I2S, MSB или LSB-выравненные.

Если сигнал FS используется как старт фрейма и смена канала, то в числе слотов заявляют сумму для правого и левого канала (пополам). Если число тактов бита в половине фрейма больше числа слотов этого канала, то при TRIS = 0 из регистра SAI_xCR2, в остальных тактах передаётся 0, при TRIS = 1 линия SD уходит HI-Z состояние. При приёме остаток игнорируется вплоть до смены канала.

Рис. 285. Роль FS как старта фрейма и канал (FSDEF = TRIS = 1)



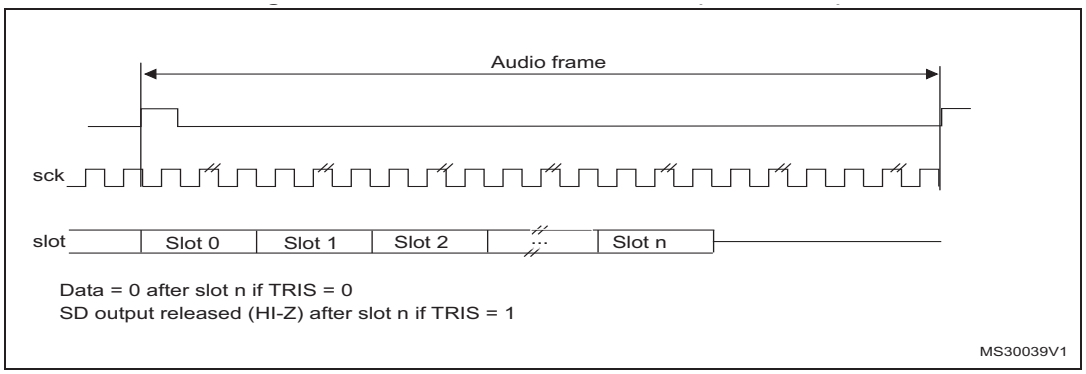
1. Длина фрейма должна быть чётной.

При снятом бите SAI_xFRCR/FSDEF сигнал FS это Старт фрейма и если число слотов из битов SAI_xSLOTR/NBSLOT[3:0] умноженное на длину слота (SAI_xSLOTR/SLOTSZ[1:0]) меньше размера фрейма (SAI_xFRCR/FRL[7:0]), то при передаче:

- При SAI_xCR2/TRIS = 0 остаток битов за последним слотом передаётся нулями,
- При SAI_xCR2/TRIS = 1, линия уходит в HI-Z состояние.

При приёме эти биты отбрасываются.

Рис. 286. Роль FS как старта фрейма (FSDEF = 0)



29.8. Конфигурация слота

Слот это базовый элемент фрейма. Число слотов равно SAI_xSLOTR/NBSLOT[3:0]+1. До 16.

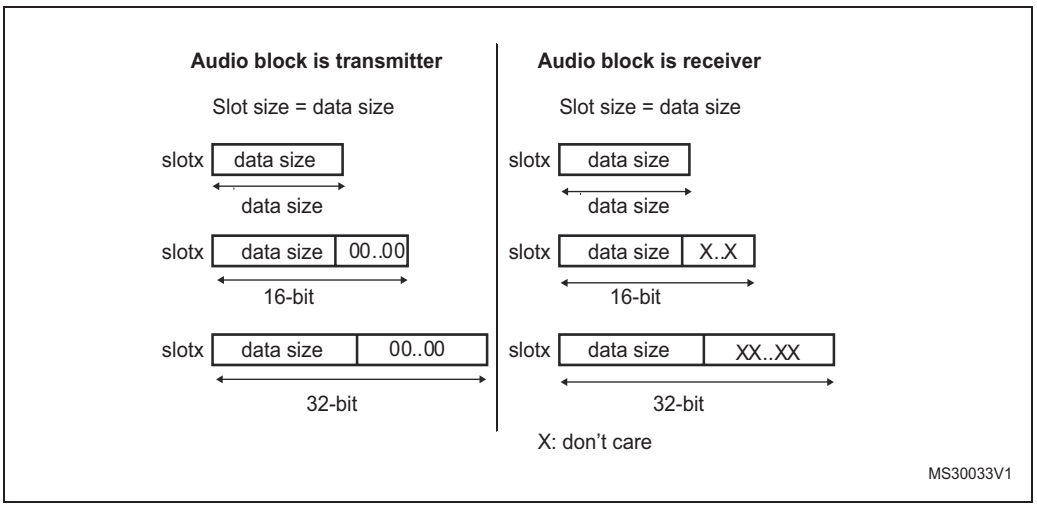
В протоколе AC'97 (бит PRTCFG[1:0] = 10) число слотов фиксировано и NBSLOT[3:0] не нужно.

Слот определяется действительным в битах SAI_xSLOTR/SLOTEN[15:0]. При передаче плохого слота на линию подаются нули или она уходит в HI-Z.

При приёме данные от конца такого слота игнорируются и к FIFO не суются.

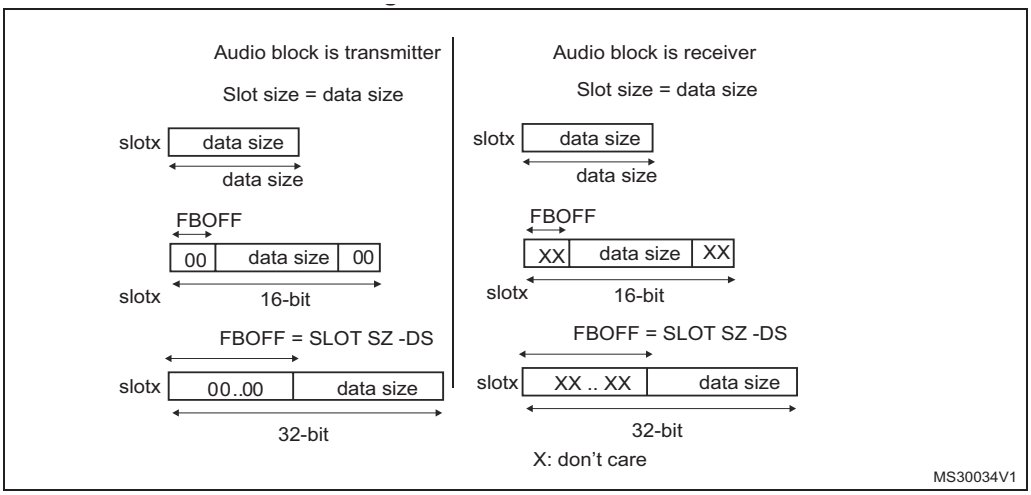
Размер слота пишут в SAI_xSLOTR/SLOTSZ[1:0]. Он един для всего фрейма.

Рис. 287. Размер слота с FBOFF = 0 в SAI_xSLOTR



Биты SAI_xSLOTR/FBOFF[5:0] задают смещение первого бита данных внутри слота. В начало вставляются нули, при приёме эти биты теряются. Это надо для LSB выравненного протокола.

Рис. 288. Смещение первого бита



Во избежание дурного поведения SAI надо соблюдать:

FBOFF ≤(SLOTSZ - DS),
DS ≤SLOTSZ,
NBSLOT x SLOTSZ ≤FRL (frame length),

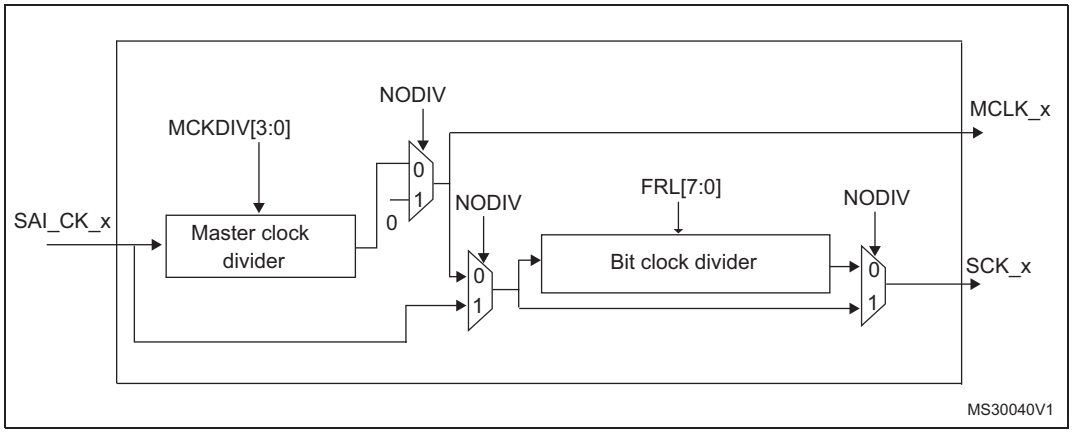
При стоящем бите SAI_xFRCR/FSDEF число слотов должно быть чётным. В режиме AC'97 размер слота задаётся автоматически.

29.9. Тактовый генератор SAI

У каждого блока есть свой тактовый генератор. Они совсем одинаковые.

Генератор ведущего выдаёт такты битов для связи и ведущие такты для внешнего декодера. Генератор ведомого выключается.

Рис. 289. Генератор звукового блока



NB: При стоящем NODIV сигнал MCLK_x будет низким.

Такты на генератор идут от системы. Из тактов SAI_CK_x после деления на MCKDIV[3:0] получают ведущие такты для внешних декодеров:

MCLK_x = SAI_CK_x / (MCKDIV[3:0] * 2), если MCKDIV[3:0] не равно 0000.

MCLK_x = SAI_CK_x, если MCKDIV[3:0] равно 0000.

Сигнал MCLK_x используется только в TDM.

Деление должно быть чётным чтобы получить скважность 50% для MCLK и SCK_x. При MCKDIV[3:0] = 0000, MCLK_x = SAI_CK_x.

В SAI принято отношение MCLK/FS = 256. В большинстве случаев есть три диапазона.

Таблица 130. Пример возможных частот выборки

Входная частота SAI_CK_x	Обычная частота выборки	MCKDIV[3:0]
192 kHz x 256	192 kHz	MCKDIV[3:0] = 0000
	96 kHz	MCKDIV[3:0] = 0001
	48 kHz	MCKDIV[3:0] = 0010
	16 kHz	MCKDIV[3:0] = 0100
	8 kHz	MCKDIV[3:0] = 1000
44.1 kHz x 256	44.1 kHz	MCKDIV[3:0] = 0000
	22.05 kHz	MCKDIV[3:0] = 0001
	11.025 kHz	MCKDIV[3:0] = 0010
SAI_CK_x = MCLK(1)	MCLK	MCKDIV[3:0] = 0000

1. Бывает, что вместо PLL используют внешний источник.

Если блок объявлен ведущим с битом SAI_xCR1/NODIV = 0, то для ведущих тактов внешних декодеров можно выбрать сигнал из другого места устройства.

Такты бита получают из ведущих тактов:

SCK_x = MCLK x (FRL[7:0] +1) / 256

где:

256 это стандартное соотношение между MCLK и частотой выборки.

SAI_xFRCR/FRL[7:0] это число битов в фрейме - 1.

Для ведущего число (FRL[7:0] +1) должно быть степенью 2 чтобы получить чётное число импульсов MCLK_x на такт бита. На нём гарантируется 50% скважность SCK_x.

Такты SAI_CK_x тоже могут быть равны частоте тактов бита.

В этом случае ставят бит SAI_xCR1/NODIV и значения делителей MCKDIV и тактов бита игнорируются и число битов на фрейм не должно быть степенью 2.

Стробирующий фронт SCK задают битом SAI_xCR1/CKSTR.

29.10. Внутренние FIFO

У каждого блока SAI есть один свой FIFO и для передачи и для приёма, так что на бит SAI_xSR/FREQ поступает только один запрос. Прерывание выдаётся при SAI_xIM/FREQIE =1. Оно зависит от:

- Установок порога FIFO (биты SAI_CR2/FLTH)
- Направления связи (приём или передача)

Прерывания при передаче

- При настройке порога FIFO на пустоту (SAI_xCR2/FTH[2:0] = 000b)
  запрос прерывания (SAI_xSR/FREQ=1) выдаётся при отсутствии данных в регистре SAI_xDR (SAI_xSR/FLTH[2:0] < 001b).
  Снимается аппаратно при появлении данных в FIFO (SAI_xSR/FLTH[2:0] != 000b).
- При настройке порога FIFO на четверть объёма (SAI_xCR2/FTH[2:0] = 001b)
  запрос прерывания (SAI_xSR/FREQ=1) выдаётся когда менее четверти FIFO содержит данные (SAI_xSR/FLTH[2:0] < 010b).
  Снимается аппаратно при заполнении не менее четверти FIFO (SAI_xSR/FLTH[2:0] >= 010b).
- При настройке порога FIFO на половину объёма (SAI_xCR2/FTH[2:0] = 010b)
  запрос прерывания (SAI_xSR/FREQ=1) выдаётся когда менее половины FIFO содержит данные

(SAI_xSR/FLTH[2:0] < 011b). Снимается аппаратно при заполнении не менее половины FIFO (SAI_xSR/FLTH[2:0] >= 011b).

- При настройке порога FIFO на три четверти объёма (SAI_xCR2/FTH[2:0] = 011b) запрос прерывания (SAI_xSR/FREQ=1) выдаётся когда менее трёх четвертей FIFO содержит данные (SAI_xSR/FLTH[2:0] < 100b).
  Снимается аппаратно при заполнении не менее трёх четвертей FIFO (SAI_xSR/FLTH[2:0] >= 100b).
- При настройке порога FIFO на весь объём (SAI_xCR2/FTH[2:0] = 100b) запрос прерывания (SAI_xSR/FREQ=1) выдаётся когда не весь FIFO содержит данные (SAI_xSR/FLTH[2:0] < 101b).
  Снимается аппаратно при заполнении всего объёма FIFO (SAI_xSR/FLTH[2:0] = 101b).

Прерывания при приёме

- При настройке порога FIFO на пустоту (SAI_xCR2/FTH[2:0] = 000b) запрос прерывания (SAI_xSR/FREQ=1) выдаётся при наличии данных в регистре SAI_xDR (SAI_xSR/FLTH[2:0] >= 001b).
  Снимается аппаратно при опустении FIFO (SAI_xSR/FLTH[2:0] = 000b)
- При настройке порога FIFO на четверть объёма (SAI_xCR2/FTH[2:0] = 001b) запрос прерывания (SAI_xSR/FREQ=1) выдаётся когда для записи доступно менее четверти FIFO (SAI_xSR/FLTH[2:0]< 010b).
  Снимается аппаратно при заполнении не менее четверти FIFO (SAI_xSR/FLTH[2:0] >= 010b).
- При настройке порога FIFO на половину объёма (SAI_xCR2/FTH[2:0] = 010b) запрос прерывания (SAI_xSR/FREQ=1) выдаётся когда для записи доступна хоть половина FIFO (SAI_xSR/FLTH[2:0] >= 011b). Снимается аппаратно при доступности не менее половины FIFO (SAI_xSR/FLTH[2:0] < 011b).
- При настройке порога FIFO на три четверти объёма (SAI_xCR2/FTH[2:0] = 011b) запрос прерывания (SAI_xSR/FREQ=1) выдаётся когда доступно не менее трёх четвертей FIFO (SAI_xSR/FLTH[2:0] >= 100b).
  Снимается аппаратно при доступности менее трёх четвертей FIFO (SAI_xSR/FLTH[2:0] < 100b).
- При настройке порога FIFO на весь объём (SAI_xCR2/FTH[2:0] = 100b) запрос прерывания (SAI_xSR/FREQ=1) выдаётся когда весь FIFO содержит данные (SAI_xSR/FLTH[2:0] = 101b).
  Снимается аппаратно при появлении места в FIFO (SAI_xSR/FLTH[2:0] < 101b).

Подобно выдаче прерываний, SAI может использовать (ставить бит SAI_xCR1/DMAEN). Механизм установки бита FREQ такой же, как и вышеописанный для FREQIE.

Все FIFO содержат по 8 слов. Одно обращение к FIFO затрагивает одно слово, независимо от размера доступа. Каждое слово FIFO содержит один звуковой фрейм. Указатели FIFO инкрементируются при каждом обращении к регистру SAI_xDR.

При записи в SAI_xDR данные должны быть выравнены вправо.

При чтении из SAI_xDR они именно такие.

Инициализируют указатели FIFO у выключенного SAI установкой бита SAI_xCR2/FFLUSH. Если FFLUSH ставить у включённого SAI, то данные в FIFO автоматически теряются.

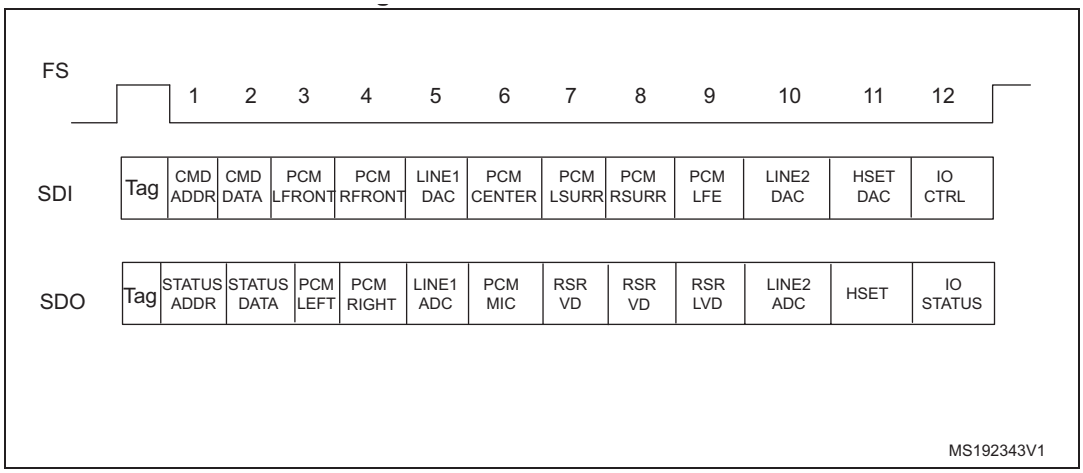
29.11. Контроллер связи AC'97

Протокол:

- Номер и размер слота фиксированы.
- Форма сигнала FS фиксирован.
- Выбирается записью SAI_xCR1/PRTCFG[1:0]=10. Данные могут быть только 16 и 20 бит.
- Биты NBSLOT[3:0] и SLOTSZ[1:0] игнорируются.
- Слотов всегда 13. Первый из 16 битов, остальные по 20 (слоты данных).
- Биты SAI_xSLOTR/FBOFF[5:0] игнорируются.
- Регистр SAI_xFRCR игнорируется.

Сигнал FS всегда выходной, независимо от конфигурации ведущий/ведомый/.

Рис. 290. Звуковой фрейм AC'97



NB: В протоколе AC'97 бит 2 of зарезервирован (всегда передаётся 0, независимо от содержимого SAI FIFO.

Один SAI можно использовать для двухточечной связи AC'97.

В режиме приёма SAI, действуя как контроллер связи AC'97 потребует отсутствия запросов FIFO, так что при низком бите готовности кодека в слоте 0 данные в FIFO не пишутся. Бит SAI_xIM/CNRDYIE разрешает прерывания по флагу SAI_xSR/CNRDY. Это для протокола AC'97.

29.12. Специальные режимы

Они доступны через регистр SAI_xCR2.

29.12.1.Немой режим

Передатчик

Включать можно в любое время, немой режим действует на фрейм целиком. Установка бита SAI_xCR2/MUTE во время передачи фрейма запрашивает немой режим.

Бит проверяется только в конце фрейма и, если он стоит, то немой режим включается в начале грядущего фрейма. И так постоянно.

Если число слотов в SAI_xSLOTR/NBSLOT[3:0]<=2, то битом SAI_xCR2/MUTEVAL можно указать передачу в немом режиме нулей или остатка каждого слота.

Если число слотов в SAI_xSLOTR/NBSLOT[3:0]>2, то SAI_xCR2/MUTEVAL бестолков, всё равно в каждом бите каждого слота передаётся 0.

В немом режиме указатели FIFO инкрементируются, то есть данные с немым режимом отбрасываются.

Приёмник

Немой режим можно узнать по тому, что во всех объявленных и допустимых слотах фрейма приняты 0 для заданного числа последовательных фреймов (биты SAI_xCR2/MUTECNT[5:0]).

При обнаружении немых фреймов встаёт флаг SAI_xSR/MUTEDET и запрашивается прерывание. Его разрешают битом SAI_xCR2/MUTEDETIE.

Счётчик немых фреймов чистится при выключении блока или появлении в фрейме хоть одного допустимого данного. Прерывание выдаётся только один раз, во время перехода счётчика пустых фреймов в значение MUTECNT[5:0]. При обнулении счётчика всё повторяется.

29.12.2.Моно/Стерео

Режим Моно включается битом SAI_xCR1/MONO при числе слотов равном 2 (SAI_xSLOTR/NBSLOT[3:0] = 0001).

Тогда при передаче число обращений к FIFO уполовинивается и данные слота 0 дублируются в слоте 1.

При приёме в FIFO пишутся только данные слота 0.

Если поток данных это реальный стерео-сигнал, то бит MONO не работает. Преобразование стерео в моно делают программно.

NB: Для включения Моно биты NBSLOT и SLOTEN должны быть равны 2 и бит MONO = 1.

29.12.3.Сжатие/разжатие ("компандинг")

В режиме TDM битами SAI_xCR2/COMP[1:0] можно устроить аппаратное сжатие передаваемого по линии сигнала и его расширение на приёме.

Поддерживаются два стандарта: μ-Law из США и Японии A-Law из Европы.

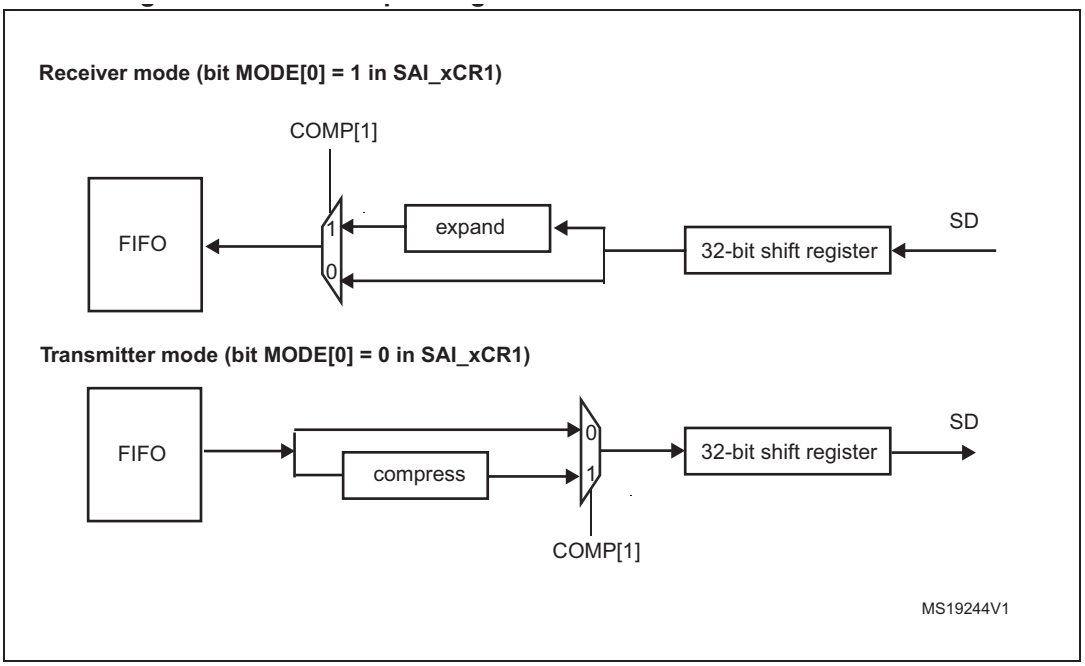
μ-Law имеет динамический диапазон 14 битов (SAI_xCR2/COMP[1:0] = 10).

A-Law имеет динамический диапазон 13 битов (SAI_xCR2/COMP[1:0] = 11).

Стандарт (μ-Law или A-Law) можно вычислить на базе дополнения 1 или 2, в зависимости от бита SAI_xCR2/CPL.

Форматы μ-Law и A-Law кодируют данные в 8-бит элементы с левым выравниваем. Всегда 8 бит. Так что при выборе в COMP[1:0] одного из режимов у включённого блока аппаратно ставится SAI_xCR1/DS[2:0]=010.

Рис. 291. Аппаратный "компандинг"



NB: В режиме AC'97 не применимо.

29.12.4. Управление линией данных для неактивного слота

При передаче установка бита SAI_xCR2/TRIS у выключенного SAI заставляет:

- У неактивных слотов по линии SD передавать 0 или
- Отпускать линию в HI-Z состояние до конца передачи последнего бита на благо других передатчиков на этой линии.

Для обеспечения зазора между передачами малые данные можно расширять до 32 бит установкой SAI_xSLOTR/SLOTSZ[1:0]=10. Тогда, если следующий слот объявлен неактивным, то после передачи LSB на время расширения данных линия SD уйдёт в HI-Z.

Также в HI-Z уходят если произведение числа слотов на длину слота меньше длины фрейма и есть расширение нулями.

Если выбранный протокол использует сигнал FS как старт фрейма и идентификатор канала (бит SAI_xFRCR/FSDEF = 1), то HI-Z работает согласно рис. 293 (где бит SAI_xCR1/TRIS = 1, FSDEF=1, половина длины фрейма > числа слотов/2 и NBSLOT=6).

Рис. 292. HI-Z стратегия на выходной линии SD для неактивного слота

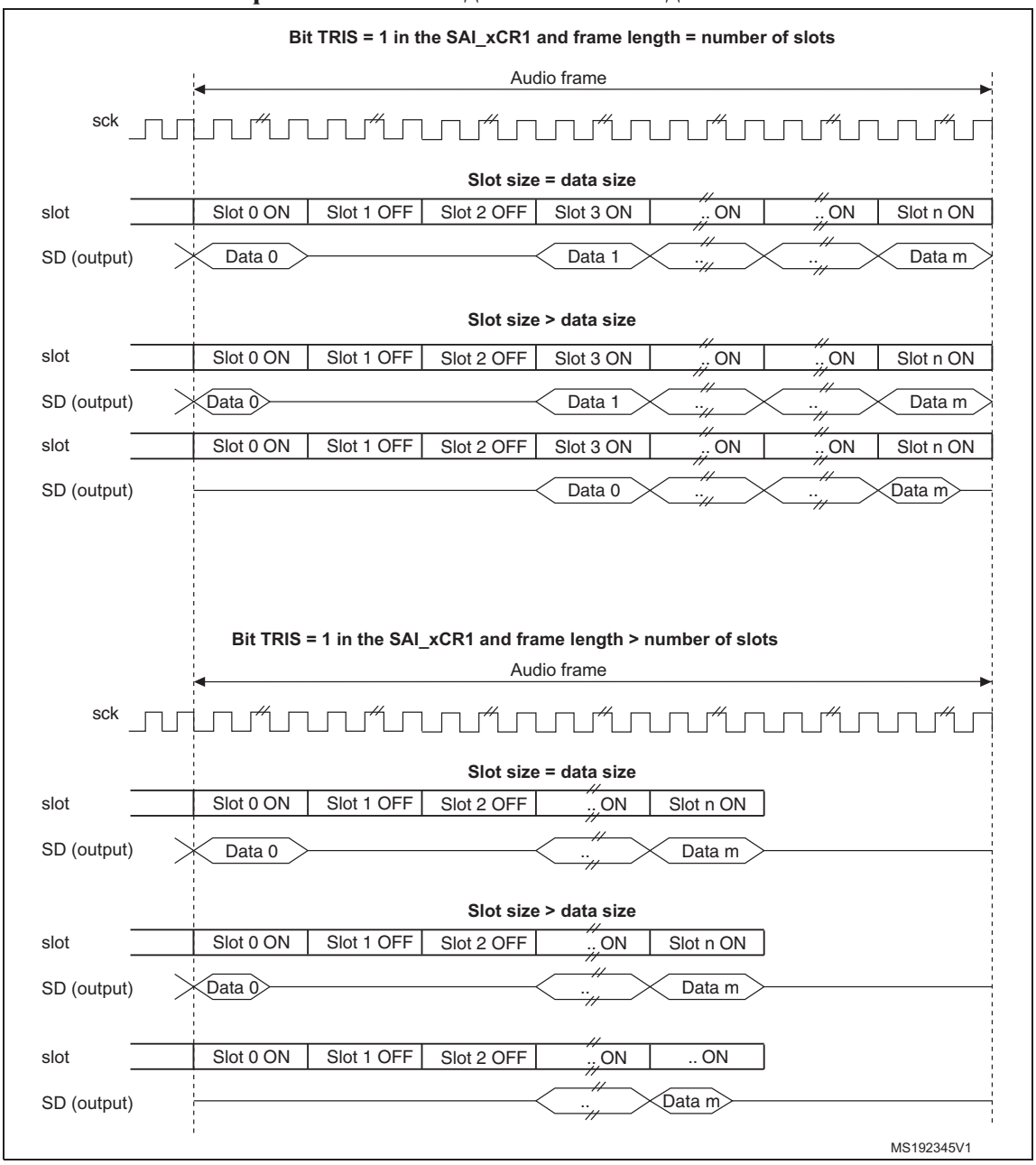
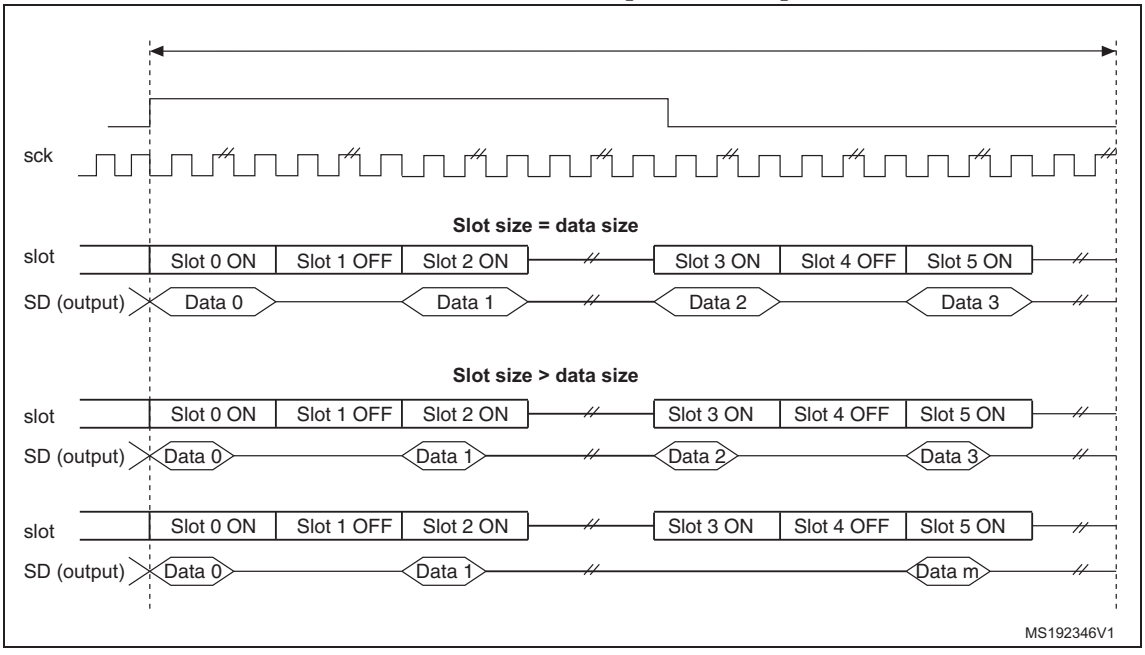


Рис. 293. HI-Z на выходной линии SD в протоколе вроде I2S



29.13. Флаги ошибок

Имеем:

- Переполнение/исчерпание FIFO,
- Ранний сигнал FS,
- Поздний сигнал FS,
- Кодек не готов (исключительно AC’97),
- Ошибка конфигурации тактов ведущего.

29.13.1. Переполнение/исчерпание FIFO (OVRUDR)

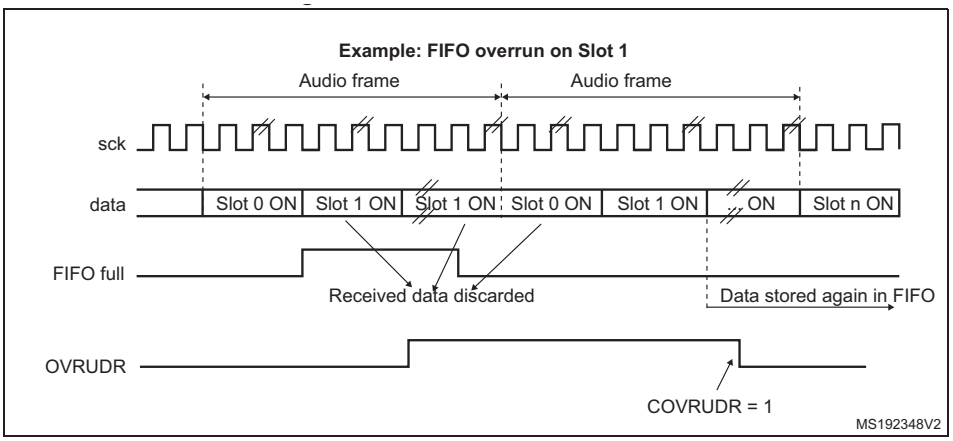
Флаг ошибки именуется SAI_xSR/OVRUDR потому что у каждого блока свой регистр SAI_xSR.

Переполнение

У приёмника он ставится когда FIFO более не способно принимать данные с линии. Прерывание разрешают битом SAI_xSR/OVRUDR. Принятое данное теряется. Номер слота запоминается внутри. При появлении свободного места в FIFO данные принимаются с запомненного номера слота в новом фрейме (выравнивание слотов не нарушается.

Флаг OVRUDR снимают установкой бита SAI_xCLRFR/COVRUDR.

Рис. 294. Переполнение



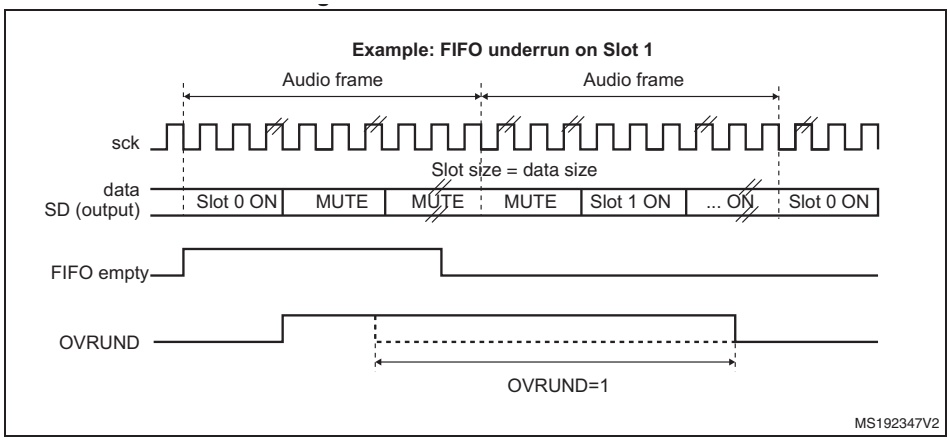
Исчерпание

Возникает когда в FIFO нет данных для запущенной передачи. Программа должна заново синхронизировать данные и слот:

1. Выключить SAI (снять SAI_xCR1/SAIEN) и проверить это.
2. Слить Tx FIFO битом SAI_xCR2/FFLUS.
3. Назначить передачу с первого активного слота нового фрейма.
4. Включить SAI (SAIEN=1).

Вставший флаг снимают установкой бита SAI_xCLRFR/COVRUDR.

Рис. 295. Исчерпание FIFO



29.13.2. Ранняя синхронизация фрейма (AFSDET)

У ведомого в регистре SAI_xFRCR известны длина, полярность и смещение фрейма. Появление сигнала FS раньше нужного времени ставит флаг ошибки SAI_xSR/AFSDET. На текущий фрейм это не влияет.

Бит SAI_xIM/AFSDETIE разрешает прерывание. Снимают флаг AFSDET установкой бита SAI_xCLRFR/CAFSDET.

После возникновения ошибки надо повторно синхронизироваться с ведущим:

1. Выключаем блок SAI снятием бита SAI_xCR1/SAIEN и проверяем сей факт.
2. Битом SAI_xCR2/FFLUS сливаем FIFO.
3. Включаем блок SAI (SAIEN = 1), затем SAI.
4. Блок SAI ждёт появления сигнала FS.

NB: В режиме AC'97 флаг не встаёт, FS выдаётся даже при ведомом режиме.

29.13.3. Поздняя синхронизация фрейма

Появление сигнала FS позже нужного времени ставит у ведомого флаг ошибки SAI_xSR/LFSDET.

Бит SAI_xIM/LFSDETIE разрешает прерывание. Снимают флаг LFSDET установкой бита SAI_xCLRFR/CLFSDET.

После возникновения ошибки надо повторно синхронизироваться с ведущим (см. выше).

Причиной могут стать глитчи на тактах SCK, сбивающие машину состояний блока, что приведёт к разрушению фрейма.

При работе в прерывном режиме фрейм не разрушается, но флаг LFSDET встаёт

NB: В режиме AC'97 флаг не встаёт, FS выдаётся даже при ведомом режиме.

29.13.4. Кодек не готов (CNRDY AC'97)

Флаг SAI_xSR/CNRDY работает только в режиме AC'97 (бит SAI_xCR1/PRTCFG[1:0] = 10). Прерывание разрешают битом SAI_xIM/CNRDYIE. Снимают флаг CNRDY установкой бита SAI_xCLRFR/CCNRDY.

Ставится при неготовности кодека принять TAG 0 (слот 0) фрейма AC'97. Запись в FIFO возобновится и определённые в регистре SAI_xSLOTR активные слоты будут приняты после восстановления готовности кодека.

29.13.5. Дурная конфигурация тактов у ведущего (с NODIV=0)

Флаг SAI_xSR/WCKCFG встаёт если при включении (SAI_xCR1/SAIxEN) ведущего (SAI_xCR1/MODE[1]=0) с битом SAI_xCR1/NODIV =0 биты SAI_xFRCR/FRL[7:0] не соответствуют формуле:

$$(FRL[7,0]) + 1 = 2^n$$

где n - от 3 до 8

Прерывание разрешают битом SAI_xIM/WCKCFGIE. Снимают флаг WCKCFG установкой бита SAI_xCLRFR/CWCKCFG.

Блок автоматически выключается.

29.14. Источники прерываний

Таблица 131. Источники прерываний

Источник	Группа	Режим блока	Разрешение	Очистка
FREQ	FREQ	Любой	SAI_xIM/FREQIE	См. описание FIFO
OVRUDR	ERROR	Любой	SAI_xIM/OVRUDRIE	SAI_xCLRFR/COVRUDR = 1
AFSDET	ERROR	Ведомый, кроме AC'97	SAI_xIM/AFSDETIE	CAFSDET = 1 в SAI_xCLRFR
LFSDET	ERROR	Ведомый, кроме AC'97	SAI_xIM/LFSDETIE	CLFSDET = 1 в SAI_xCLRFR
CNRDY	ERROR	Ведомый, только AC'97	SAI_xIM/CNRDYIE	CCNRDY = 1 в SAI_xCLRFR
MUTEDET	MUTE	Только приём	SAI_xIM/MUTEDETIE	CMUTEDET = 1 в SAI_xCLRFR
WCKCFG	ERROR	Ведущий с NODIV = 0	SAI_xIM/WCKCFGIE	CWCKCFG = 1 in SAI_xCLRFR register

После появления ошибки надо выключить SAI, заново настроить и включить его.

29.15. Выключение SAI

Блок SAI можно выключать когда угодно снятием бита SAI_xCR1/SAIxEN. Все запущенные фреймы будут переданы до конца и Бит будет стоять до конца передачи запущенных фреймов.

У синхронных блоков первым надо вырубать ведущего.

29.16. Интерфейс с DMA

У каждого блока SAI есть свой канал DMA. Включается он установкой бита SAI_xCR1/DMAEN. Направление передачи определяется конфигурацией блока SAI.

Настроим SAI с DMA:

1. Настроим SAI и уровень порога FIFO для выдачи запросов DMA
2. Ставим канал SAI DMA
3. Включим DMA
4. Включим SAI

NB: При настройке блока SAI канал SAI DMA должен быть выключен.

29.17. Регистры SAI

29.17.1.Регистр конфигурации 1 (SAI_xCR1) где x = A или B

Смещение адреса: Блок A: 0x004

Смещение адреса: Блок B: 0x024

По сбросу: 0x0000 0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								MCKDIV[3:0]				NODIV	Res.	DMAEN	SAIxEN
								rw	rw	rw	rw	rw		rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		OutDriv	MONO	SYNCEN[1:0]		CKSTR	LSBFIRST	DS[2:0]			Res.	PRTCFG[1:0]		MODE[1:0]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw

— Биты 31:24 Резерв, не трогать.

— Биты 23:20 **MCKDIV[3:0]**: Делитель тактов ведущего.

Пишутся программно при выключенном блоке. У ведомого не работают.

0000: делитель равен 1. Иначе по формуле: MCLK_x = SAI_CK_x / (MCKDIV[3:0] * 2)

— Бит 19 **NODIV**: Без делителя.

Пишется программно.

0: Делитель используется

1: Без делителя

— Бит 18 Резерв, не трогать.

— Бит 17 **DMAEN**: Разрешение DMA.

Пишется программно.

0: Нельзя

1: Нужно

NB: У приёмника биты MODE надо писать до установки DMAEN

— Бит 16 **SAIxEN**: Разрешение блока A или B.

Ставят программно, снимается аппаратно после записи 0 и завершения передач.

0: Выкл.

1: Вкл. Ставить можно только уже снятый бит.

NB: Ведущий SAIx можно включать только при наличии тактов на входе.

— Биты 15:14 Резерв, не трогать.

— Бит 13 **OUTDRIV**: Включение выхода..

Пишется программно.

0: Выход включается при установке SAIEN

1: Выход включается сразу после установки этого бита.

NB: Надо ставить до включения блока, но после его конфигурации.

— Бит 12 **MONO**: Режим МОНО.

Пишется программно.

0: Стерео

1: Моно.

Имеет значение только при числе слотов, равном 2.

При передаче в режиме Моно данные слота 0 дублируются в слот 1. При приёме данные слота 1 выбрасываются.

— Биты 11:10 **SYNCEN[1:0]**: Разрешение синхронизации.

Пишутся программно при выключенном блоке.

00: Асинхронный блок.

01: Блок синхронный с другим внутренним. Должен быть ведомым

10: Резерв.

11: Не используется

— Бит 9 **CKSTR**: Фронт такта стробирования данных.

Пишется программно при выключенном блоке.

0: Задний

1: Передний

— Бит 8 **LSBFIRST**: Младший бит передаётся первым (не для AC’97).

Пишется программно при выключенном блоке.

0: Первым MSB

1: Первым LSB

— Биты 7:5 **DS[2:0]**: Размер данных.

Пишется программно при выключенном блоке.

000: Не используется

001: Не используется

010: 8-бит

011: 10-бит

100: 16-бит

101: 20-бит

110: 24-бит

111: 32-бит

В режиме сжатия/расжатия биты DS[1:0] не нужны, данные всегда 8 бит.

NB: В AC’97 разрешены только 16 и 20 бит, иначе SAI спятит.

— Бит 4 Резерв, не трогать.

— Биты 3:2 **PRTCFG[1:0]**: Конфигурация протокола.

Пишется программно при выключенном блоке.

00: Свободный протокол

01: Не используется

10: Протокол AC’97

11: Не используется

Свободный протокол настраивается на любой (типа I2S, LSB/MSB выравнивание, TDM, PCM/DSP...) установкой регистров.

— Биты 1:0 **MODE[1:0]**: Режим блока.

Пишется программно при выключенном блоке.

00: Ведущий передатчик

01: Ведущий приёмник

10: Ведомый передатчик

11: Ведомый приёмник

NB: Ведущий передатчик сразу начинает выдавать FS и такты.

29.17.2.Регистр конфигурации 2 (SAI_xCR2) где x = A или B

Смещение адреса: Блок A: 0x008

Смещение адреса: Блок B: 0x028

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP[1:0]		CPL	MUTECNT[5:0]						MUTE VAL	Mute	TRIS	FFLUS	FTH		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:16 Резерв, не трогать.

— Биты 15:14 **COMP[1:0]**: Режим сжатия (передача)/расжатия (приём).

Пишется программно.

00: Без так сказать "компандирования"

01: Резерв.

10: Алгоритм μ-Law

11: Алгоритм A-Law

NB: Применимо только для режима TDM.

— Бит 13 **CPL**: Тип комплемента.

Пишется программно.

0: 1’s complement.

1: 2’s complement.

NB: Бит работает только для алгоритмов μ-Law и A-Law.

— Биты 12:7 **MUTECNT[5:0]**: Счётчик немого режима.

Пишется программно.

Только для режима приёма.

При совпадении этого числа с числом последовательных немых фреймов встаёт флаг MUTEDET.

— Бит 6 **MUTEVAL**: Немое значение.

Пишется программно при выключенном блоке.

0: В режиме MUTE посылается 0.

1: В режиме MUTE посылается последнее значение из предыдущего фрейма.

Имеет смысл только у передатчика при числе слотов <=2 и стоящем бите MUTE.

При большем числе слотов в немом режиме всегда посылается 0.

— Бит 5 **MUTE**: Младший бит передаётся первым (не для AC’97).

Пишется программно при выключенном блоке.

0: Первым MSB

1: Первым LSB

— Бит 4 **TRIS**: Управление состоянием HI-Z линии данных.

Пишется программно при выключенном блоке.

0: При неактивном слоте линия SD управляется.

1: Линия SD идёт в HI-Z с конца активного слота, если следующий неактивен.

Имеет смысл только у передатчика.

— Бит 3 **FFLUS**: Слив FIFO.

Пишется программно при выключенном блоке и потоке/прерываниях DMA . Читается всегда нулём.

0: Ничего.

1: Слить FIFO.

— Биты 2:0 **FTH[2:0]**: Порог FIFO.

Пишется программно.

000: FIFO пуст

001: 1⁄4 FIFO

010: 1⁄2 FIFO

011: 3⁄4 FIFO

100: FIFO полон

101: Резерв

110: Резерв

111: Резерв

29.17.3.Регистр конфигурации фрейма (SAI_xFRCR) где x = A или B

Смещение адреса: Блок A: 0x00C

Смещение адреса: Блок B: 0x02C

По сбросу: 0x0000 0007

NB: Не имеет смысла для режима AC'97

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													FSOFF	FSPOL	FSDEF
													rw	rw	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	FSALL[6:0]							FRL[7:0]							
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:19 Резерв, не трогать.

— Бит 18 **FSOFF**: Смещение синхонизации фрейма.

Пишется программно при выключенном блоке.

0: FS появляется на первом бите слота 0.

1: FS появляется за один бит до первого бита слота 0.

— Бит 17 **FSPOL**: Активный уровень сигнала FS.

Пишется программно при выключенном блоке.

0: Низкий (задний фронт)

1: Высокий (передний фронт)

— Бит 16 **FSDEF**: Определение роли сигнала FS.

Пишется программно при выключенном блоке.

0: Старт фрейма

1: Старт фрейма + сторона канала

При стоящем бите число слотов в SAI_ASLOTR должно быть чётным, половина - левый, половина - правый канал (надо ставить для протоколов I2S или MSB/LSB-выравненных ...)

— Бит 15 Резерв, не трогать.

— Биты 14:8 **FSALL[6:0]**: Активная длина сигнала FS.

Пишется программно при выключенном блоке.

Это число тактов бита (SCK) + 1 (FSALL[6:0] + 1) активного уровня FS.

— Биты 7:0 **FRL[7:0]**: Длина фрейма.

Пишется программно при выключенном блоке.

Это число тактов SCK на фрейм. Оно равно FRL[7:0] + 1. Минимальное число передаваемых битов равно 8, или блок сбрендит. Это случай с одним слотом 0 и 8 бит длины.

При выводе ведущих тактов на ножку MCLK_x в режиме ведущего длина фрейма должна быть степенью 2 от 8 до 256, для пользы внешних DAC/ADC в декодерах.

Длина фрейма должна быть чётной.

29.17.4.Регистр слотов (SAI_xSLOTR) где x = A или B

Смещение адреса: Блок A: 0x010

Смещение адреса: Блок B: 0x030

По сбросу: 0x0000 0000

NB: Не имеет смысла для режима AC'97

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SLOTEN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				NBSLOT[3:0]				SLOTSZ[1:0]		Res	FBOFF[4:0]				
				rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

— Биты 31:16 **SLOTEN[15:0]**: Разрешение слота.
Пишется программно при выключенном блоке.
Каждый бит соответствует позиции слота от 0 до 15 (максимум 16 слотов)
0: Неактивный.
1: Активный.
— Биты 15:12 Резерв, не трогать.
— Биты 11:8 **NBSLOT[3:0]**: Число слотов во фрейме, включая неактивные, максимум 16.
Пишется программно при выключенном блоке.
При стоящем бите SAI_AFRCR/FSDEF должно быть чётным. Если размер больше размера данных , то остаток битов передаётся в соответствии с битом SAI_xCR1/TRIS
— Биты 7:6 **SLOTSZ[1:0]**: Размер слота.
Пишется программно при выключенном блоке.
00: Равен размеру данных из SAI_ACR1/DS[3:0].
01: 16-бит
10: 32-бит
11: Резерв
Размер слота должен быть больше или равен размеру данных, иначе SAI психанёт.
— Бит 5 Резерв, не трогать.
— Биты 4:0 **FBOFF[4:0]**: Смещение первого бита в слоте.
Пишется программно при выключенном блоке.
До этого бита передаются нули, а на приёме они выбрасываются.

29.17.5.Регистр маски прерываний (SAI_xIM) где x = A или B

Смещение адреса: Блок A: 0x014
Смещение адреса: Блок B: 0x034
По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									LFSDETIE	AFSDETIE	CNRDYIE	FREQIE	WCKCFGIE	MUTEDETIE	OVRUDRIE
									rw	rw	rw	rw	rw	rw	rw

Всё пишется программно.
0: Запрещено
1: Разрешено
— Биты 31:7 Резерв, не трогать.
— Бит 6 **LFSDETIE**: Разрешение прерывания по флагу LFSDET.
Бит не имеет смысла в режиме AC’97 и у ведущего блока.
— Бит 5 **AFSDETIE**: Разрешение прерывания по флагу AFSDET.
Бит не имеет смысла в режиме AC’97 и у ведущего блока.
— Бит 4 **CNRDYIE**: Разрешение прерываний по флагу CNRDY.
Бит имеет смысл только в режиме AC’97 и у ведомого блока блока.
— Бит 3 **FREQIE**: Разрешение прерываний по флагу FREQ.
У приёмника биты MODE надо ставить до бита FREQIE.
— Бит 2 **WCKCFGIE**: Разрешение прерываний по флагу WCKCFG.
NB: Работает только в режиме TDM.
— Бит 1 **MUTEDETIE**: Разрешение прерываний по флагу MUTEDET.
Имеет смысл только у приёмника.
— Бит 0 **OVRUDRIE**: Разрешение прерываний по флагу OVRUDR.

29.17.6.Регистр состояния (SAI_xSR) где x = A или B

Смещение адреса: Блок A: 0x018
Смещение адреса: Блок B: 0x038

По сбросу: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													FLTH		
													r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									LFSDET	AFSDET	CNRDY	FREQ	WCKCFG	MUTEDET	OVRUDR
									r	r	r	r	r	r	r

— Биты 31:19 Резерв, не трогать.

— Биты 18:16 **FLTH**: Порог FIFO.

Правится аппаратно в зависимости от режима SAI.

Передатчик:

000: FIFO пуст

001: FIFO <= 1⁄4 но не пуст

010: 1⁄4 < FIFO <= 1⁄2

011: 1⁄2 < FIFO <= 3⁄4

100: 3⁄4 < FIFO но не полон

101: FIFO полон

Приёмник:

000: FIFO пуст

001: FIFO < 1⁄4 но не пуст

010: 1⁄4 <= FIFO < 1⁄2

011: 1⁄2 =< FIFO < 3⁄4

100: 3⁄4 =< FIFO но не полон

101: FIFO полон

— Биты 15:7 Резерв, не трогать.

— Бит 6 **LFSDET**: Флаг ошибки поздней синхронизации.

Прерывание разрешают битом SAI_xIM/LFSDETIE.

Снимают установкой бита SAI_xCLRFR/CLFSDET.

Бит не имеет смысла в режиме AC’97.

— Бит 5 **AFSDET**: Флаг ошибки ранней синхронизации.

Прерывание разрешают битом SAI_xIM/AFSDETIE.

Снимают установкой бита SAI_xCLRFR/CAFSDET.

Бит не имеет смысла в режиме AC’97.

— Бит 4 **CNRDY**: Внешний кодек AC’97 не готов.

Прерывание разрешают битом SAI_xIM/CNRDYIE.

Снимают установкой бита SAI_xCLRFR/CCNRDY.

Бит имеет смысл только в режиме AC’97 и у ведомого блока блока.

— Бит 3 **FREQ**: Запрос FREQ.

При передаче это мольба о записи в SAI_xDR.

При передаче это реклама чтения из SAI_xDR.

Прерывание разрешают битом SAI_xIM/FREQIE.

— Бит 2 **WCKCFG**: Флаг ошибки конфигурации тактов (SAI_xFRCR/FRL[7:0])

Только у ведущего (SAI_xCR1/MODE[1]=0) при SAI_xCR1/NODIV=0.

Прерывание разрешают битом SAI_xIM/WCKCFGIE. Снимают установкой SAI_xCLRFR/CWCKCFG.

— Бит 1 **MUTEDET**: Флаг появления на входе MUTECNT последовательных немых фреймов.

Прерывание разрешают битом SAI_xIM/MUTEDETIE. Снимают установкой SAI_xCLRFR/CMUTEDET.

— Бит 0 **OVRUDR**: Ошибка переполнения(приём)/исчерпания(передача) FIFO.

Прерывание разрешают битом SAI_xIM/OVRUDRIE. Снимают установкой SAI_xCLRFR/COVRUDR.

29.17.7.Регистр очистки флагов (SAI_xCLRFR) где x = A или B

Смещение адреса: Блок A: 0x01C

Смещение адреса: Блок B: 0x03C

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved									CLFSDET	CAFSDET	CCNRDY	Reserved	CWCKCFG	CMUTEDET	COVRUDR
									rw	rw	rw		rw	rw	rw

Пишется программно.

0: Ничего

1: Снять флаг

— Биты 31:7 Резерв, не трогать.

— Бит 6 **CLFSDET**: Чистка флага LFSDET.

Бит не имеет смысла в режиме AC’97.

— Бит 5 **CAFSDET**: Чистка флага AFSDET.

Бит не имеет смысла в режиме AC’97.

— Бит 4 **CCNRDY**: Чистка флага CNRDY.

Бит имеет смысл только в режиме AC’97.

— Бит 3 Резерв, не трогать.

— Бит 2 **CWCKCFG**: Чистка флага WCKCFG.

— Бит 1 **CMUTEDET**: Чистка флага MUTEDET.

— Бит 0 **COVRUDR**: Чистка флага OVRUDR.

29.17.8.Регистр данных (SAI_xDR) где x = A или B

Смещение адреса: Блок A: 0x020

Смещение адреса: Блок B: 0x040

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:0 **DATA[31:0]**: Данные.

Запись регистра это загрузка в FIFO, если он не полон.

Чтение регистра это извлечение из FIFO, если он не пуст.

29.17.9.Карта регистров SAI

Table 132. Карта регистров SAI

0x0004 or 0x0024	SAI_xCR1	Reserved								MCJDIV[3:0]				NODIV	Res.	DMAEN	SAIxEN	Reserved.		OutDri	MONO	SYNCEN[1:0]		CKSTR	LSBFIRST	DS[2:0]			Res.	PRTCFG[1:0]		MODE[1:0]	
	Reset value									0	0	0	0	0		0	0			0	0	0	0	0	0	0	1	0		0	0	0	0
0x0008 or 0x0028	SAI_xCR2	Reserved																COMP[1:0]		CPL	MUTECN[5:0]						MUTE VAL	MUTE	TRIS	FFLUS	FTH		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x000Cor 0x002C	SAI_xFRCR	Reserved													FSOFF	FSPOL	FSDEF	Reserved	FSALL[6:0]							FRL[7:0]							
	Reset value														0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0x0010 or 0x0030	SAI_xSLOTR	SLOTEN[15:0]																Reserved				NBSLOT[3:0]				SLOTSZ[1:0}		Reserved	FBOFF[4:0]				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0		0	0	0	0	0
0x0014 or 0x0034	SAI_xIM	Reserved																									LFSDET	AFSDETIE	CNRDYIE	FREQIE	WCKCFG	MUTEDET	OVRUDRIE
	Reset value																										0	0	0	0	0	0	0
0x0018 or 0x0038	SAI_xSR	Reserved													FLVL[2:0]			Reserved									LFSDET	AFSDET	CNRDY	FREQ	WCKCFG	MUTEDET	OVRUDR
	Reset value														0	0	0										0	0	0	0	1	0	0

Offset	Register and reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x001C or 0x003C	SAI_xCLRFR	Reserved																									LFSDET	CAFSDET	CNRDY	Res.	WCKCFG	MUTEDET	OVRUDR
	Reset value																										0	0	0		0	0	0
0x0020 or 0x0040	SAI_xDR	DATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

30. Интерфейс USART

30.1. Введение в USART

Универсальный синхронно-асинхронный приёмо-передатчик (USART) предлагает гибкие способы полно-дуплексного обмена с внешним оборудованием по стандарту NRZ в широком диапазоне скоростей с дробной частотой.

Поддерживает синхронную одностороннюю связь и полудуплекс по одному проводу. Также поддерживается LIN (коммутируемая локальная сеть), протокол Smartcard, спецификации IrDA SIR ENDEC и работа модема (CTS/RTS). Дозволена многопроцессорная связь.

Высокоскоростная связь может использовать многобуферный DMA.

30.2. Основные свойства USART

- Полный дуплекс, асинхронная связь
- Формат стандарта NRZ (Mark/Space)
- Множественная выборка по 16 или по 8
- Генератор дробной скорости
  - —Общая программируемая скорость приёма и передачи до 4.5 Мбит/с
- Программируемая длина слова (8 или 9 бит)
- Программируемые биты стоп (1 или 2)
- Посылка синхронного LIN Break ведущим и приём LIN Break ведомым
  - —генерация 13-бит Break и обнаружение 10/11 бит Break при USART в режиме LIN
- Вывод тактов передатчика для синхронных операций
- IrDA SIR Кодер/Декодер
  - —Поддержка 3/16 бит длительности нормального режима
- Эмуляция Smartcard
  - —Интерфейс Smartcard поддерживает асинхронный стандарт Smartcards ISO 7816-3
  - —0.5, 1.5 Стоп Биты для операций Smartcard operation
- Однопроводная полудуплексная связь
- DMA с несколькими буферами
  - —Буферы байтов приёма/передачи в резервной SRAM с центральным DMA
- Раздельные биты включения приёма и передачи
- Флаги передачи:
  - —Буфер приёма полон
  - —Буфер передачи пуст
  - —Флаги Конца Передачи
- Контроль чётности:
  - —Бит чётности передачи
  - —Проверка чётности принятого байта данных
- Четыре флага ошибок:

  - —Переполнение
  - —Ошибка шума
  - —Ошибка фрейма
  - —Ошибка чётности
- Десять источников прерываний с флагами:
  - —CTS изменился
  - —Обнаружен LIN break
  - —Регистр данных передачи пуст
  - —Передача завершена
  - —Регистр данных приёма полон
  - —Принят Простой линии
  - —Переполнение
  - —Ошибка фрейма
  - —Ошибка шума
  - —Ошибка чётности
- Многопроцессорная связь - уход в немоту при несовпадении адреса
- Пробуждение из немоты (по простою линии или метке адреса)
- Два режима пробуждения приёмника: Бит адреса (девятый бит), Простой линии.

30.3. Функциональное описание USART

Наружу торчат три ножки, но для двунаправленной последовательной связи хватает двух: Приём данных (RX) и Передача данных (TX):

RX: Для выделения сигнала из шума используется множественная выборка.

TX: При выключенном передатчике эта нога возвращается порту IO. Если включённому передатчику говорить нечего, то нога TX стоит высокой. В однопроводных режимах она используется для приёма и передачи (на уровне USART данные затем принимаются на `SW_RX`).

Данные оформляются в фреймы:

- Простой линии перед приёмом и передачей
- Стартовый бит
- Слово данных (8 или 9 бит), младшим вперёд
- 0.5,1, 1.5, 2 стоповых бита конца фрейма
- Используется дробный генератор частоты - 12-бит мантисса и 4-бит дробь
- Регистр состояния (`USART_SR`)
- Регистр данных (`USART_DR`)
- Регистр частоты (`USART_BRR`) - 12-бит мантисса и 4-бит дробь.
- Регистр защиты (Guardtime Register) (`USART_GTPR`) в режиме Smartcard.

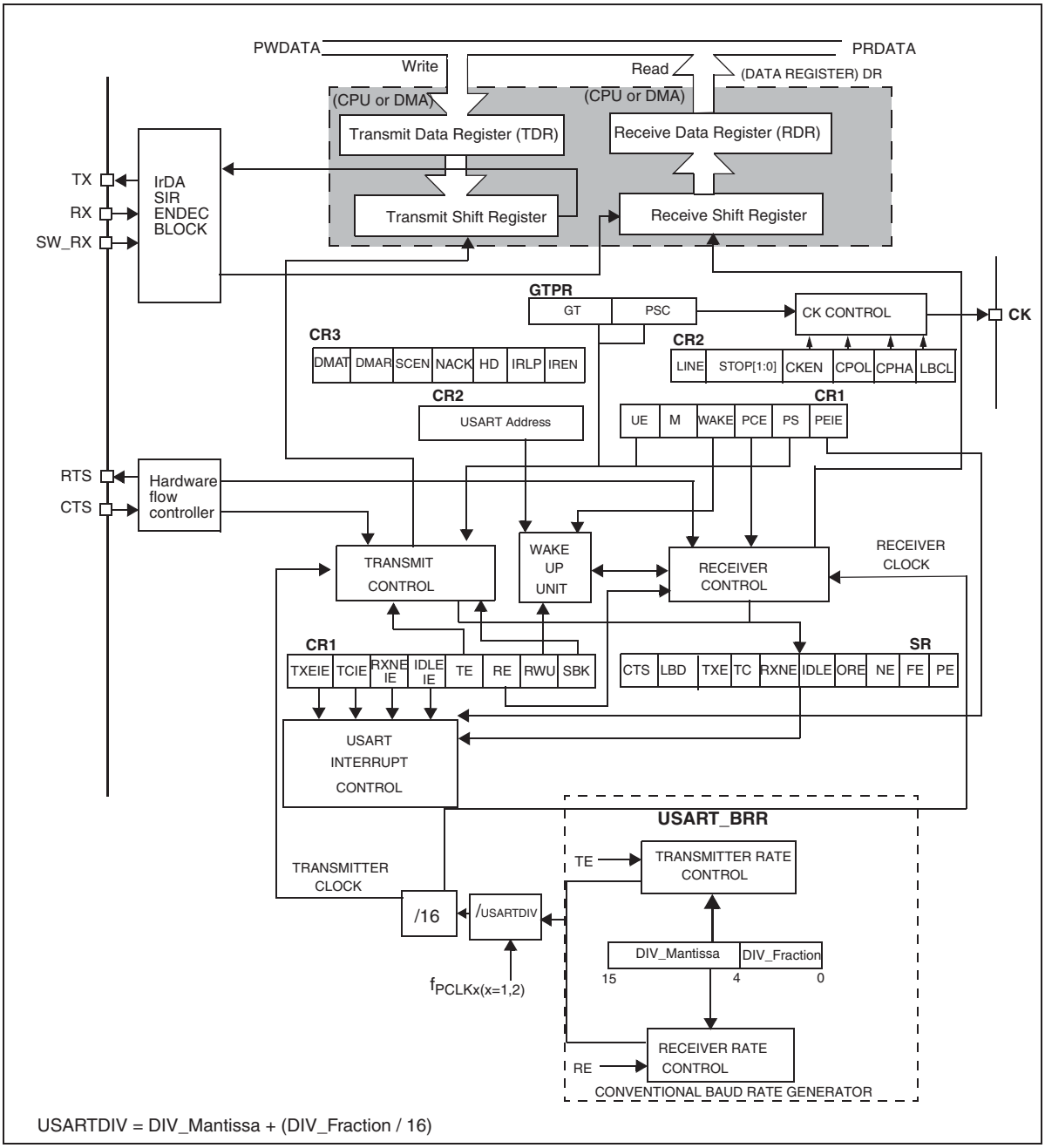
В синхронном режиме работают ноги:

- **CK:** Такты передатчика. Соответствует режиму ведущего SPI (биты старт и стоп без импульса, необязательный импульс на последний бит данных). Параллельно можно синхронно принимать данные на RX. Фаза и полярность тактов программируются. Можно использовать в режиме Smartcard.

При аппаратном управлении используются ноги:

- **CTS:** Высоким блокирует передачу данных в конце текущей передачи
- **RTS:** Низким запрашивает передачу данных.

Рис. 296. Блок-схема USART



30.3.1. Описание символа USART

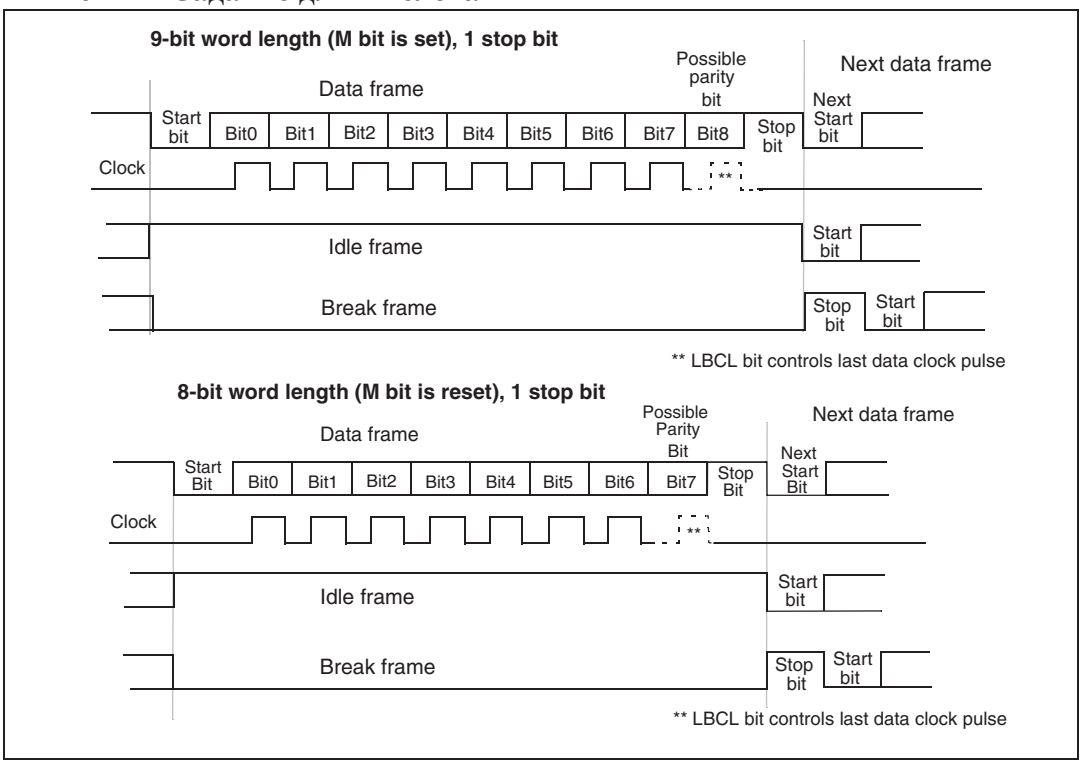
Длина слова (8 или 9 бит) выбирается битом `USART_CR1/M`. Ножка TX во время бита старта низкая, а при бите стопа - высокая.

Символ простоя это целый фрейм единиц с последующим битом старта следующего фрейма с данными (число единиц включает биты стопа).

Символ разрыва это нули во время всего периода фрейма. В конце фрейма разрыва выдаются нормальные биты стоп (логическая 1) чтобы узнать бит старта.

Такты от общего генератора подаются приёмнику и передатчику при их включении нужным битом.

Рис. 297. Задание длины слова



30.3.2. Передатчик

Передача включается установкой бита TE, начиная выдачу битов из регистра сдвига, начиная с младшего, на ножку TX и тактовых импульсов на ножку CK.

Передача символа

В регистр сдвига данные поступают из регистра данных USART_DR (TDR), а туда с шины. Каждому символу предшествует бит старта низкого уровня. Завершается символ 0.5, 1, 1.5 или 2 битами стоп.

NB: Бит TE нельзя снимать во время передачи данных.

После установки бита TE посылается фрейм простоя.

Биты Стоп

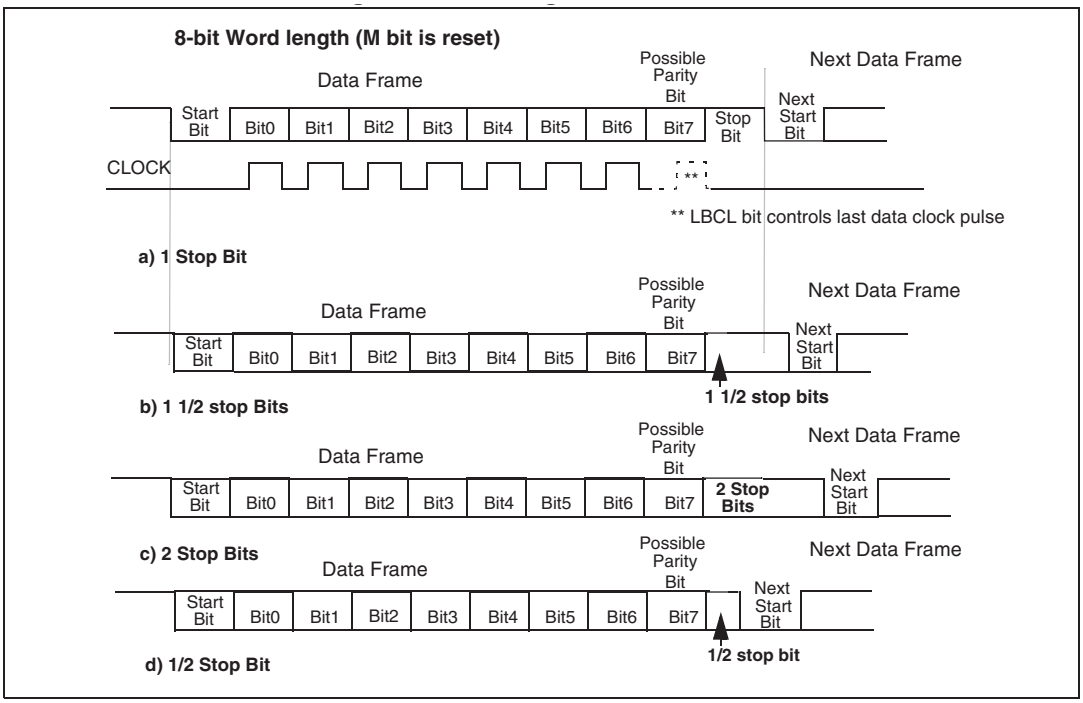
Число битов стоп задаются битами 13,12 регистра управления 2.

1. **1 stop bit:** Значение по умолчанию.
2. **2 stop bits:** Режим одного провода и режим модема USART.
3. **0.5 stop bit:** Приём данных в режиме Smartcard.
4. **1.5 stop bits:** Приём и передача данных в режиме Smartcard.

Фрейм простоя включает биты стоп.

Передача Разрыва это 10 низких битов с последующими битами стоп (при M = 0) или 11 низких битов с последующими битами стоп (when M = 1). Более длинные Разрывы передавать нельзя.

Рис. 298. Биты Стоп



Процедура:

1. Включаем USART установкой бита UE в регистре USART_CR1.
2. Задаём длину слова битом M в регистре USART_CR1.
3. Пишем число битов стопа в регистре USART_CR2.
4. Выбираем DMA (DMAT) в регистре USART_CR3 при многих буферах надобных. Заполняем регистры DMA.
5. Выбираем скорость регистром USART_BRR.
6. Ставим бит TE в регистре USART_CR1 ради посылки фрейма простоя.
7. Пишем посылаемые данные в регистр USART_DR (бит TXE снимается). Повторяем для каждого посылаемого данного при использовании одного буфера.
8. После записи в USART_DR последнего данного, ждём сигнала конца передачи фрейма установленным битом TC=1.

Однобайтовая связь

Бит TXE снимается записью в регистр данных, ставится он аппаратно, указывая:

- Данные переданы из TDR в регистр сдвига и передача началась.
- Регистр TDR пуст.
- Надо писать в USART_DR следующее данное.

При стоящем бите TXEIE выдаётся прерывание.

Запись в USART_DR во время передачи пишет в регистр TDR.

Запись в USART_DR не во время передачи пишет сразу в регистр сдвига, начинает передачу и ставит бит TXE.

После передачи битов стопа всего фрейма и установки бита TXE ставится бит TC. При стоящем бите TCIE регистра USART_CR1 выдаётся прерывание.

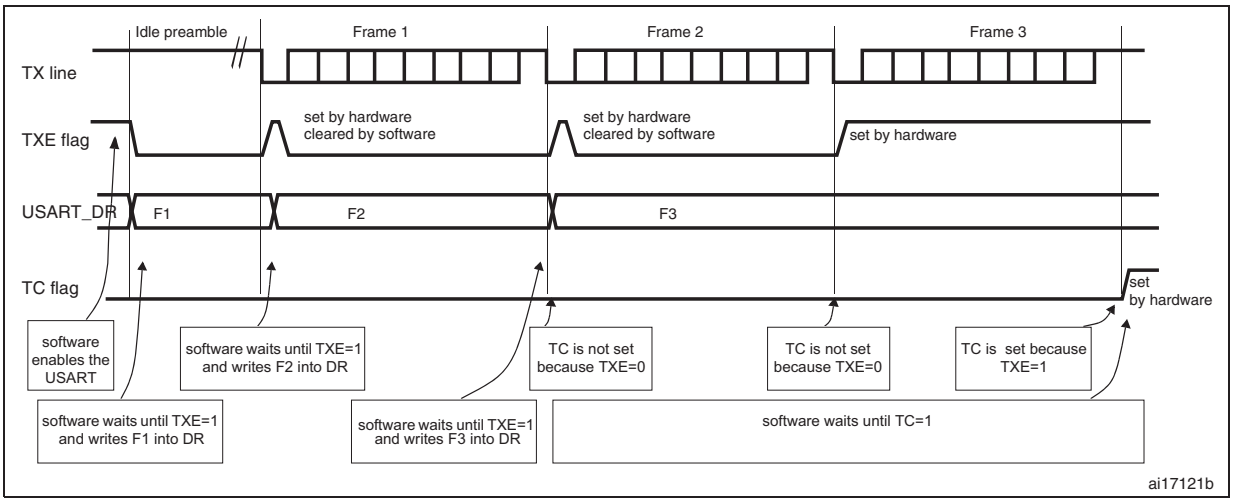
Перед выключением USART с возможным Сном MCU, после записи последнего данного в регистр USART_DR, надо обязательно дождаться TC=1.

Программно бит TC чистится так:

1. Читаем регистр USART_SR
2. Пишем в регистр USART_DR

Бит TC можно снять записью нуля, но это рекомендуется только в режиме многих буферов.

Рис. 299. TC/TXE при передаче



Символ разрыва

Установка бита SBK передаёт символ разрыва. Длина фрейма разрыва зависит от бита M.

Стоящий бит SBK после завершения передачи текущего символа посылает на линию TX символ разрыва. Снимается SBK аппаратно во время передачи битов стоп символа разрыва. USART вставляет логическую 1 в конец фрейма разрыва, чтобы узнавать бит старта следующего фрейма.

Снятие бита SBK до начала передачи разрыва отменяет саму передачу. Для передачи двух последовательных разрывов бит SBK надо ставить после битов стопа предыдущего разрыва.

Символ простоя

Установка бита TE понуждает USART послать фрейм простоя перед первым фреймом данных.

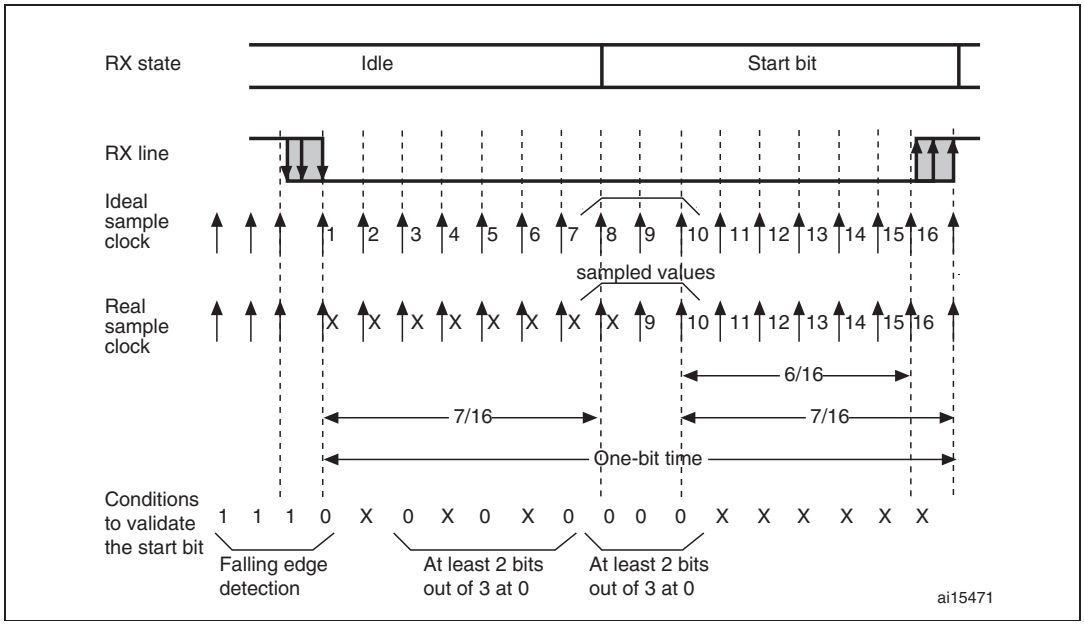
30.3.3. Приёмник

Длина принимаемых слов (8 или 9 битов) определяется битом M в регистре USART_CR1.

Обнаружение бита старта

USART узнаёт бит старта по последовательности битов 1 1 1 0 X 0 X 0 X 0 0 0 0.

Рис. 300. Обнаружение бита старта при множественной выборке 16 или 8



NB: Если последовательность не завершена, то приёмник перестаёт искать бит и идёт в простой, где и ждёт заднего фронта (флаг не ставится).

Старт подтверждается (ставится бит RXNE, при RXNEIE=1 выдаётся прерывание) при 3 выборках нуля (первый ноль на 3м, 5м и 7м импульсах и второй ноль на 8м, 9м и 10м импульсах выборки).

Старт принимается (ставится бит RXNE, при RXNEIE=1 выдаётся прерывание), но ставится флаг шума NE, если для обеих выборок (на 3м, 5м и 7м импульсах и на 8м, 9м и 10м импульсах выборки)

не меньше 2 из 3 раз читается 0. Если не так, то приёмник перестаёт искать бит и идёт в простой, где и ждёт заднего фронта (флаг не ставится).

Приём символа

Во время приёма вдвигаются, начиная с младшего бита, с ножки RX в регистр сдвига. Буфер USART_DR (RDR) получает данное из него.

Процедура:

1. Включаем USART установкой бита UE в регистре USART_CR1.
2. Задаём длину слова битом M в регистре USART_CR1.
3. Пишем число битов стопа в регистре USART_CR2.
4. Выбираем DMA (DMAR) в регистре USART_CR3 при многих буферах надобных. Заполняем регистры DMA.
5. Выбираем скорость регистром USART_BRR.
6. Ставим бит RE в регистре USART_CR1. Приёмник начинает искать бит старта.

Когда символ принят:

- Ставится бит RXNE. То есть, регистр сдвига переписан в RDR и его можно читать.
- При стоящем бите RXNEIE выдаётся прерывание.
- Могут встать флаги ошибки фрейма, шума и переполнения.
- При многих буферах бит RXNE ставится после каждого байта и снимается DMA.
- При одном буфере бит RXNE снимается программно чтением регистра USART_DR. Он может сниматься записью ноля. Во избежание переполнения чистить RXNE нужно до конца приёма следующего байта.

NB: Бит RE нельзя снимать во время приёма.

Символ разрыва

Обрабатывается как ошибка фрейма.

Символ простоя

Принимается как обычное данное, при стоящем бите IDLEIE выдаётся прерывание.

Переполнение

Возникает после приёма символа при стоящем RXNE. Данные в регистр RDR не переносятся.

Флаг RXNE ставится после приёма каждого байта. При появлении прерывания:

- Ставится бит ORE.
- Содержимое RDR не теряется.
- Следующим байтом перепишется регистр сдвига.
- Прерывание выдаётся при стоящем бите RXNEIE или обоих стоящих битах EIE DMAR.
- Бит ORE снимается чтением регистра USART_SR с последующим чтением USART_DR.

NB: Стоящий бит ORE означает потерю не менее 1 данного. Есть два выхода:

- При RXNE=1, можно прочитать из RDR последний достоверный байт,
- При RXNE=0, из RDR уже всё прочитали, но одновременно с приёмом нового (потерянного) байта или между чтением USART_SR и чтением USART_DR.

Выбор метода многократной выборки

Для обнаружения используется метод многократной выборки (исключая синхронный режим).

Бит USART_CR1/OVER8 назначает метод выборки по 8 или 16 раз за такт.

В зависимости от задачи:

- При OVER8=1 всё идёт быстрее (скорость до $f_{PCLK}/8$), но падает толерантность приёмника к девиации тактов
- При OVER8=0 толерантность поднимается, но скорость только до $f_{PCLK}/16$

Бит USART_CR3/ONEBIT задаёт метод определения логического уровня:

- мажоритарное голосование трёх выборок в центре принятого бита. Тут если три выборки не равны, то ставится бит NF
- одна выборка в центре принятого бита

При обнаружении шума в фрейме:

- По переднему фронту бита RXNE ставится бит NE.
- В регистр USART_DR пишутся непотребные данные.
- При однобайтовой передаче прерывание ошибки не выдаётся, но бит RXNE ставится. При многобуферной передаче прерывание ошибки разрешается битом USART_CR3/EIE.

Бит NE снимается чтением регистра USART_SR с последующим чтением регистра USART_DR.

Выборка по 8 недоступна в режимах Smartcard, IrDA и LIN. Бит OVER8 снимается аппаратно.

Рис. 301. Многая выборка по 16

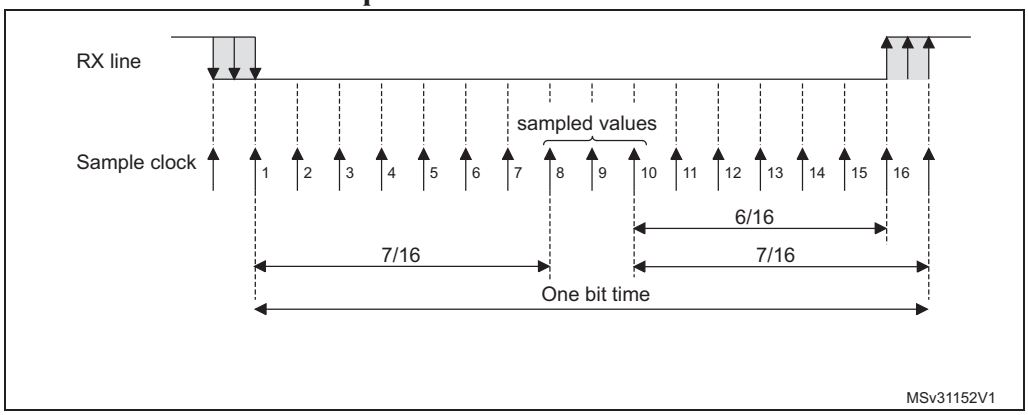


Рис. 302. Многая выборка по 8

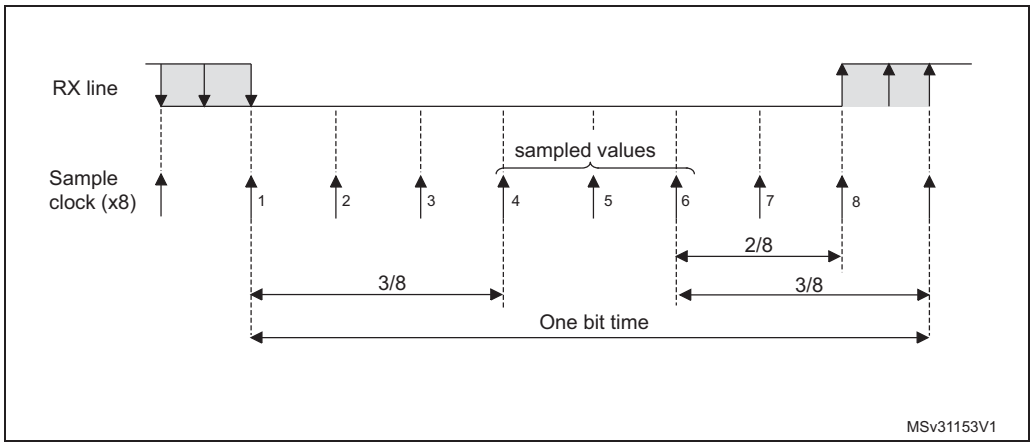


Таблица 133. Выделение шума

Считанное значение	Состояние NE	Принятый бит
000	0	0
001	1	0
010	1	0
011	1	1
100	1	0
101	1	1
110	1	1
111	0	1

Ошибка фрейма

Возникает когда в нужное время не обнаруживаются биты стоп с последующей потерей синхронизации или при ужасном шуме.

Если появилась:

- Аппаратно ставится бит FE
- В регистр USART_DR пишется лажа.
- При однобайтовой передаче прерывание ошибки не выдаётся, но бит RXNE ставится. При многобуферной передаче прерывание ошибки разрешается битом USART_CR3/EIE.

Бит FE снимается чтением регистра USART_SR с последующим чтением регистра USART_DR.

Биты стоп при приёме

В нормальном режиме используются 1 или 2 бита, в режиме Smartcard 0.5 или 1.5 бита.

— **0.5 бита стоп (приём в Smartcard)**: Выборки 0.5 бита стопа нет, так что ошибка фрейма и разрыв не обнаруживаются.

— **1 бит стоп**: Чтение 1 по 8, 9 и 10 импульсам выборки.

— **1.5 бита стоп (Smartcard)**: При передаче в режиме Smartcard устройство должно проверять правильность отсылаемых данных. Так что, блок приёма должен быть включён (USART_CR1/RE=1) и бит стоп проверяется на предмет обнаружения устройством ошибки чётности. В случае ошибки устройство удерживает сигнал данных низким (NACK), сообщая об ошибке фрейма. Флаг FE ставится вместе с RXNE в конце 1.5 битов стоп. Определение 1.5 битов стоп выполняется по 16, 17 и 18 импульсам выборки (1 такт связи после начала бита стоп).

1.5 бита можно разделить на 2 части: 0.5 такта без событий и 1 нормальный такт с выборкой бита стоп в его середине. См. *Секцию 27.3.11*.

— **2 бита стоп**: Считывание битов стоп выполняется по 8, 9 и 10 импульсам выборки. Ошибка фрейма определяется при первом бите стоп, при втором ошибка не проверяется. Флаг RXNE ставится по концу первого бита стоп.

30.3.4. Генерация дробной частоты

Частота приёмника и передатчика (Rx и Tx) задаётся мантиссой и дробной частью числа в регистре USARTDIV.

Уравнение 1: Скорость для стандартного USART (включая режим SPI)

$$\text{Tx/Rx baud} = \frac{f_{CK}}{8 \times (2 - \text{OVER8}) \times \text{USARTDIV}}$$

Уравнение 2: Скорость для режимов Smartcard, LIN и IrDA

$$\text{Tx/Rx baud} = \frac{f_{CK}}{16 \times \text{USARTDIV}}$$

USARTDIV это беззнаковое число с фиксированной запятой, в регистре USART_BRR.

- При OVER8=0, дробная часть это 4 бита в USART_BRR/DIV_fraction[3:0]
- При OVER8=1, дробная часть это 3 бита в USART_BRR/DIV_fraction[2:0], бит DIV_fraction[3] должен быть чистым.

NB: Счётчик тактов связи обновляется из регистра USART_BRR. То есть, во время работающей связи этот регистр менять нельзя.

Как получить USARTDIV из значения регистра USART_BRR при OVER8=0

Пример 1:

Если DIV_Mantissa = 0d27 и DIV_Fraction = 0d12 (USART_BRR = 0x1BC), то

Мантисса (USARTDIV) = 0d27

Дробная часть (USARTDIV) = 12/16 = 0d0.75

Так что USARTDIV = 0d27.75

Пример 2:

Пишем USARTDIV = 0d25.62

Далее:

DIV_Fraction = 16*0d0.62 = 0d9.92

Ближайшее целое 0d10 = 0xA

DIV_Mantissa = мантисса (0d25.62) = 0d25 = 0x19

Тогда, USART_BRR = 0x19A отсюда USARTDIV = 0d25.625

Пример 3:

Пишем USARTDIV = 0d50.99

Далее:

DIV_Fraction = 16*0d0.99 = 0d15.84

Ближайшее целое 0d16 = 0x10 => переполнение DIV_frac[3:0] => перенос добавляем к мантиссе

DIV_Mantissa = мантисса (0d50.990 + перенос) = 0d51 = 0x33

Тогда, USART_BRR = 0x330 отсюда USARTDIV = 0d51.000

Как получить USARTDIV из значения регистра USART_BRR при OVER8=0

Пример 1:

Если DIV_Mantissa = 0x27 и DIV_Fraction = 0d6 (USART_BRR = 0x1B6), то

Мантисса (USARTDIV) = 0d27

Дробная часть (USARTDIV) = 6/8 = 0d0.75

Так что USARTDIV = 0d27.75

Пример 2:

Пишем USARTDIV = 0d25.62

Далее:

DIV_Fraction = 8*0d0.62 = 0d4.96

Ближайшее целое 0d5 = 0x5

DIV_Mantissa = мантисса (0d25.62) = 0d25 = 0x19

Тогда, USART_BRR = 0x195 отсюда USARTDIV = 0d25.625

Пример 3:

Пишем USARTDIV = 0d50.99

Далее:

DIV_Fraction = 8*0d0.99 = 0d7.92

Ближайшее целое 0d8 = 0x8 => переполнение DIV_frac[2:0] => перенос добавляем к мантиссе

DIV_Mantissa = мантисса (0d50.990 + перенос) = 0d51 = 0x33

Тогда, USART_BRR = 0x330 отсюда USARTDIV = 0d51.000

Таблица 134. Вычисление ошибки при f_{PCLK} = 8 MHz или f_{PCLK} = 12 MHz, выборке по 16(1)

Выборка по 16 (OVER8=0)							
Baud rate		f_{PCLK} = 8 MHz			f_{PCLK} = 12 MHz		
S.No	Желаемая (Хочу)	Действ.	Число в регистре baud rate	% Ошибки = (Вычисл. - Хочу) B.rate / Хочу B.rate	Действ.	Число в регистре baud rate	% Ошибки
1	1.2 KBps	1.2 KBps	416.6875	0	1.2 KBps	625	0
2	2.4 KBps	2.4 KBps	208.3125	0.01	2.4 KBps	312.5	0
3	9.6 KBps	9.604 KBps	52.0625	0.04	9.6 KBps	78125	0
4	19.2 KBps	19.185 KBps	26.0625	0.08	19.2 KBps	39.0625	0
5	38.4 KBps	38.462 KBps	13	0.16	38.339 KBps	19.5625	0.16
6	57.6 KBps	57.554 KBps	8.6875	0.08	57.692 KBps	13	0.16
7	115.2 KBps	115.942 KBps	4.3125	0.64	115.385 KBps	6.5	0.16
8	230.4 KBps	228.571 KBps	2.1875	0.79	230.769 KBps	3.25	0.16
9	460.8 KBps	470.588 KBps	1.0625	2.12	461.538 KBps	1625	0.16
10	921.6 KBps	NA	NA	NA	NA	NA	NA
11	2 MBps	NA	NA	NA	NA	NA	NA
12	3 MBps	NA	NA	NA	NA	NA	NA

1. Точность и скорость передачи зависят от тактов CPU.

Таблица 135. Вычисление ошибки при f_{PCLK} = 8 MHz или f_{PCLK} = 12 MHz, выборке по 8(1)

Выборка по 8 (OVER8=1)							
Baud rate		f_{PCLK} = 8 MHz			f_{PCLK} = 12 MHz		
S.No	Желаемая (Хочу)	Действ.	Число в регистре baud rate	% Ошибки = (Вычисл. - Хочу) B.rate / Хочу B.rate	Действ.	Число в регистре baud rate	% Ошибки
1	1.2 KBps	1.2 KBps	833375	0	1.2 KBps	1250	0
2	2.4 KBps	2.4 KBps	416625	0.01	2.4 KBps	625	0
3	9.6 KBps	9.604 KBps	104125	0.04	9.6 KBps	156.25	0
4	19.2 KBps	19.185 KBps	52125	0.08	19.2 KBps	78125	0
5	38.4 KBps	38.462 KBps	26	0.16	38.339 KBps	39125	0.16
6	57.6 KBps	57.554 KBps	17375	0.08	57.692 KBps	26	0.16
7	115.2 KBps	115.942 KBps	8625	0.64	115.385 KBps	13	0.16
8	230.4 KBps	228.571 KBps	4375	0.79	230.769 KBps	6.5	0.16
9	460.8 KBps	470.588 KBps	2125	2.12	461.538 KBps	3.25	0.16
10	921.6 KBps	888.889 KBps	1125	3.55	923.077 KBps	1625	0.16
11	2 MBps	NA	NA	NA	NA	NA	NA
12	3 MBps	NA	NA	NA	NA	NA	NA

1. Точность и скорость передачи зависят от тактов CPU.

Таблица 136. Вычисление ошибки при f_{PCLK} = 16 MHz или f_{PCLK} = 24 MHz, выборке по 16(1)

Выборка по 16 (OVER8=0)							
Baud rate		f_{PCLK} = 16 MHz			f_{PCLK} = 24 MHz		
S.No	Желаемая (Хочу)	Действ.	Число в регистре baud rate	% Ошибки = (Вычисл. - Хочу) B.rate / Хочу B.rate	Действ.	Число в регистре baud rate	% Ошибки
1	1.2 KBps	1.2 KBps	833.3125	0	1.2	1250	0
2	2.4 KBps	2.4 KBps	416.6875	0	2.4	625	0
3	9.6 KBps	9.598 KBps	104.1875	0.02	9.6	156.25	0
4	19.2 KBps	19.208 KBps	52.0625	0.04	19.2	78125	0
5	38.4 KBps	38.369 KBps	26.0625	0.08	38.4	39.0625	0
6	57.6 KBps	57.554 KBps	17375	0.08	57554	26.0625	0.08
7	115.2 KBps	115.108 KBps	8.6875	0.08	115385	13	0.16
8	230.4 KBps	231.884 KBps	4.3125	0.64	230769	6.5	0.16
9	460.8 KBps	457.143 KBps	2.1875	0.79	461538	3.25	0.16
10	921.6 KBps	941.176 KBps	1.0625	2.12	923077	1625	0.16
11	2 MBps	NA	NA	NA	NA	NA	NA
12	3 MBps	NA	NA	NA	NA	NA	NA

1. Точность и скорость передачи зависят от тактов CPU.

Таблица 137. Вычисление ошибки при f_{PCLK} = 16 MHz или f_{PCLK} = 24 MHz, выборке по 8(1)

Выборка по 8 (OVER8=1)							
Baud rate		f_{PCLK} = 16 MHz			f_{PCLK} = 24 MHz		
S.No	Желаемая (Хочу)	Действ.	Число в регистре baud rate	% Ошибки = (Вычисл. - Хочу) B.rate / Хочу B.rate	Действ.	Число в регистре baud rate	% Ошибки
1	1.2 KBps	1.2 KBps	1666.625	0	1.2 KBps	2500	0
2	2.4 KBps	2.4 KBps	833375	0	2.4 KBps	1250	0
3	9.6 KBps	9.598 KBps	208375	0.02	9.6 KBps	312.5	0
4	19.2 KBps	19.208 KBps	104125	0.04	19.2 KBps	156.25	0
5	38.4 KBps	38.369 KBps	52125	0.08	38.4 KBps	78125	0
6	57.6 KBps	57.554 KBps	34.75	0.08	57.554 KBps	52125	0.08

7	115.2 KBps	115.108 KBps	17375	0.08	115.385 KBps	26	0.16
8	230.4 KBps	231.884 KBps	8625	0.64	230.769 KBps	13	0.16
9	460.8 KBps	457.143 KBps	4375	0.79	461.538 KBps	6.5	0.16
10	921.6 KBps	941.176 KBps	2125	2.12	923.077 KBps	3.25	0.16
11	2 MBps	2000 KBps	1	0	2000 KBps	1.5	0
12	3 MBps	NA	NA	NA	3000 KBps	1	0

1. Точность и скорость передачи зависят от тактов CPU.

Таблица 138. Вычисление ошибки при f_{PCLK} = 8 MHz или f_{PCLK} = 16 MHz, выборке по 16(1)

Выборка по 16 (OVER8=0)							
Baud rate		f_{PCLK} = 8 MHz			f_{PCLK} = 16 MHz		
S.No	Желаемая (Хочу)	Действ.	Число в регистре baud rate	% Ошибки = (Вычисл. - Хочу) B.rate / Хочу B.rate	Действ.	Число в регистре baud rate	% Ошибки
1.	2.4 KBps	2.400 KBps	208.3125	0 %	2.400 KBps	416.6875	0 %
2.	9.6 KBps	9.604 KBps	52.0625	0 %	9.598 KBps	104.1875	0 %
3.	19.2 KBps	19.185 KBps	26.0625	0 %	19.208 KBps	52.0625	0 %
4.	57.6 KBps	57.554 KBps	8.6875	0 %	57.554 KBps	17.3750	0 %
5.	115.2 KBps	115.942 KBps	4.3125	1 %	115.108 KBps	8.6875	0 %
6.	230.4 KBps	228.571 KBps	2.1875	1 %	231.884 KBps	4.3125	1 %
7.	460.8 KBps	470.588 KBps	1.0625	2 %	457.143 KBps	2.1875	1 %
8.	896 KBps	NA	NA	NA	888.889 KBps	1.1250	1 %
9.	921.6 KBps	NA	NA	NA	941.176 KBps	1.0625	2 %
10.	1.792 MBps	NA	NA	NA	NA	NA	NA
11.	1.8432 MBps	NA	NA	NA	NA	NA	NA
12.	3.584 MBps	NA	NA	NA	NA	NA	NA
13.	3.6864 MBps	NA	NA	NA	NA	NA	NA
14.	7.168 MBps	NA	NA	NA	NA	NA	NA
15.	7.3728 MBps	NA	NA	NA	NA	NA	NA

1. Точность и скорость передачи зависят от тактов CPU.

Таблица 140. Вычисление ошибки при f_{PCLK} = 30 MHz или f_{PCLK} = 60 MHz, выборке по 16(1)(2)

Выборка по 16 (OVER8=0)							
Baud rate		f_{PCLK} = 30 MHz			f_{PCLK} = 60 MHz		
S.No	Желаемая (Хочу)	Действ.	Число в регистре baud rate	% Ошибки = (Вычисл. - Хочу) B.rate / Хочу B.rate	Действ.	Число в регистре baud rate	% Ошибки
1.	2.4 KBps	2.400 KBps	781.2500	0 %	2.400 KBps	1562.5000	0 %
2.	9.6 KBps	9.600 KBps	195.3125	0 %	9.600 KBps	390.6250	0 %
3.	19.2 KBps	19.194 KBps	97.6875	0 %	19.200 KBps	195.3125	0 %
4.	57.6 KBps	57.582KBps	32.5625	0 %	57.582 KBps	65.1250	0 %
5.	115.2 KBps	115.385 KBps	16.2500	0 %	115.163 KBps	32.5625	0 %
6.	230.4 KBps	230.769 KBps	8.1250	0 %	230.769KBps	16.2500	0 %
7.	460.8 KBps	461.538 KBps	4.0625	0 %	461.538 KBps	8.1250	0 %
8.	896 KBps	909.091 KBps	2.0625	1 %	895.522 KBps	4.1875	0 %
9.	921.6 KBps	909.091 KBps	2.0625	1 %	923.077 KBps	4.0625	0 %
10.	1.792 MBps	1.1764 MBps	1.0625	2 %	1.8182 MBps	2.0625	1 %
11.	1.8432 MBps	1.8750 MBps	1.0000	2 %	1.8182 MBps	2.0625	2 %
12.	3.584 MBps	NA	NA	NA	3.2594 MBps	1.0625	2 %
13.	3.6864 MBps	NA	NA	NA	3.7500 MBps	1.0000	2 %
14.	7.168 MBps	NA	NA	NA	NA	NA	NA
15.	7.3728 MBps	NA	NA	NA	NA	NA	NA

1. Точность и скорость передачи зависят от тактов CPU.
2. Только USART1 и USART6 тактируются от PCLK2. Остальные от PCLK1.

Таблица 141. Вычисление ошибки при f_{PCLK} = 30 MHz или f_{PCLK} = 60 MHz, выборке по 8(1)(2)

Выборка по 8 (OVER8=1)							
Baud rate		f_{PCLK} = 30 MHz			f_{PCLK} = 60 MHz		
S.No	Желаемая (Хочу)	Действ.	Число в регистре baud rate	% Ошибки = (Вычисл. - Хочу) B.rate / Хочу B.rate	Действ.	Число в регистре baud rate	% Ошибки
1.	2.4 KBps	2.400 KBps	1562.5000	0 %	2.400 KBps	3125.0000	0 %
2.	9.6 KBps	9.600 KBps	390.6250	0 %	9.600 KBps	781.2500	0 %
3.	19.2 KBps	19.194 KBps	195.3750	0 %	19.200 KBps	390.6250	0 %
4.	57.6 KBps	57.582 KBps	65.1250	0 %	57.582 KBps	130.2500	0 %
5.	115.2 KBps	115.385 KBps	32.5000	0 %	115.163 KBps	65.1250	0 %
6.	230.4 KBps	230.769 KBps	16.2500	0 %	230.769 KBps	32.5000	0 %
7.	460.8 KBps	461.538 KBps	8.1250	0 %	461.538 KBps	16.2500	0 %
8.	896 KBps	909.091 KBps	4.1250	1 %	895.522 KBps	8.3750	0 %
9.	921.6 KBps	909.091 KBps	4.1250	1 %	923.077 KBps	8.1250	0 %
10.	1.792 MBps	1.7647 MBps	2.1250	2 %	1.8182 MBps	4.1250	1 %
11.	1.8432 MBps	1.8750 MBps	2.0000	2 %	1.8182 MBps	4.1250	1 %
12.	3.584 MBps	3.7500 MBps	1.0000	5 %	3.5294 MBps	2.1250	2 %
13.	3.6864 MBps	3.7500 MBps	1.0000	2 %	3.7500 MBps	2.0000	2 %
14.	7.168 MBps	NA	NA	NA	7.5000 MBps	1.0000	5 %
15.	7.3728 MBps	NA	NA	NA	7.5000 MBps	1.0000	2 %

1. Точность и скорость передачи зависят от тактов CPU.
2. Только USART1 и USART6 тактируются от PCLK2. Остальные от PCLK1.

Таблица 142. Вычисление ошибки при f_{PCLK} = 42 MHz или f_{PCLK} = 84 MHz, выборке по 16(1)(2)

Выборка по 16 (OVER8=0)							
Baud rate		f_{PCLK} = 42 MHz			f_{PCLK} = 84 MHz		
S.No	Желаемая (Хочу)	Действ.	Число в регистре baud rate	% Ошибки = (Вычисл. - Хочу) B.rate / Хочу B.rate	Действ.	Число в регистре baud rate	% Ошибки
1.	1.2 KBps	1.2 KBps	2187.5	0	1.2 KBps	NA	0
2.	2.4 KBps	2.4 KBps	1093.75	0	2.4 KBps	2187.5	0
3.	9.6 KBps	9.6 KBps	273.4375	0	9.6 KBps	546875	0
4.	19.2 KBps	19.195 KBps	136.75	0.02	19.2 KBps	273.4375	0
5.	38.4 KBps	38.391 KBps	68375	0.02	38.391 KBps	136.75	0.02
6.	57.6 KBps	57.613 KBps	45.5625	0.02	57.613 KBps	91125	0.02
7.	115.2 KBps	115.068 KBps	22.8125	0.11	115.226 KBps	45.5625	0.02
8.	230.4 KBps	230.769 KBps	11375	0.16	230.137 KBps	22.8125	0.11
9.	460.8 KBps	461.538 KBps	5.6875	0.16	461.538 KBps	11375	0.16
10.	921.6 KBps	913.043 KBps	2875	0.93	923.076 KBps	5.6875	0.93
11.	1.792 MBps	1.826 MBps	1.4375	1.9	1.787 MBps	2.9375	0.27
12.	1.8432 MBps	1.826 MBps	1.4375	0.93	1.826 MBps	2875	0.93
13.	3.584 MBps	N.A	N.A	N.A	3.652 MBps	1.4375	1.9
14.	3.6864 MBps	N.A	N.A	N.A	3.652 MBps	1.4375	0.93
15.	7.168 MBps	N.A	N.A	N.A	N.A	N.A	N.A
16.	7.3728 MBps	N.A	N.A	N.A	N.A	N.A	N.A
17.	9 MBps	N.A	N.A	N.A	N.A	N.A	N.A
18.	10.5 MBps	N.A	N.A	N.A	N.A	N.A	N.A

1. Точность и скорость передачи зависят от тактов CPU.
2. Только USART1 и USART6 тактируются от PCLK2. Остальные от PCLK1.

Таблица 143. Вычисление ошибки при f_{PCLK} = 42 MHz или f_{PCLK} = 84 MHz, выборке по 8(1)(2)

Выборка по 8 (OVER8=1)							
Baud rate		f_{PCLK} = 42 MHz			f_{PCLK} = 84 MHz		
S.No	Желаемая (Хочу)	Действ.	Число в регистре baud rate	% Ошибки = (Вычисл. - Хочу) B.rate / Хочу B.rate	Действ.	Число в регистре baud rate	% Ошибки
1.	2.4 KBps	2.4 KBps	2187.5	0	2.4 KBps	NA	0
2.	9.6 KBps	9.6 KBps	546875	0	9.6 KBps	1093.75	0
3.	19.2 KBps	19.195 KBps	273.5	0.02	19.2 KBps	546875	0
4.	38.4 KBps	38.391 KBps	136.75	0.02	38.391 KBps	273.5	0.02
5.	57.6 KBps	57.613 KBps	91125	0.02	57.613 KBps	182.25	0.02
6.	115.2 KBps	115.068 KBps	45625	0.11	115.226 KBps	91125	0.02
7.	230.4 KBps	230.769 KBps	22.75	0.11	230.137 KBps	45625	0.11
8.	460.8 KBps	461.538 KBps	11375	0.16	461.538 KBps	22.75	0.16
9.	921.6 KBps	913.043 KBps	5.75	0.93	923.076 KBps	11375	0.93
10.	1.792 MBps	1.826 MBps	2875	1.9	1.787Mbps	5875	0.27
11.	1.8432 MBps	1.826 MBps	2875	0.93	1.826 MBps	5.75	0.93
12.	3.584 MBps	3.5 MBps	1.5	2.34	3.652 MBps	2875	1.9
13.	3.6864 MBps	3.82 MBps	1375	3.57	3.652 MBps	2875	0.93
14.	7.168 MBps	N.A	N.A	N.A	7 MBps	1.5	2.34
15.	7.3728 MBps	N.A	N.A	N.A	7.636 MBps	1375	3.57
16.	9 MBps	N.A	N.A	N.A	9.333 MBps	1125	3.7
17.	10.5 MBps	N.A	N.A	N.A	10.5 MBps	1	0

1. Точность и скорость передачи зависят от тактов CPU.
2. Только USART1 и USART6 тактируются от PCLK2. Остальные от PCLK1.

30.3.5. Устойчивость приёмника к колебаниям частоты

Асинхронный приёмник USART может нормально работать только при общем отклонении тактов системы меньшем устойчивости приёмника USART. Слагаемые общего отклонения:

- DTRA: Ошибка передатчика (включает ошибку генератора передатчика)
- DQUANT: Ошибка частоты дискретизации приёмника
- DREC: Отклонение генератора приёмника
- DTCL: Отклонение линии передачи (обычно из-за асимметрии времён переднего и заднего фронтов сигнала передатчика)

DTRA + DQUANT + DREC + DTCL < устойчивости приёмника USART

Устойчивость приёмника USART зависит от:

- 10- или 11-бит длины символа, согласно биту `USART_CR1/M`
- Выборки по 8 или 16, согласно биту `USART_CR1/OVER8`
- использования дробной частоты связи
- Учёта 1 или 3 бит выборки согласно биту `USART_CR3/ONEBIT`

Таблица 144. Устойчивость приёмника USART при DIV_Fraction равном 0

M bit	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT=0	ONEBIT=1	ONEBIT=0	ONEBIT=1
0	3.75%	4.375%	2.50%	3.75%
1	3.41%	3.97%	2.27%	3.41%

Таблица 194. Устойчивость приёмника USART при DIV_Fraction отличном от 0

M bit	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT=0	ONEBIT=1	ONEBIT=0	ONEBIT=1
0	3.33%	3.88%	2%	3%
1	3.03%	3.53%	1.82%	2.73%

30.3.6. Многопроцессорная связь

Многопроцессорную сеть можно создать объединяя несколько USART на одной линии. Выход TX одного (ведущего) подключён к входам RX других USART (ведомых). Их выходы TX проводным логическим И объединены на входе RX ведущего.

При этом излишней нагрузки на линию можно избежать передавая сообщения только нужным адресованным устройствам, оставшиеся переключаются в немой режим. В этом режиме:

- Биты приёма не встают.
- Приёмные прерывания запрещены.
- Бит USART_CR1/RWU стоит. RWU управляется программно и аппаратно.

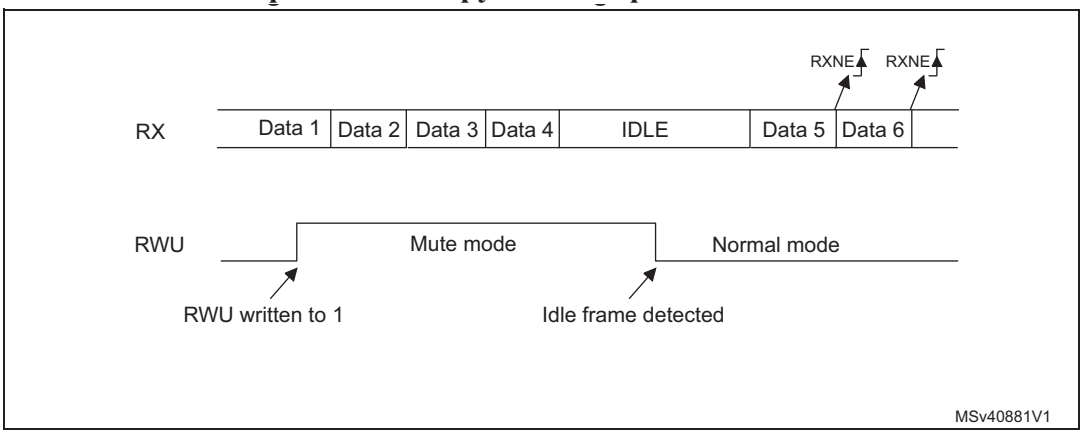
Методы входа и выхода USART из немого режима зависят от бита WAKE регистра USART_CR1:

- Обнаружение Простоя линии при сброшенном бите WAKE,
- Обнаружение Метки адреса при стоящем бите WAKE.

Обнаружение простоя линии (WAKE=0)

USART просыпается при появлении фрейма простоя. Бит RWU снимается аппаратно, но бит IDLE регистра USART_SR не ставится. RWU можно снимать программно.

Рис. 303. Немой режим с обнаружением простоя линии



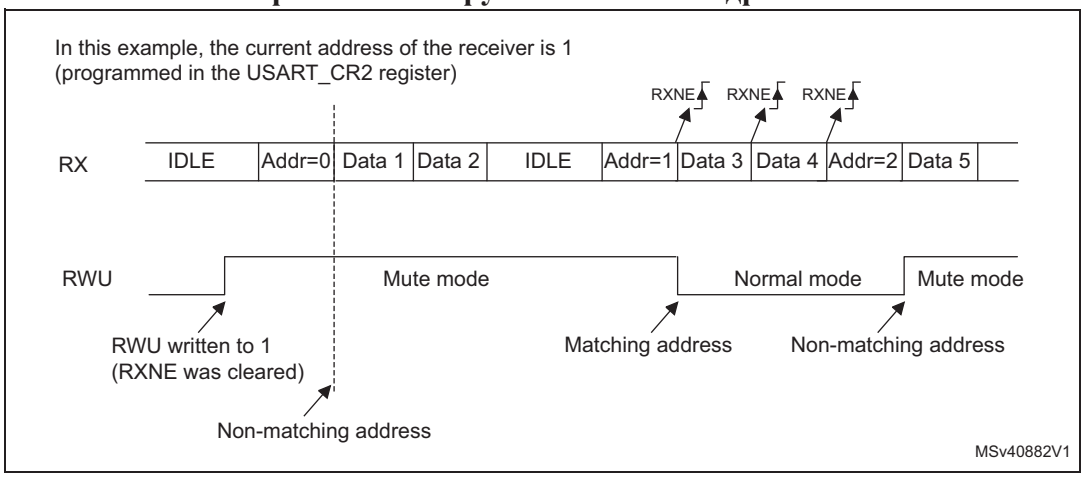
Обнаружение метки адреса (WAKE=1)

В этом режиме байт адреса распознаётся по стоящему старшему биту, младшие 4 бита это адрес устройства. Он аппаратно сравнивается с битами USART_CR2/ADD. При несовпадении USART уходит в немой режим, бит RWU ставится аппаратно, флаг RXNE не ставится, прерывание и запрос DMA не выдаются.

При совпадении адресов снимается бит RWU и последующие байты принимаются. Для символа адреса бит RXNE ставится, так как бит RWU уже очищен.

RWU можно писать только при пустом буфере (USART_SR/RXNE=0), иначе он не пишется.

Рис. 304. Немой режим с обнаружением метки адреса



30.3.7. Контроль чётности

Включается установкой бита USART_CR1/PCE.

Форматы фреймов USART зависят от бита M.

Таблица 195. Форматы фрейма(1)

Бит M	Бит PCE	Фрейм USART
0	0	I SB I 8 бит данных I STB I
0	1	I SB I 7 бит данных I PB I STB I
1	0	I SB I 9 бит данных I STB I
1	1	I SB I 8 бит данных I PB I STB I

1. Обозначения: SB: бит старта, STB: бит стопа, PB: бит чётности

Бит PS определяет число передаваемых "1" с учётом 7 или 8 младших битов (зависит от бита M), включая сам бит чётности.

Чётность:

Пример: данное=00110101; 4 бита стоят => (бит USART_CR1/PS = 0).

Нечётность:

Пример: данное=00110101; 4 бита стоят => (бит USART_CR1/PS = 1).

Проверка чётности на приёме

При сбое чётности встаёт флаг USART_SR/PE, бит USART_CR1/PEIE разрешает прерывание. Бит PE снимается чтением регистра состояния с последующим обращением к регистру USART_DR.

NB: При пробуждении по метке адреса чётность не проверяется и флаг PE не встаёт.

Формирование чётности при передаче

При стоящем бите USART_CR1/PCE старший передаваемый бит изменяется в зависимости бита PS для передачи чётного или нечётного числа "1". При сбое чётности ставится флаг PE регистра USART_SR. Прерывание разрешается битом USART_CR1/PEIE.

NB: Бит PE снимается чтением регистра состояния с последующим обращением к регистру USART_DR. В полудуплексе это иногда может не вовремя снять флаг PE.

30.3.8. Коммутируемая локальная сеть LIN

Режим LIN включается установкой бита USART_CR2/LINEN. При этом должны быть очищены биты:

- STOP[1:0], CLKEN в регистре USART_CR2
- SCEN, HDSEL и IREN в регистре USART_CR3.

Передача LIN

Для передачи ведущего LIN используется процедура из *Секции 30.3.2*. со следующими отличиями:

- Снимается бит длины слова M (8-бит).
- Ставится бит LINEN режима LIN. Теперь установка бита SBK в качестве символа разрыва посылает 13 нулей с последующей посылкой ‘1’ для обнаружения грядущего бита старт.

Приём LIN

Схема обнаружения разрыва реализована отдельно и срабатывает и при простое шины и при передаче фрейма.

При включённом приёмнике (USART_CR1/RE=1) схема ждёт сигнал старта на ножке RX. Метод обнаружения всё тот же. После обнаружения старта, следующие биты считываются как биты данных (по 8, 9 и 10 импульсам выборки). Если получены десять (при USART_CR2/LBDL = 0) или одиннадцать (при USART_CR2/LBDL=1) последовательных нулей с последующим разделителем, то ставится флаг USART_SR/LBD. Прерывание разрешается битом LBDIE=1.

Для обнаружения разрыва линия RX должна вернуться в высокий уровень. При появлении ‘1’ перед 10 или 11 схема обнаружения разрыва сразу переключается назад в поиск бита старта.

При выключенном LIN (LINEN=0), приёмник, как нормальный USART, символ разрыва не ищет.

При включённом LIN (LINEN=1), После ошибки фрейма (нулевой бит стопа, при фрейме разрыва), приёмник останавливается либо до получения схемой разрыва '1’ при незавершенном слове разрыва, либо символа разделителя при завершённом слове разрыва.

Рис. 305. Обнаружение разрыва в режиме LIN (11-бит разрыв - бит LBDL стоит)

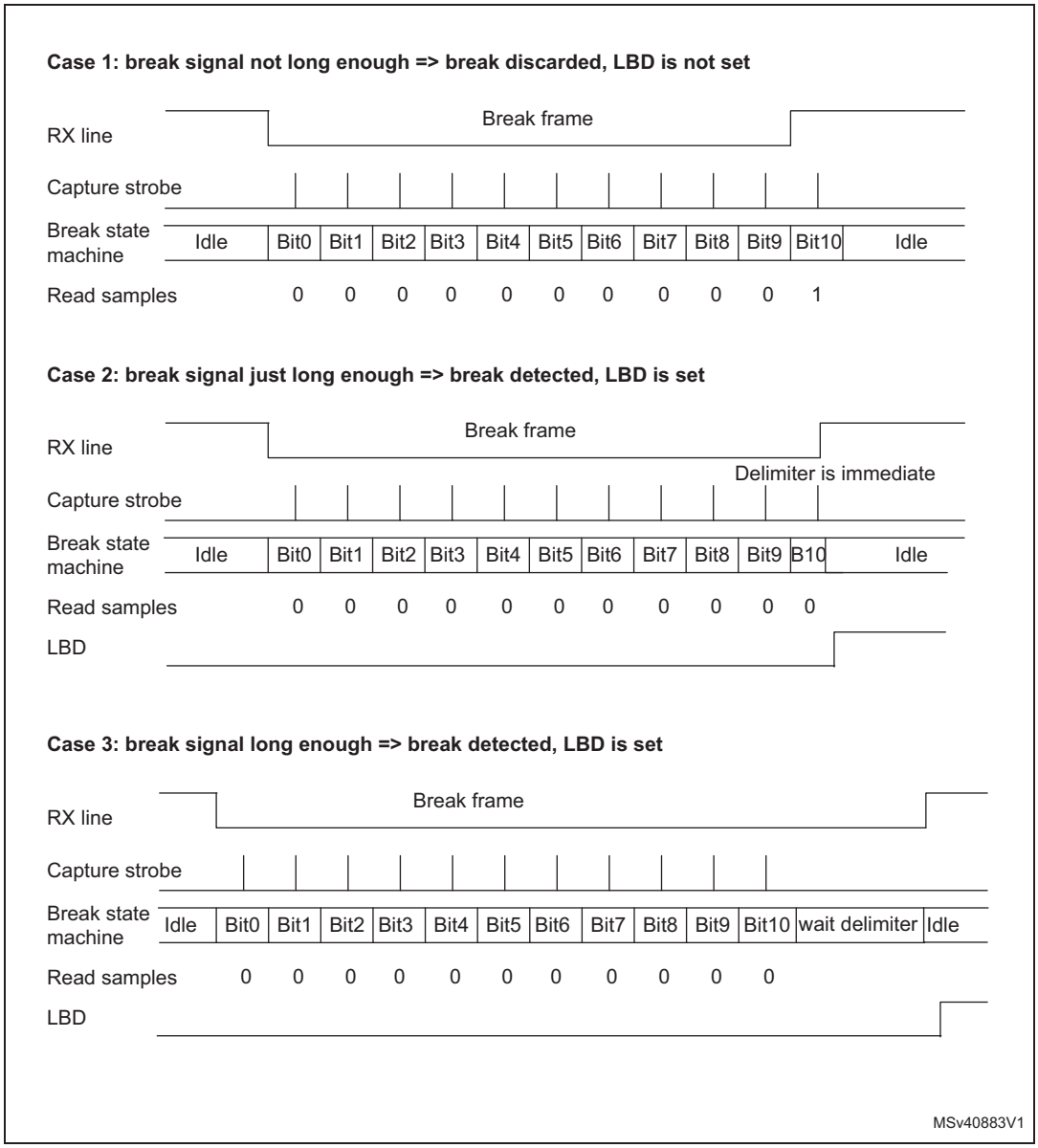
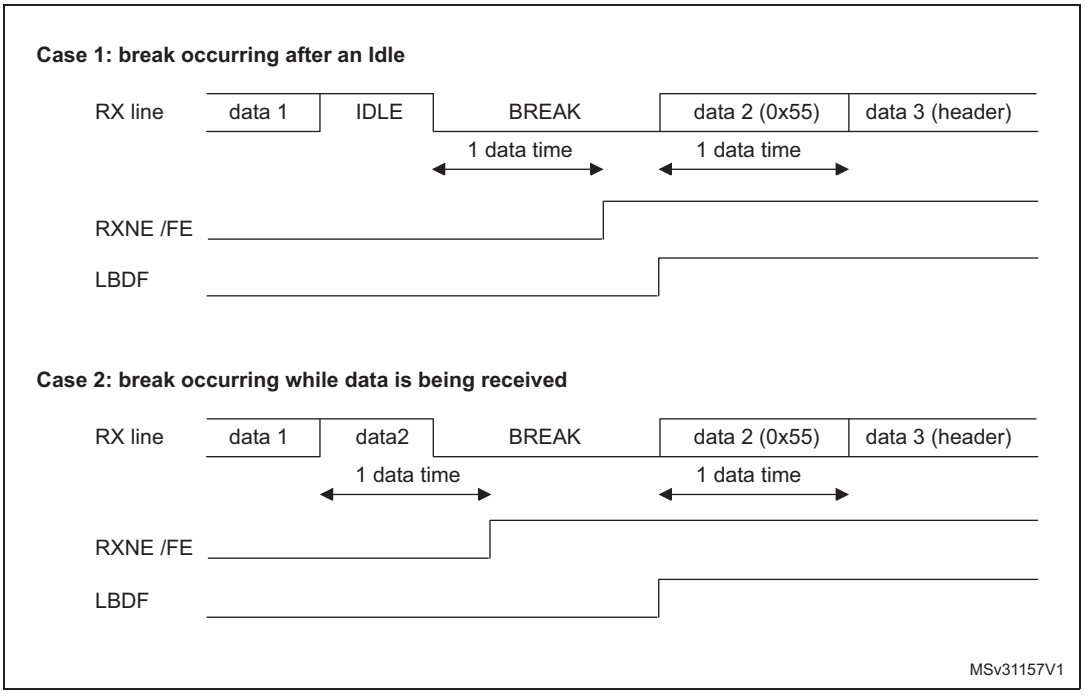


Рис. 305. Обнаружение разрыва в режиме LIN и Ошибка фрейма



30.3.9. Синхронный режим USART

Включается установкой бита USART_CR2/CLKEN. При этом надо снять биты:

- LINEN в регистре USART_CR2,
- SCEN, HDSEL и IREN в регистре USART_CR3.

USART может работать ведущим в двунаправленном синхронном режиме. Ножка CK выдаёт такты передатчика USART. Во время битов старта и стопа тактовые импульсы не выдаются. Бит USART_CR2/LBCL разрешает выдачу тактов во время передачи последнего действительного бита данных (метка адреса). Бит USART_CR2/CPOL задаёт полярность, а бит USART_CR2/CPHA задаёт полярность внешних тактов.

Во время Простоя, Преамбулы и посылки разрыва внешних тактов CK нет.

Синхронный передатчик USART работает так же как асинхронный, но CK и TX синхронизируются (в соответствии с CPOL CPHA).

Работа синхронного приёмника USART отличается от асинхронного. При RE=1, данные считываются по фронту CK (см. CPOL и CPHA), а не множественной выборкой. Времена установки и удержания должны быть соответствующими (1/16 времени бита).

NB: Возможно принимать только данные включённого передатчика.

Ради нормальной работы тактов, биты LBCL, CPOL и CPHA пишутся только у выключенных приёмнике и передатчике (TE=RE=0), у работающих блоков их изменять нельзя.

Для минимизации времён установки и удержания, биты TE и RE лучше всего ставить одной командой.

USART работает только ведущим (CK всегда вывод).

Рис. 307. Пример синхронной передачи USART

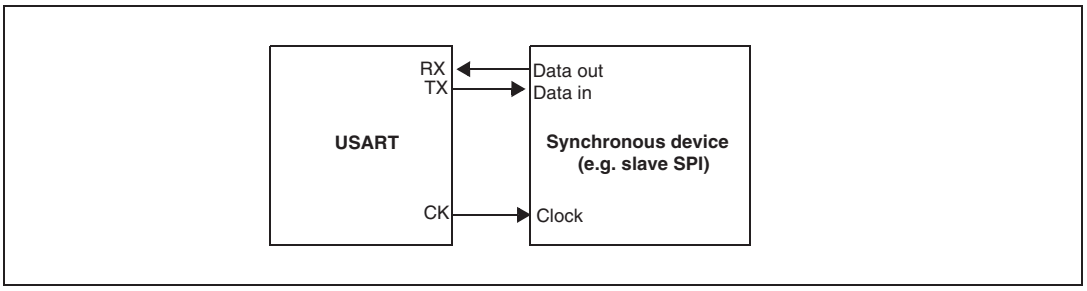


Рис. 308. Временная диаграмма тактов USART (M=0)

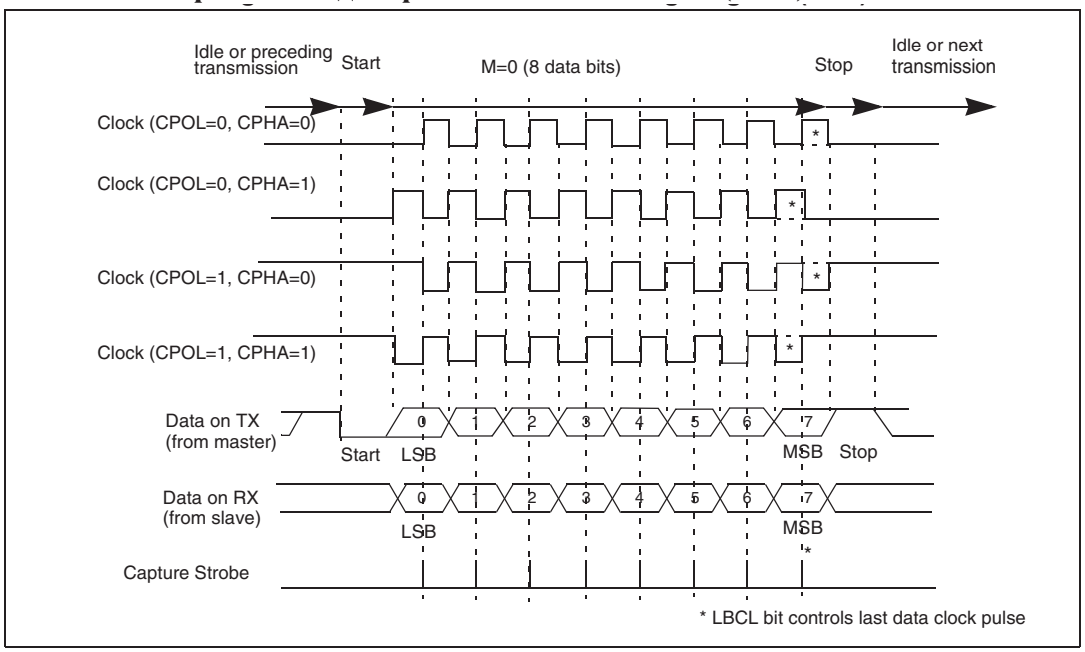


Рис. 309. Временная диаграмма тактов USART (M=1)

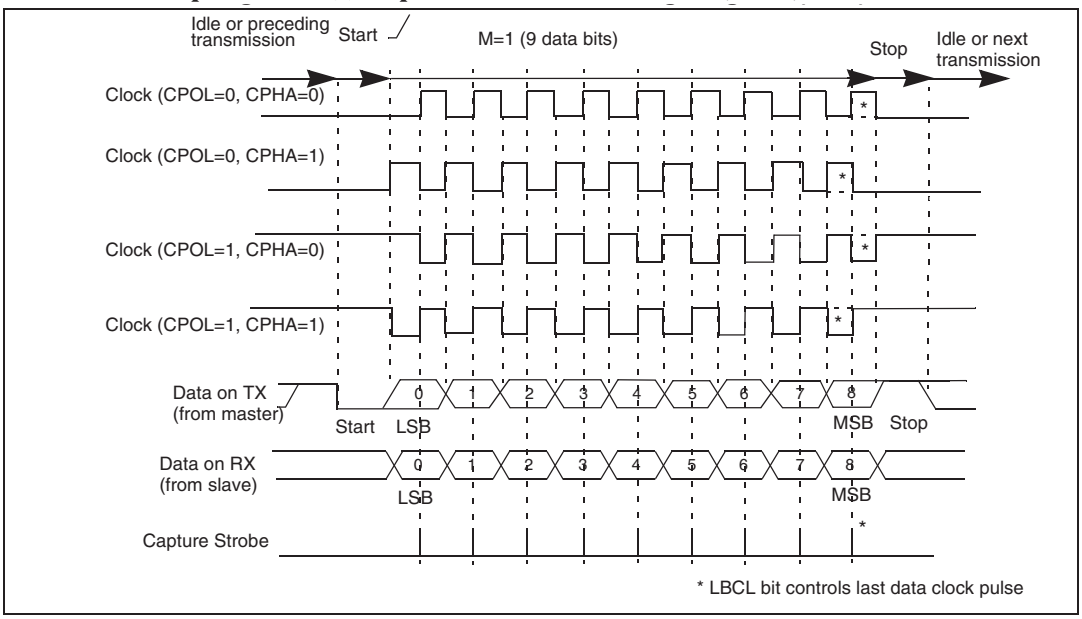
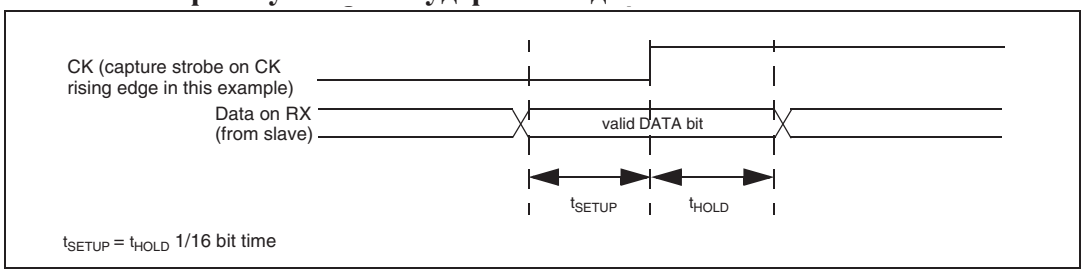


Рис. 310. Время установки/удержания данных RX



В режиме Smartcard ножка CK работает по-другому.

30.3.10.Однопроводной полудуплекс

Включается установкой бита HDSEL в регистре USART_CR3. При этом должны быть сняты биты:

- LINEN и CLKEN в регистре USART_CR2,
- SCEN и IREN в регистре USART_CR3.

В этом режиме ножки TX и RX замыкаются внутри. Полудуплекс или полный дуплекс выбирается битом ‘HALF DUPLEX SEL’ (HDSEL в USART_CR3).

При стоящем HDSEL:

- ножки TX и RX замкнуты внутри
- RX не используется,
- Не передающая ножка TX всегда свободна, прямо как стандартный свободный или приёмный IO. То есть ножка IO для не передающей TX должна быть плавающим входом (открытый сток при выводе).

Всё остальное соответствует нормальному режиму USART. Конфликты на шине разрешаются программно. В частности, передача аппаратно не блокируется и продолжается сразу после записи регистра данных при стоящем бите TE.

30.3.11.Режим Smartcard

Включается установкой бита SCEN в регистре USART_CR3 При этом должны быть сняты биты:

- LINEN в регистре USART_CR2,
- HDSEL и IREN в регистре USART_CR3.

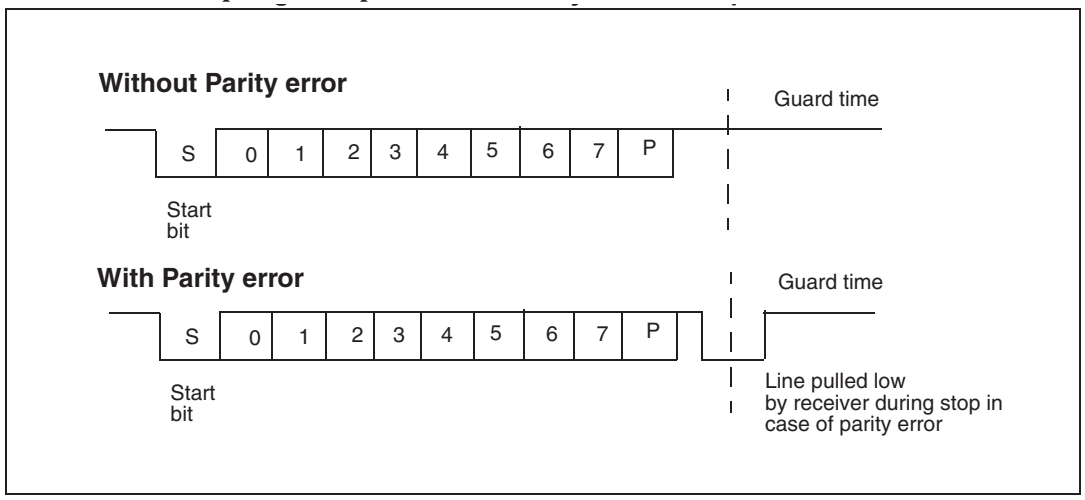
Более того, бит CLKEN можно ставить для подачи тактов на карту.

Интерфейс Smartcard определён стандартом ISO 7816-3. USART конфигурируется так:

- 8 бит и чётность: ставим M=1 и PCE=1 в регистре USART_CR1
- 1.5 бита стоп приёма и передачи (STOP=’11’ в регистре USART_CR2).

Можно выбрать и 0.5 бита стоп для приёма, но тогда надо переключаться.

Рис. 311. Асинхронный протокол ISO 7816-3



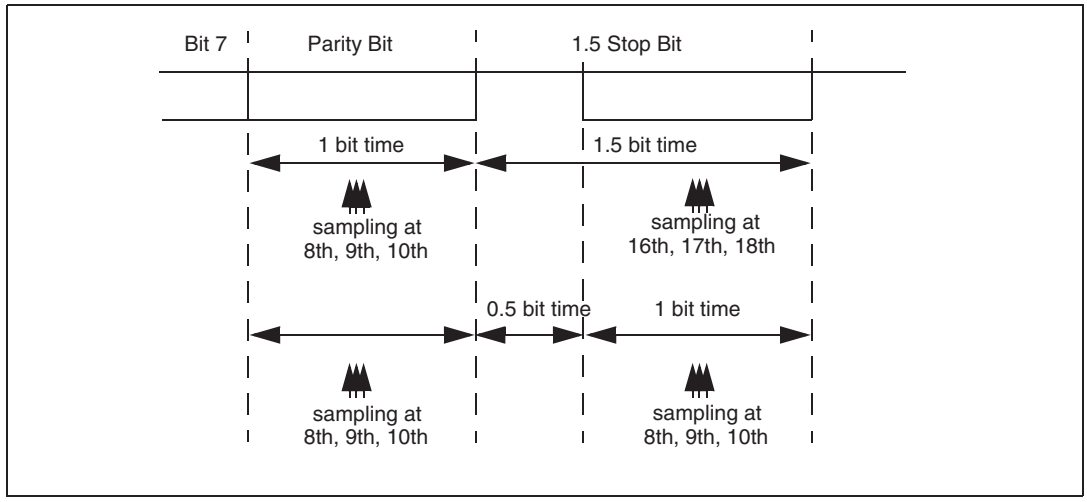
Двунаправленная линия TX управляется и USART и картой, так что должна конфигурироваться с открытым стоком.

Smartcard это однопроводной полудуплексный протокол.

- Передача данных из регистра сдвига гарантированно задерживается на минимум 1/2 такта связи. При нормальной работе регистр сдвигается по следующему фронту такта связи, в режиме Smartcard эта передача ещё задерживается на 1/2 такта связи.
- При ошибке чётности приёма с 0.5 или 1.5 бита стоп после завершения приёма фрейма, линия TX уводится вниз на один такт связи (сигнал `NACK`), показывая карте, что данные приняты с ошибкой. На передающей стороне возникнет ошибка фрейма (у ней 1.5 бита стоп). Программа может послать данные повторно. Ошибка чётности отправляется приемником только при стоящем бите `NACK`.
- В режиме Smartcard установка бита `TC` может быть задержана относительно нормальной работы с помощью регистра защиты (Guard Time). В этом режиме опустение регистра сдвига запускает счётчик защиты, он считает до значения регистра защиты, удерживая `TC` низким. После этого `TC` встаёт.
- Режим Smartcard на снятие флага `TC` не влияет.
- Сигнал `NACK` от приёмника на передающей стороне не воспринимается как бит старта. Длительность принятого `NACK` может быть 1 или 2 такта связи.
- На приёмной стороне возвращаемый `NACK` не считается битом старта.

NB: В режиме Smartcard передача `0x00` с ошибкой фрейма не является символом разрыва, его тут нет. При переключении бита `TE` фрейм `IDLE` не передаётся.

Рис. 312. Обнаружение ошибки чётности при 1.5 битах стоп



По ноге CK USART может выдавать карте такты. Они не связаны с передачей данных, а просто так получаются из входных тактов f_{CK} устройства делением на значение регистра. `USART_GTPR`. CK может быть от $f_{CK}/2$ до $f_{CK}/62$.

30.3.12.Блок IrDA SIR ENDEC

Режим IrDA включается установкой бита IREN регистра USART_CR3. Чистить надо биты:

- LINEN, STOP и CLKEN в регистре USART_CR2,
- SCEN и HDSEL в регистре USART_CR3.

Физический уровень IrDA SIR использует модуляцию RZI, логический 0 это световой импульс.

Кодер передатчика SIR получает NRZ-модулированный поток битов от USART. Для SIR ENDEC поддерживается только скорость 115.2 Кбит/с. В нормальном режиме ширина импульса равна 3/16 периода бита.

Декодер приёмника SIR демодулирует RZ поток битов от детектора излучения в NRZ поток битов для USART. Вход декодера без сигнала обычно высокий, выход кодера низкий. Бит старта на входе декодера низкий.

- IrDA это полудуплексный протокол. Если передатчик занят (USART посылает данные кодеру IrDA), то данные на линии IrDA декодером игнорируются. Если занят приёмник, то данные на TX от USART в IrDA не кодируются. При приёме данных надо избегать передачи.
- A‘0’ передаётся высоким импульсом, а ‘1’ передаётся как ‘0’. Ширина импульса в нормальном режиме равна 3/16 периода бита.
- Декодер SIR преобразует приёмный сигнал IrDA в битовый поток USART.
- Приёмная логика SIR воспринимает высокий уровень как логическую единицу, а низкий уровень как ноль.
- The transmit encoder output has the opposite polarity to the decoder input. The SIR output is in low state when idle.
- Спецификация IrDA требует приёма импульса длиннее 1.41 us. Это программируемый параметр. Логика определения глитчей отфильтровывает импульсы короче 2 периодов PSC (PSC это значение предделителя IrDA в регистре USART_GTPR). Импульсы короче 1 периода PSC всегда отбрасываются, короче 2 периодов могут приниматься, длиннее 2 периодов принимаются как импульсы. При PSC=0 кодер/декодер IrDA не работают.
- Приёмник может работать с экономным передатчиком.
- В режиме IrDA используется один бит стопа.

Экономный режим IrDA

Передатчик

В этом режиме ширина импульса не равна 3/16 периода бита. Она равна трём тактам частоты экономного режима, обычно 1.8432 MHz (1.42 MHz < PSC < 2.12 MHz). Для этой частоты есть программируемый предделитель экономного режима.

Приёмник

Это похоже на приём нормального режима. Детектор глитчей USART отбрасывает импульсы короче 1/PSC. Принимаются импульсы длиннее 2 периодов тактов IrDA экономного режима (значение PSC в USART_GTPR).

NB: Импульс длиннее 1 и короче 2 периодов PSC могут приниматься, а могут и не приниматься.

Время установки приёмника управляется программно. Физический уровень IrDA должен иметь минимальную задержку 10 ms между приёмом и передачей.

Рис. 313. Блок-схема IrDA SIR ENDEC

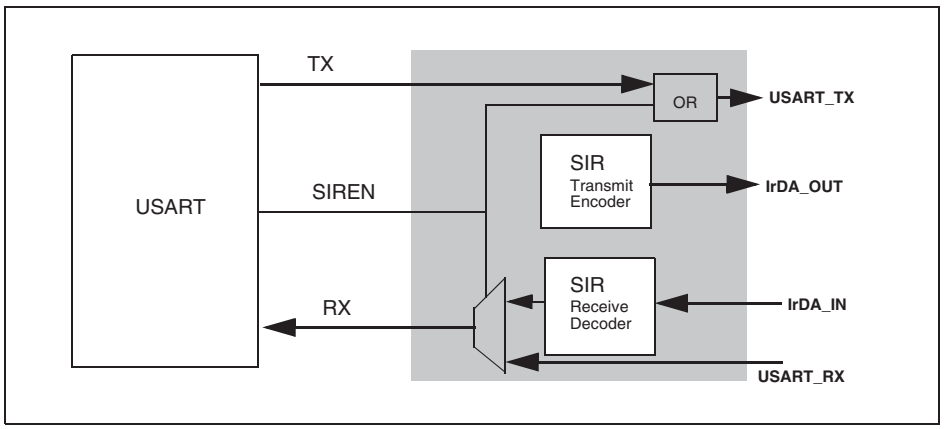
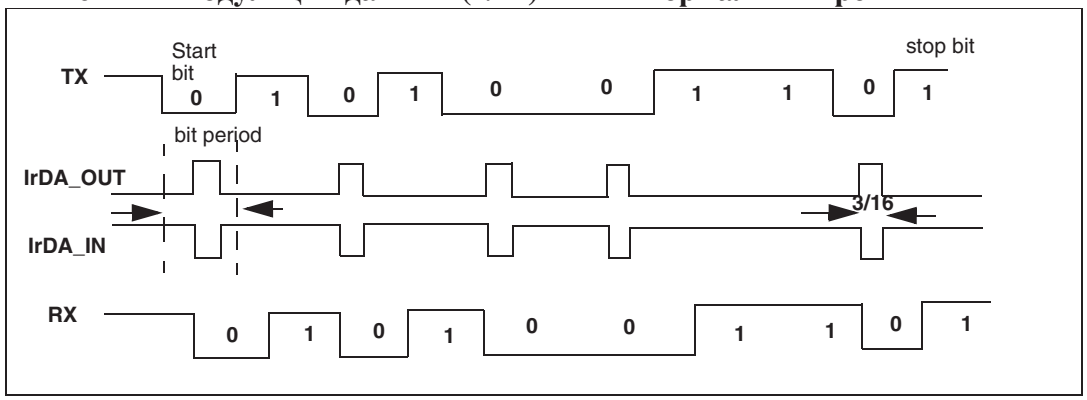


Рис. 314. Модуляция данных (3/16) IrDA - нормальный режим



30.3.13.Непрерывная связь с DMA

Для непрерывных передач нужно вовремя программно снимать биты TXE/ RXNE, а DMA снимает их сам.

Передача с DMA

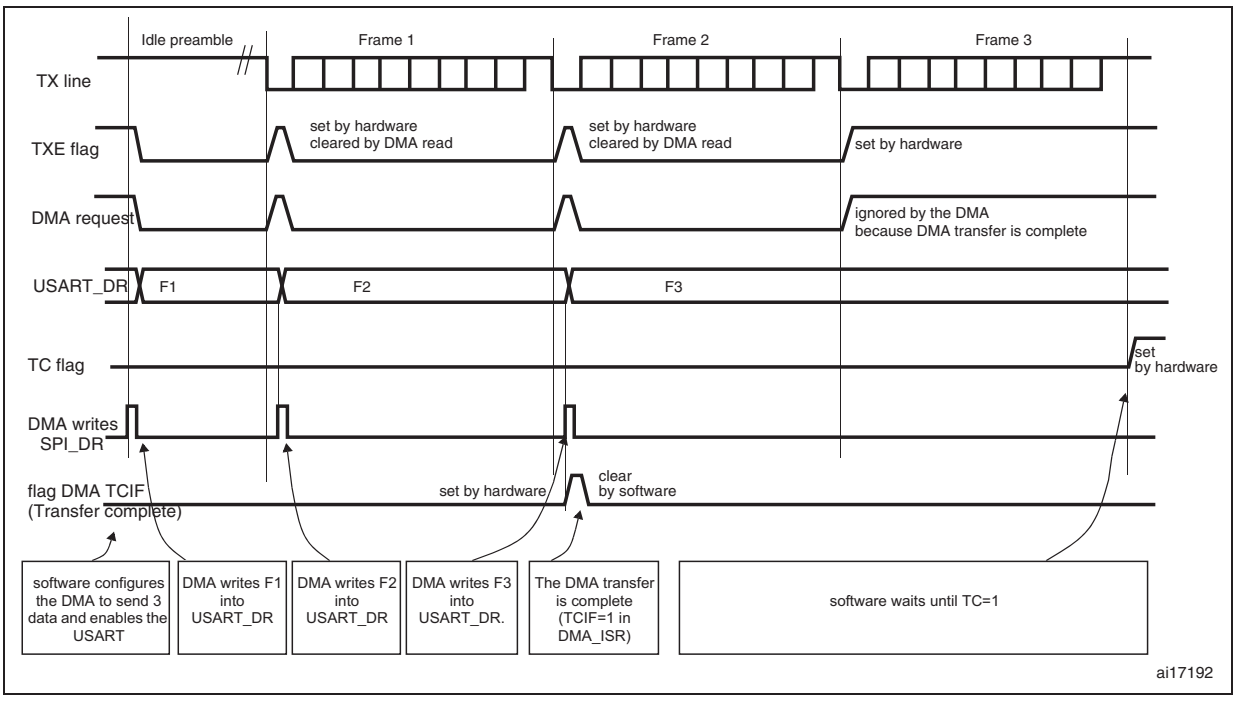
Включается установкой бита USART_CR3/DMAT. Данные пишутся из SRAM в регистр USART_DR по каждой установке бита TXE. Канал DMA назначается USART процедурой:

1. Пишем адрес регистра USART_DR в регистр адреса получателя DMA.
2. Пишем адрес памяти в регистр адреса источника DMA.
3. Пишем общее число передаваемых байтов в регистр управления DMA.
4. Ставим приоритет канала DMA
5. Определяем выдачу прерываний DMA после половины/всей передачи.
6. Снимаем бит TC регистра SR записью 0.
7. Активируем канал DMA.

После передачи заданного числа байтов контроллер DMA выдаёт прерывание по своему вектору.

После передачи DMA всех заданных байтов (флаг TCIF в регистре DMA_ISR), конец операции USART можно отследить по флагу TC. Для выключения USART или останова MCU нужно программно дождаться TC=1. Флаг TC аппаратно ставится по концу последнего фрейма передачи.

Рис. 315. Передача с DMA



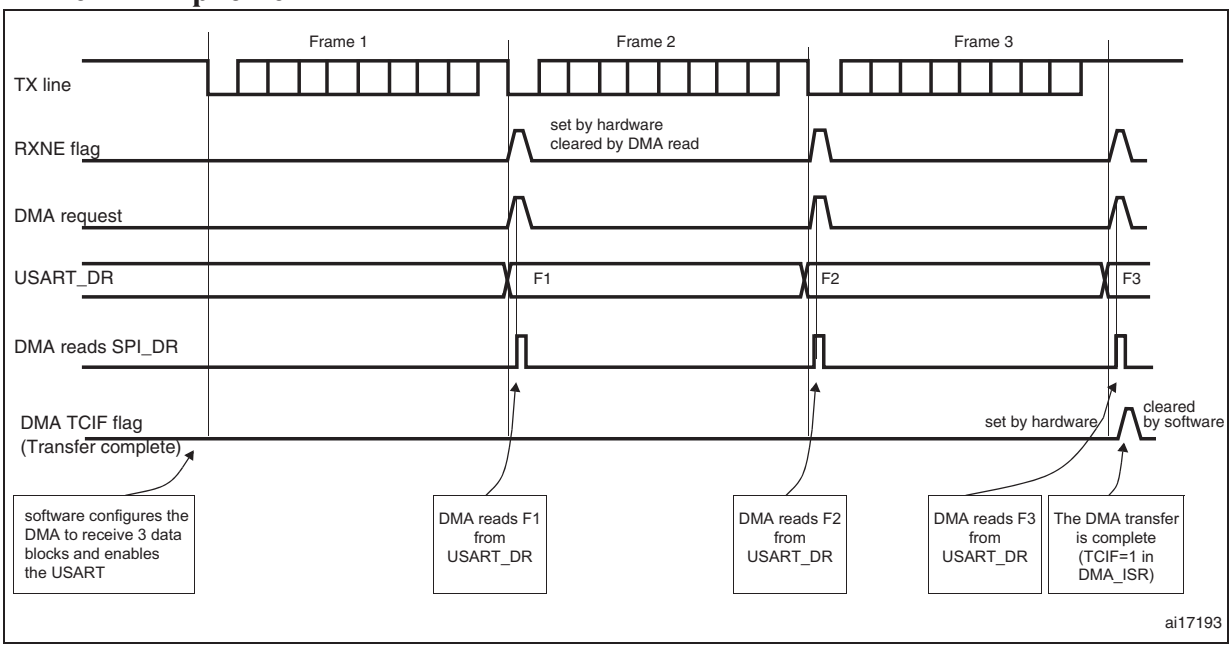
Приём с DMA

Включается установкой бита USART_CR3/DMAR. Данные пишутся из USART_DR в SRAM. Канал DMA назначается USART процедурой:

1. Пишем адрес регистра USART_DR в регистр адреса источника DMA.
2. Пишем адрес памяти в регистр адреса получателя DMA.
3. Пишем общее число передаваемых байтоа в регистр управления DMA.
4. Ставим приоритет канала DMA
5. Определяем выдачу прерываний DMA после половины/всей передачи.
6. Снимаем бит TC регистра SR записью 0.
7. Активируем канал DMA.

После передачи заданного числа байтов контроллер DMA выдаёт прерывание по своему вектору.

Рис. 316. Приём с DMA



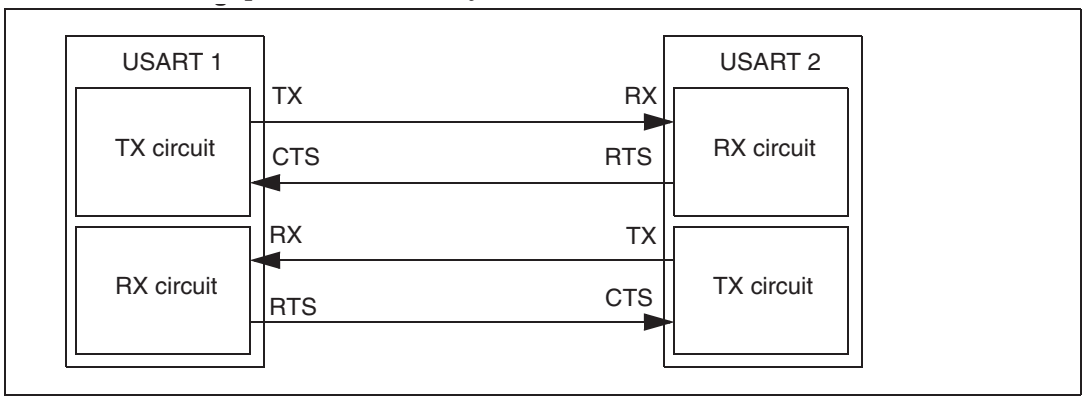
Флаги ошибок и прерывания при многобуферных передачах

При многобуферных передачах флаги ошибок ставятся после сбойного байта. Прерывание нужно разрешать. Ошибка фрейма, переполнение и флаг шума, которые при передаче одного байта ставятся вместе с RXNE, есть отдельный общий флаг разрешения прерывания ошибки (бит EIE регистра USART_CR3).

30.3.14.Аппаратное управление потоком

Для передач между двумя USART можно использовать их входы CTS и выходы RTS.

Рис. 317. Аппаратная связь двух USART

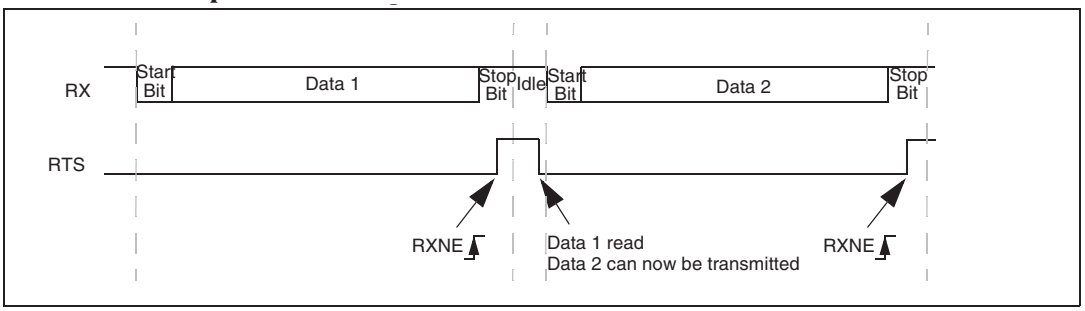


RTS и CTS включаются независимо установкой битов RTSE и CTSE регистра USART_CR3.

Управление RTS

При RTSE=1 нога RTS держится низкой во время готовности USART принять новое данное. При заполнении регистра сдвига, RTS снимается, извещая, что передатчику пора стопорить.

Рис. 318. Управление потоком RTS

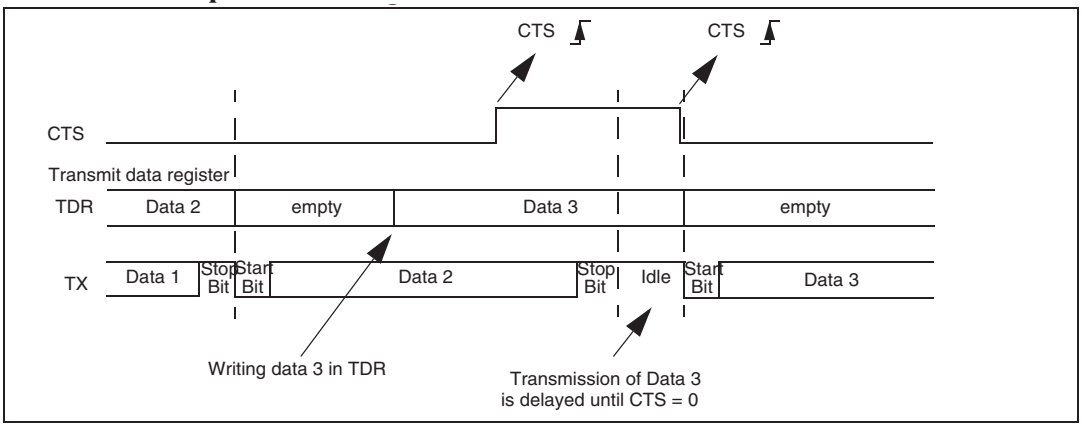


Управление CTS

При CTSE=1 передатчик перед передачей данных ждёт низкого входа CTS. Если CTS снимается во время передачи, то она прекращается.

При CTSE=1 флаг CTSIF аппаратно ставится при переключении входа CTS, показывая готовность и неготовность приёмника. Прерывание разрешается битом CTSIE регистра USART_CR3.

Рис. 319. Управление потоком CTS



NB: При включённом потоке CTS передатчик не проверяет состояние входа CTS на разрыв.

30.4. Прерывания USART

Таблица 147. Запросы прерываний USART

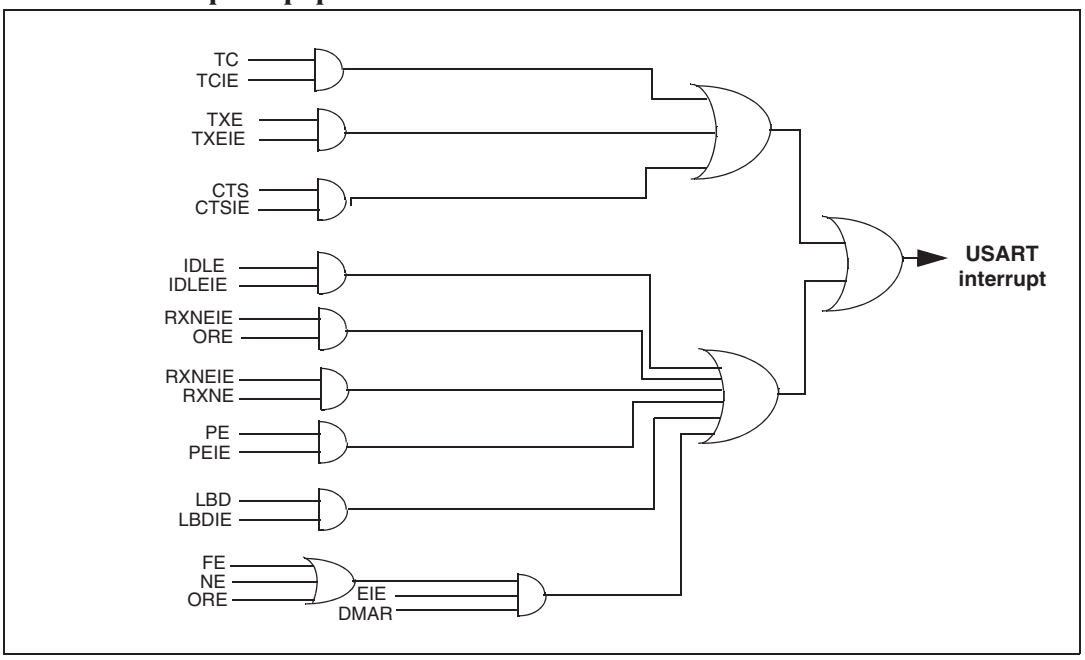
Прерывание	Флаг	Бит разрешения
Регистр данных передатчика пуст	TXE	TXEIE
Флаг CTS	CTS	CTSIE
Передача завершена	TC	TCIE
Принятые данные можно читать	RXNE	RXNEIE
Переполнение	ORE	
Простой линии	IDLE	IDLEIE
Ошибка чётности	PE	PEIE
Флаг разрыва	LBD	LBDIE
Флаг Шума, Переполнения и Ошибки фрейма при многобуферной связи	NE или ORE или FE	EIE(1)

1. Используется только при передачах с DMA.

Все прерывания USART объединены в один вектор.

- При передаче: Передача завершена, Могу Передавать или Регистр Данных Пуст.
- При приёме: Простой линии, Переполнение, Регистр Данных не Пуст, Ошибка чётности, Разрыв LIN, Флаг Шума (только многобуферный) и Ошибка Фрейма (только многобуферный).

Рис. 320. Карта прерываний USART



30.5. Режимы USART

Таблица 148. Режимы USART(1)

Режим USART	USART1	USART2	USART3	UART4	UART5
Асинхронный	X	X	X	X	X
Аппаратное управление режимом	X	X	X	NA	NA
Многобуферная связь (DMA)	X	X	X	X	NA
Многопроцессорная связь	X	X	X	X	X
Синхронная связь	X	X	X	NA	NA
Smartcard	X	X	X	NA	NA
Полудуплекс (Однопроводной)	X	X	X	X	X
IrDA	X	X	X	X	X
LIN	X	X	X	X	X

1. X = поддерживается; NA = не применим.

30.6. Регистры USART

Регистры доступны полусловами (16-бит) и словами (32-бит).

30.6.1. Регистр состояния (USART_SR)

Смещение адреса: 0x00

По сбросу: 0x0000 00C0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CTS	LBD	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
						rc_w0	rc_w0	r	rc_w0	rc_w0	r	r	r	r	r

— Биты 31:10 Резерв, не трогать.

— Бит 9 **CTS**: Флаг CTS

Ставится аппаратно при переключении входа CTS и стоящем CTSE. Снимается записью 0. Прерывание выдаётся при CTSIE=1 в регистре USART_CR3

0: Линия CTS не менялась

1: Линия CTS менялась

Бит недоступен для UART4 и UART5.

— Бит 8 **LBD**: Флаг разрыва LIN

Ставится аппаратно при обнаружении разрыва LIN. Снимается записью 0. Прерывание выдаётся при LBDIE = 1 в регистре USART_CR2.

0: Разрыва LIN не было

1: Был разрыв LIN

— Бит 7 **TXE**: Регистр данных передатчика пуст

Ставится аппаратно после передачи TDR в регистр сдвига. Прерывание выдаётся при TXEIE=1 в регистре USART_CR1. Снимается записью в USART_DR.

0: Регистр данных занят

1: Регистр данных пуст

NB: Используется при однобуферной передаче.

— Бит 6 **TC**: Передача завершена

Ставится аппаратно после завершения передачи фрейма при стоящем TXE. Прерывание выдаётся при TCIE=1 в регистре USART_CR1. Снимается чтением из USART_SR и затем записью в USART_DR. В многобуферном случае рекомендуется снимать записью '0'.

0: Передача не завершена

1: Передача завершена

— Бит 5 **RXNE**: Регистр данных приёмника не пуст

Ставится аппаратно после передачи регистра сдвига в USART_DR. Прерывание выдаётся при RXNEIE=1 в регистре USART_CR1. Снимается чтением из USART_DR. В многобуферном случае рекомендуется снимать записью '0'.

0: Данные не приняты.

1: Данные можно читать.

— Бит 4 **IDLE**: Простой линии

Ставится аппаратно при обнаружении Простоя линии. Прерывание выдаётся при IDLEIE=1 в регистре USART_CR1. Снимается чтением из USART_SR и затем чтением USART_DR.

0: Простоя не было

1: Символ простоя был

NB: Снова бит IDLE не встанет пока не встанет бит RXNE (был новый простой).

— Бит 3 **ORE**: Переполнение

Ставится аппаратно после приёма в регистр сдвига при ещё нечитаном RDR пока RXNE=1. Прерывание выдаётся при RXNEIE=1 в регистре USART_CR1. Снимается чтением из USART_SR и затем чтением USART_DR.

0: Не было

1: Переполнилось

NB: Если бит стоит, то содержимое RDR, но заменяется содержимое регистра сдвига. Прерывание выдаётся при стоящем бите EIE в многобуферном случае.

— Бит 2 **NE**: Флаг Шума

Ставится аппаратно при обнаружении шума в принятом фрейме. Снимается чтением из USART_SR и затем чтением USART_DR.

0: Не было

1: Нашумело

NB: Прерывание по этому биту выдаётся при стоящем бите EIE в многобуферном случае, а обычно прерывается по RXNE.

— Бит 1 **FE**: Ошибка фрейма

Ставится аппаратно при потере синхронизации, излишнем шуме или символе разрыва. Снимается чтением из USART_SR и затем чтением USART_DR.

0: Не было

1: Была ошибка или разрыв

NB: Если ошибка фрейма и переполнение возникают одновременно, то фрейм передаётся и ставится только бит ORE.

Прерывание по этому биту выдаётся при стоящем бите EIE в многобуферном случае, а обычно прерывается по RXNE..

— Бит 0 **PE**: Ошибка чётности

Ставится аппаратно при ошибке чётности на приёме. Снимается чтением из USART_SR и затем чтением USART_DR. Перед очисткой надо дождаться флага RXNE.

Прерывание выдаётся при PEIE = 1 в регистре USART_CR1.

0: Ошибки нет

1: Ошибка чётности

30.6.2. Регистр данных (USART_DR)

Смещение адреса: 0x04

По сбросу: Всякая фигня.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							DR[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:9 Резерв, не трогать.

— Биты 8:0 **DR[8:0]**: Данные приёма (RDR) и передачи (TDR)

При включённом контроле чётности старший передаваемый бит заменяется битом чётности.

30.6.3. Регистр скорости (USART_BRR)

NB: Соответствующий счётчик скорости останавливается при снятом TE или RE.

Смещение адреса: 0x08

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIV_Mantissa[11:0]												DIV_Fraction[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:9 Резерв, не трогать.

— Биты 15:4 **DIV_Mantissa[11:0]**: мантисса USARTDIV

— Биты 3:0 **DIV_Fraction[3:0]**: дробная часть USARTDIV

30.6.4. Регистр 1 управления (USART_CR1)

Смещение адреса: 0x0C

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	Reserved	UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
rw	Res.	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:16 Резерв, не трогать.

— Бит 15 **OVER8**: Режим выборки

0: по 16

1: по 8

NB: Выборки по 8 нет в режимах Smartcard, IrDA и LIN: при SCEN=1,IREN=1 или LINEN=1 бит OVER8 аппаратно сбрасывается

— Бит 14 Резерв, не трогать.

— Бит 13 **UE**: Включение USART

Ставится и снимается программно.

0: Предделитель и выходы USART отключаются после завершения передачи байта

1: USART работает

— Бит 12 **M**: Длина слова

Ставится и снимается программно.

0: 1 бит Старт, 8 битов данных, n битов Стоп

1: 1 бит Старт, 9 битов данных, n битов Стоп

NB: Нельзя изменять во время чтения или записи

— Бит 11 **WAKE**: Способ побудки

Ставится и снимается программно.

0: Простой линии

1: Метка адреса

— Бит 10 **PCE**: Включение контроля чётности

Ставится и снимается программно. При включённом контроле чётности старший передаваемый бит (9 при M=1, 8 при M=0) заменяется битом чётности. На передаваемый в момент установки байт не влияет.

0: Выключен

1: Включён

— Бит 9 **PS**: Выбор чётности

Ставится и снимается программно. На передаваемый в момент установки байт не влияет.

0: Чётность

1: Нечётность

— Бит 8 **PEIE**: Разрешение прерывания PE

Ставится и снимается программно.

0: Нельзя

1: Можно

— Бит 7 **TXEIE**: Разрешение прерывания TXE

Ставится и снимается программно.

0: Нельзя

1: Можно

— Бит 6 **TCIE**: Разрешение прерывания TC конца передачи

Ставится и снимается программно.

0: Нельзя

1: Можно

— Бит 5 **RXNEIE**: Разрешение прерывания RXNE и ORE

Ставится и снимается программно.

0: Нельзя

1: Можно

— Бит 4 **IDLEIE**: Разрешение прерывания IDLE

Ставится и снимается программно.

0: Нельзя

1: Можно

— Бит 3 **TE**: Включение передатчика

Ставится и снимается программно.

0: Выключен

1: Включён

NB:

1: Установка TE посылает на линию преамбулу (Простой линии) после текущего слова, кроме режима Smartcard.

2: При стоящем TE перед началом передачи появляется задержка на 1 бит.

— Бит 2 **RE**: Разрешение приёмника

Ставится и снимается программно.

0: Выключен

1: Включён, начинает искать бит старта

— Бит 1 **RWU**: Побудка приёмника

Задаёт немой режим приемника. Ставится и снимается программно, может сняться аппаратно при обнаружении последовательности побудки.

0: Приёмник активен

1: Приёмник нем

NB:

1: Перед установкой бита RWU USART должен сначала принять байт данных, иначе не сможет проснуться по Простою линии.

2: При побудке по Метке адреса (WAKE bit=1) бит RWU нельзя менять при стоящем с RXNE.

— Бит 0 **SBK**: Посылка символа разрыва

Ставится программно, снимается аппаратно во время бита стоп разрыва.

0: Ничего

1: Будет передан символ разрыва

30.6.5. Регистр 2 управления (USART_CR2)

Смещение адреса: 0x10

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LINEN	STOP[1:0]		CLK EN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	Res.	ADD[3:0]			
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

— Биты 31:15 Резерв, не трогать.

— Бит 14 **LINEN**: Включение режима LIN

Ставится и снимается программно.

0: Выключен

1: Включён, можно обнаруживать и посылать символ разрыва(13 низких битов).

— Биты 13:12 **STOP**: Число битов STOP

00: 1 бит

01: 0.5 бита

10: 2 бита

11: 1.5 бита

The 0.5 и 1.5 бита в UART4 и UART5 нет.

— Бит 11 **CLKEN**: Включение тактов на ноге CK

0: Выключено

1: Включено

Нету в UART4 и UART5.

— Бит 10 **CPOL**: Полярность тактов CK в синхронном режиме

Работает вместе с битом CPHA

0: Ножка CK спокойна низким вне окна передачи.

1: Ножка CK спокойна высоким вне окна передачи.

Нету в UART4 и UART5.

— Бит 9 **CPHA**: Фаза тактов CK в синхронном режиме

Работает вместе с битом CPOL

0: Фронт считывания первого данного в первом такте.

1: Фронт считывания первого данного во втором такте.

Нету в UART4 и UART5.

— Бит 8 **LBCL**: Тактовый импульс последнего бита в синхронном режиме

0: На ноге CK его нет

1: На ноге CK он есть

Нету в UART4 и UART5.

— Бит 7 Резерв, не трогать.

— Бит 6 **LBDIE**: Разрешение прерывания разрыва LIN

0: Нельзя

1: Можно

— Бит 5 **LBDL**: Длина определения разрыва

0: За 10 битов

1: За 11 битов

— Бит 4 Резерв, не трогать.

— Биты 3:0 **ADD[3:0]**: Адрес узла USART в многопроцессорной системе

Биты CPOL, CPHA, LBCL нельзя писать при включённом передатчике.

30.6.6. Регистр 3 управления (USART_CR3)

Смещение адреса: 0x14

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:12 Резерв, не трогать.

— Бит 11 **ONEBIT**: Выборка одного бита

При стоящем бите флаг обнаружения шума (NF) выключатся.

0: Три выборки

1: Одна выборка

NB: Применяется только к битам данных, но не к биту START.

— Бит 10 **CTSIE**: Разрешение прерывания CTS

0: Нельзя

1: Можно

Нету в UART4 и UART5.

— Бит 9 **CTSE**: Разрешение аппаратного управления CTS

0: Выключено

1: Включено, данные передаются только при низком CTS. Выключение CTS во время передачи прекращает её. Данные, записанные в DR при снятом CTS, передаются после его установки.

Нету в UART4 и UART5.

— Бит 8 **RTSE**: Разрешение аппаратного управления RTS

0: Выключено

1: Включено, данные запрашиваются только при пустом буфере (RTS низкий).

Нету в UART4 и UART5.

— Бит 7 **DMAT**: Разрешение использования DMA передатчика

Ставится и снимается программно.

0: Нельзя

1: Можно

Нету в UART5.

— Бит 6 **DMAR**: Разрешение использования DMA приёмника

Ставится и снимается программно.

0: Нельзя

1: Можно

Нету в UART5.

— Бит 5 **SCEN**: Разрешение режима Smartcard

0: Smartcard выключен

1: Smartcard включён

Нету в UART4 и UART5.

— Бит 4 **NACK**: Разрешение выдачи NACK Smartcard при ошибке чётности

0: Нельзя

1: Можно

Нету в UART4 и UART5.

— Бит 3 **HDSEL**: Выбор полудуплекса или полного дуплекса

0: Полный дуплекс

1: Полудуплекс

— Бит 2 **IRLP**: Экономный режим IrDA

0: Нормальный

1: Экономный

— Бит 1 **IREN**: Включение IrDA

Ставится и снимается программно.

0: IrDA выключен

1: IrDA включён

— Бит 0 **EIE**: Разрешение прерывания ошибки

0: Нельзя

1: Можно при DMAR=1 в регистре USART_CR3 или FE=1 или ORE=1 или NE=1 в USART_SR.

30.6.7. Регистр времени защиты и предделителя (USART_GTPR)

Смещение адреса: 0x18

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GT[7:0]								PSC[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:16 Резерв, не трогать.

— Биты 15:8 **GT[7:0]**: Значение защиты в тактах связи режима Smartcard
Флаг TC ставится после отсчёта данного значения. В UART4 и UART5 нету.

— Биты 7:9 PSC[7:0]: Значение предделителя

– **В экономном режиме IrDA**:
PSC[7:0] = Скорость связи IrDA
Делитель входных тактов (8 значащих бит):
00000000: Резерв, не трогать.
00000001: делит на 1
00000010: делит на 2
...

– **В нормальном режиме IrDA:** PSC должен быть равен 00000001.

– **В режиме Smartcard**:
PSC[4:0]: Значение предделителя
Входные такты делятся на удвоенное значение предделителя (5 значащих бит):
00000: Резерв, не трогать.
00001: делит на 2
00010: делит на 4
00011: делит на 6
...

NB: Биты [7:5] в режиме Smartcard не используются.
В UART4 и UART5 нету.

30.6.8.Карта регистров USART

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**USART_SR**	Reserved																						CTS	LBD	TXE	TC	RXNE	IDLE	ORE	NF	FE	PE
	Reset value																							0	0	1	1	0	0	0	0	0	0
0x04	**USART_DR**	Reserved																						DR[8:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x08	**USART_BRR**	Reserved																DIV_Mantissa[15:4]												DIV_Fraction [3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**USART_CR1**	Reserved																OVER8	Reserved	UE	M	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	RWU	SBK
	Reset value																	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**USART_CR2**	Reserved																	LINEN	STOP [1:0]		CLKEN	CPOL	CPHA	LBCL	Reserved	LBDIE	LBDL	Reserved	ADD[3:0]			
	Reset value																		0	0	0	0	0	0	0		0	0		0	0	0	0
0x14	**USART_CR3**	Reserved																				ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x18	**USART_GTPR**	Reserved																GT[7:0]								PSC[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

31. Интерфейс Secure digital (SDIO)

31.1. Основные свойства SDIO

Хост-интерфейс SD/SDIO MMC card (SDIO) соединяет периферийную шину AHB и MultiMediaCards (MMCs), карты памяти SD, карты SDIO и устройства CE-ATA.

Спецификация системы MultiMediaCard от технического комитета MMCA доступна на сайте MultiMediaCard Association www.mmca.org.

Спецификация системы карт памяти SD и карт SD I/O доступна на сайте SD card Association www.sdcard.org.

Спецификация системы CE-ATA доступна на сайте CE-ATA workgroup www.ce-ata.org.

Свойства SDIO:

- Полное соответствие *MultiMediaCard System Specification Version 4.2*. Поддержка трёх режимов шины данных карты: 1-бит (по умолчанию), 4-бит и 8-бит
- Полная совместимость с предыдущими версиями MultiMediaCards
- Полное соответствие *SD Memory Card Specifications Version 2.0*
- Полное соответствие *SD I/O Card Specification Version 2.0*: Поддержка двух режимов шины данных карты: 1-бит (по умолчанию) и 4-бит
- Полная поддержка CE-ATA (полное соответствие *CE-ATA digital protocol Rev 1.1*)
- Передача данных до 48 MHz в 8-бит режиме
- Сигналы разрешения вывода команд и данных для внешних двунаправленных драйверов.

NB: SDIO не имеет SPI-совместимого режима связи.

Протокол карт памяти SD это надстройка протокола MultiMediaCard, определённого в *MultiMediaCard system specification V2.11*. Некоторые команды устройств памяти SD не поддерживаются SD I/O-только картами или блоками I/O комбинированных карт. Некоторые из этих команд не используются в устройствах SD I/O, вроде команд стирания, и соответственно не поддерживаются SDIO. Кроме того, несколько команд карт памяти SD и карт SD I/O различаются и не поддерживаются SDIO. См. *SD I/O card Specification Version 1.0*. CE-ATA поддерживаются электрическим интерфейсом MMC по протоколу, использующему существующие примитивы доступа MMC. Определения электрических сигналов даны в руководствах MMC.

Шина MultiMediaCard/SD подключает карты к контроллеру.

Текущая версия SDIO поддерживает только одну карту SD/SDIO/MMC4.2 одновременно и стек MMC4.1 или ранее.

31.2. Топология шины SDIO

Связь по шине основана на передаче команд и данных в режиме команда/ответ. Информация передаётся внутри структур команды и ответа. Кроме того, некоторые операции имеют токены данных.

Данные карт памяти SD/SDIO передаются блоками. Данные MMC передаются блоками или потоком. Данные устройств CE-ATA передаются блоками.

Рис. 321. Операции "без ответа" и "без данных" SDIO.

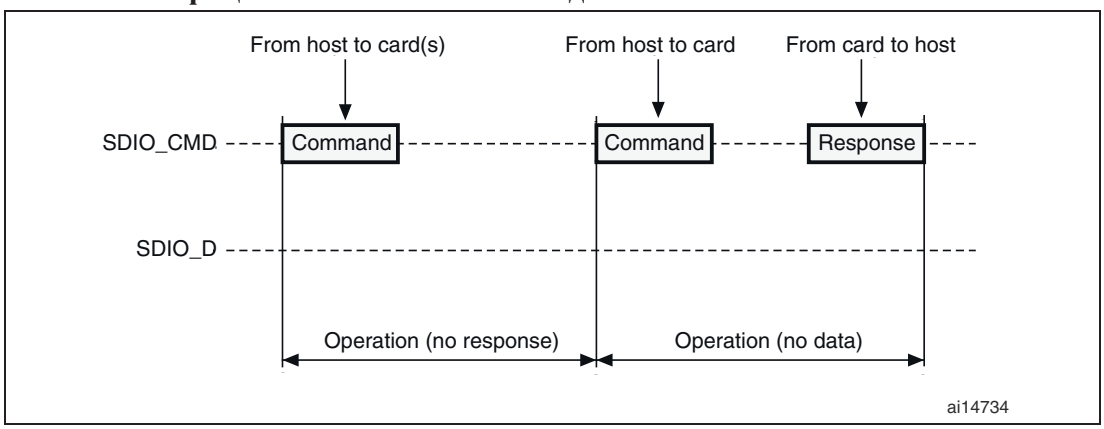


Рис. 322. Блоковое (множественное) чтение SDIO.

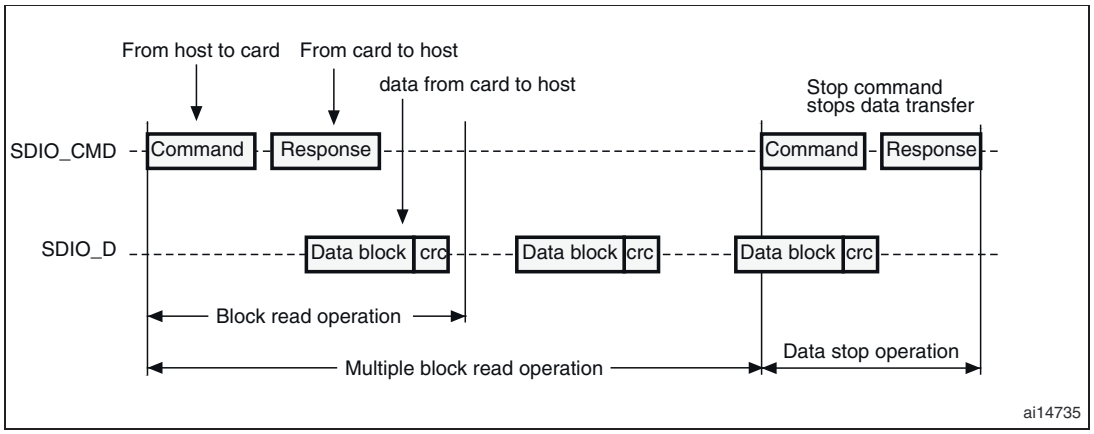
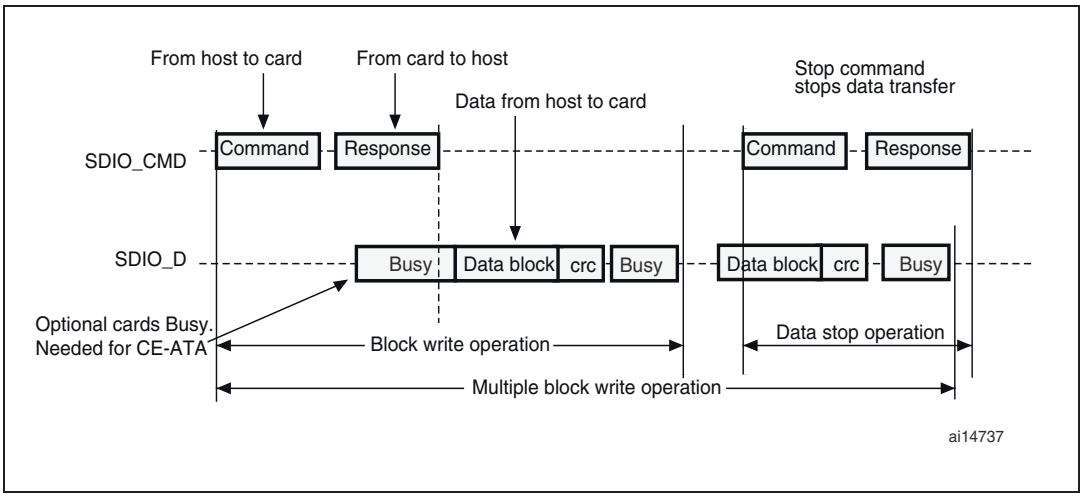


Рис. 323. Блоковая запись SDIO.



NB: При стоящем сигнале Busy (SDIO_D0 низкий) SDIO данные не посылает.

Рис. 324. Последовательное чтение SDIO.

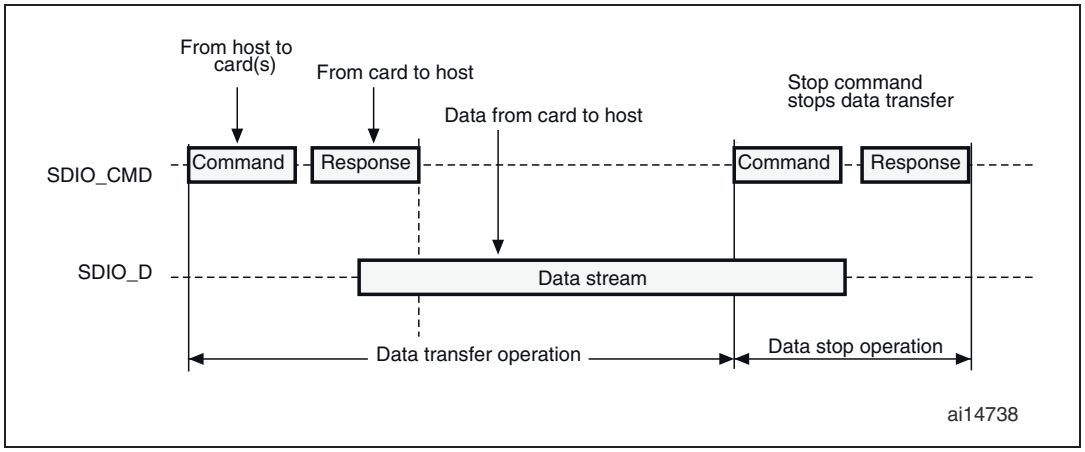
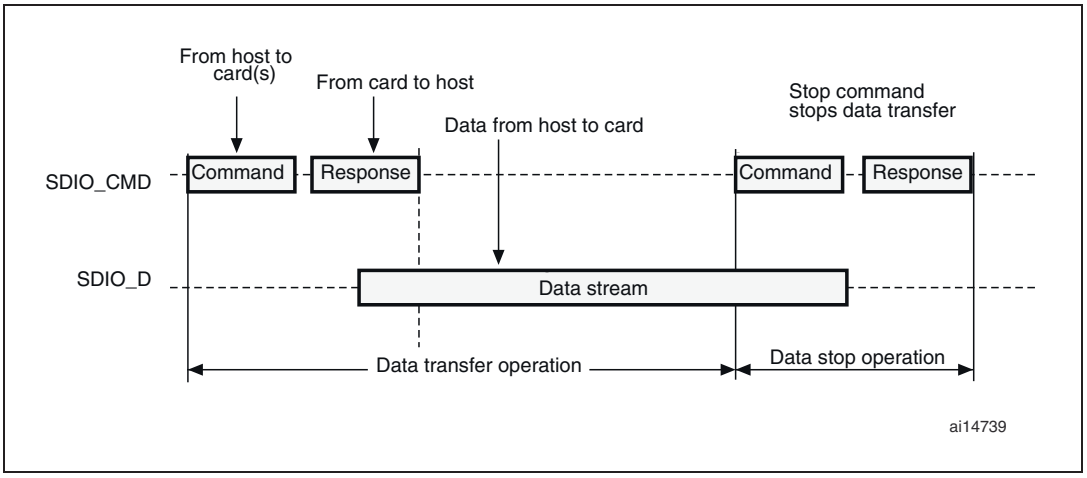


Рис. 325. Последовательная запись SDIO.

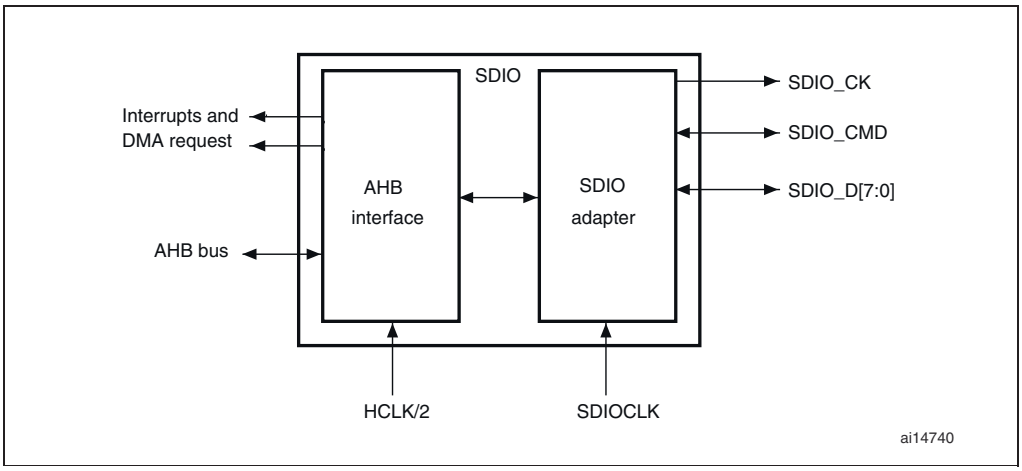


31.3. Функциональное описание SDIO

SDIO состоит из двух частей:

- Блок адаптера SDIO обеспечивает функции карт MMC/SD/SD I/O вроде тактов и передач.
- Интерфейс AHB для доступа к регистрам адаптера SDIO, выдачи прерываний и запросов DMA.

Рис. 326. Блок-схема SDIO.



По умолчанию SDIO_D0 используется для передачи данных. После инициализации хост может менять ширину шины данных.

С картами MultiMediaCard для передачи данных можно использовать SDIO_D0, SDIO_D[3:0] или SDIO_D[7:0]. MMC V3.31 и ранее поддерживают только 1 бит данных на SDIO_D0.

С картами SD SD I/O для передачи данных можно использовать SDIO_D0 или SDIO_D[3:0]. Все линии данных работают в двухтактном режиме.

SDIO_CMD работает в двух режимах:

- Открытый сток при инициализации (только для MMCV3.31 и ранее)
- Двухтактный для передачи команд (SD/SD I/O карты MMC4.2 используют двухтактный режим и при инициализации)

SDIO_CK тактирует карту: по одному такту на один бит на линиях команд и данных. Частота может меняться от 0 MHz до 20 MHz (для MultiMediaCard V3.31), от 0 до 48 MHz для MultiMediaCard V4.0/4.2, и от 0 до 25 MHz (для SD/SD I/O карт).

SDIO использует два тактовых сигнала:

- Такты адаптера SDIO (SDIOCLK = HCLK)
- Такты шины AHB (HCLK/2)

Частоты PCLK2 и SDIO_CK должны быть:

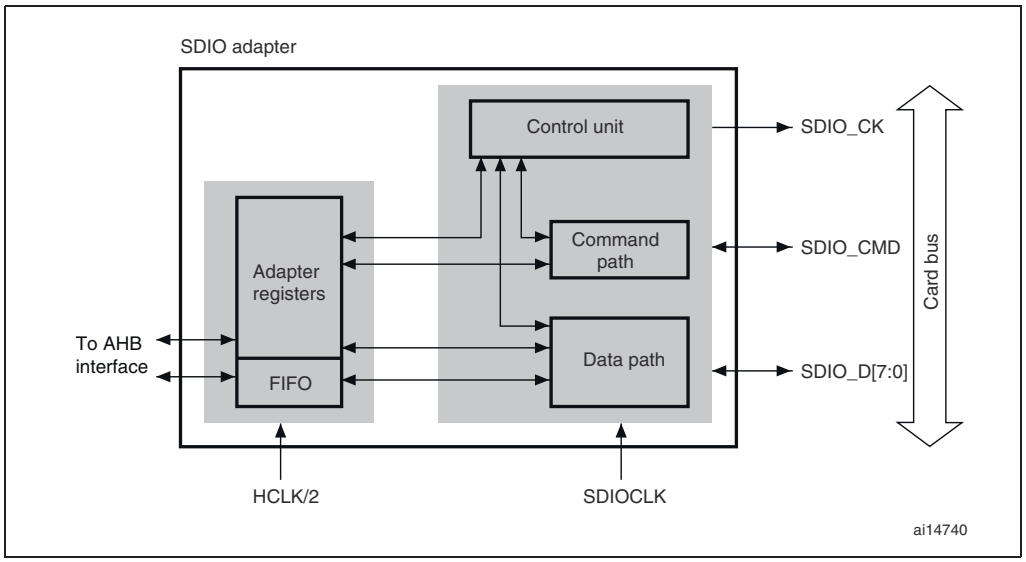
$$\text{Frequency(PCLK2)} \geq \frac{3}{8} \cdot \text{Frequency(SDIO\_CK)}$$

Таблица 150. Определения I/O SDIO.

Ножка	Направление	Описание
SDIO_CK	Вывод	Такты карт MultiMediaCard/SD/SDIO. Из хоста карте.
SDIO_CMD	Двунаправленная	Команда/ответ карты MultiMediaCard/SD/SDIO.
SDIO_D[7:0]	Двунаправленная	Данные от/для карты MultiMediaCard/SD/SDIO.

31.3.1. Адаптер SDIO

Рис. 327. Блок-схема.



Состоит из пяти под-блоков:

- Блок регистров адаптера
- Блок управления
- Путь команд
- Путь данных
- FIFO данных

NB: Регистры адаптера и FIFO используют домен тактов шины AHB (HCLK/2). Блок управления, путь команд и путь данных используют домен тактов адаптера SDIO (SDIOCLK).

Блок регистров адаптера

Содержит все системные регистры и генерирует сигналы очистки статических флагов карт мультимедиа по записи 1 в соответствующий бит регистра очистки SDIO.

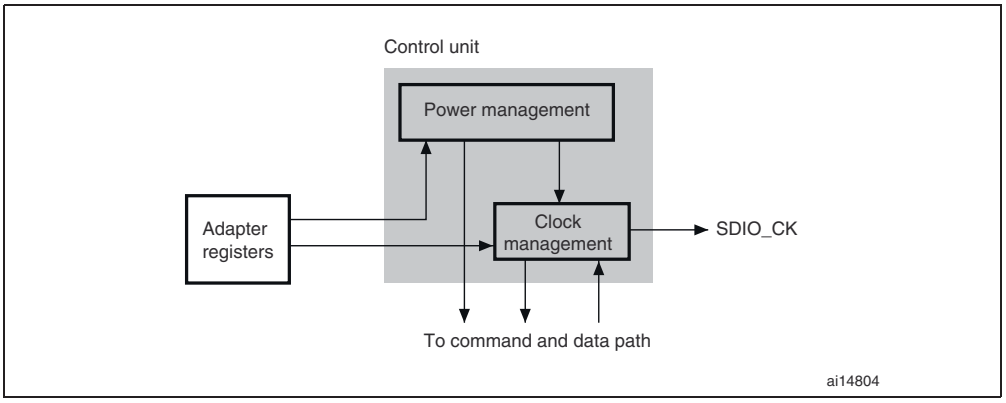
Блок управления

Содержит управление питанием и делитель тактов для карт памяти.

Есть три фазы питания:

- Выключено
- Подъём
- Подача

Рис. 328. Блок управления.



В фазах выключенного и поднимающегося питания блок управления питанием отключает выходные сигналы карты.

Блок тактирования выдаёт сигнал SDIO_CK, который можно получить делением тактов или обходом. Выход тактов неактивен:

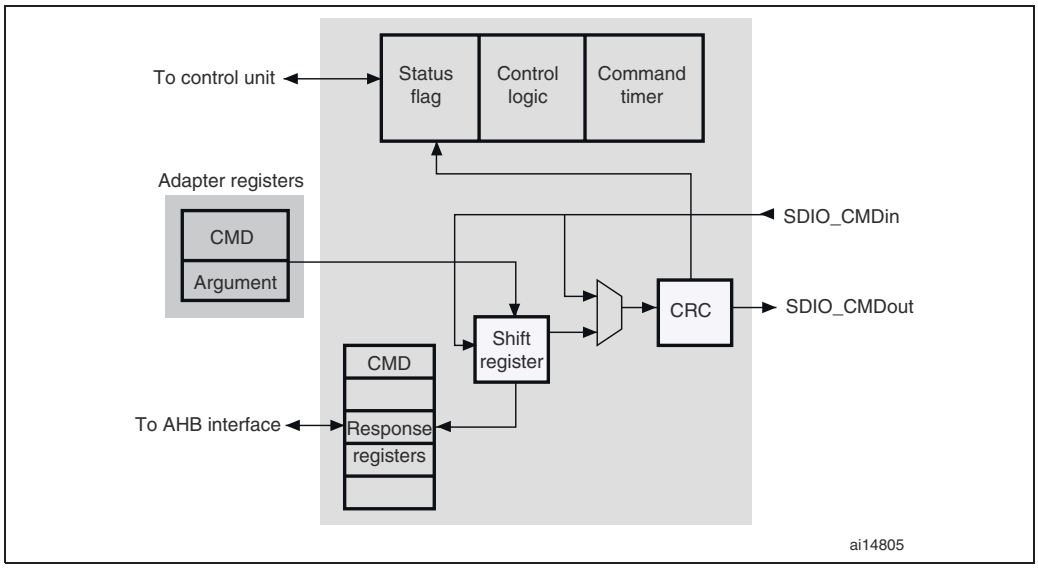
- после сброса
- во время фаз выключенного и поднимающегося питания

• в режиме низкого энергопотребления и Простое шины карты (восемь тактов после перехода путей команд и данных в Простой)

Путь команд

Путь команд посылает картам команды и принимает их ответы.

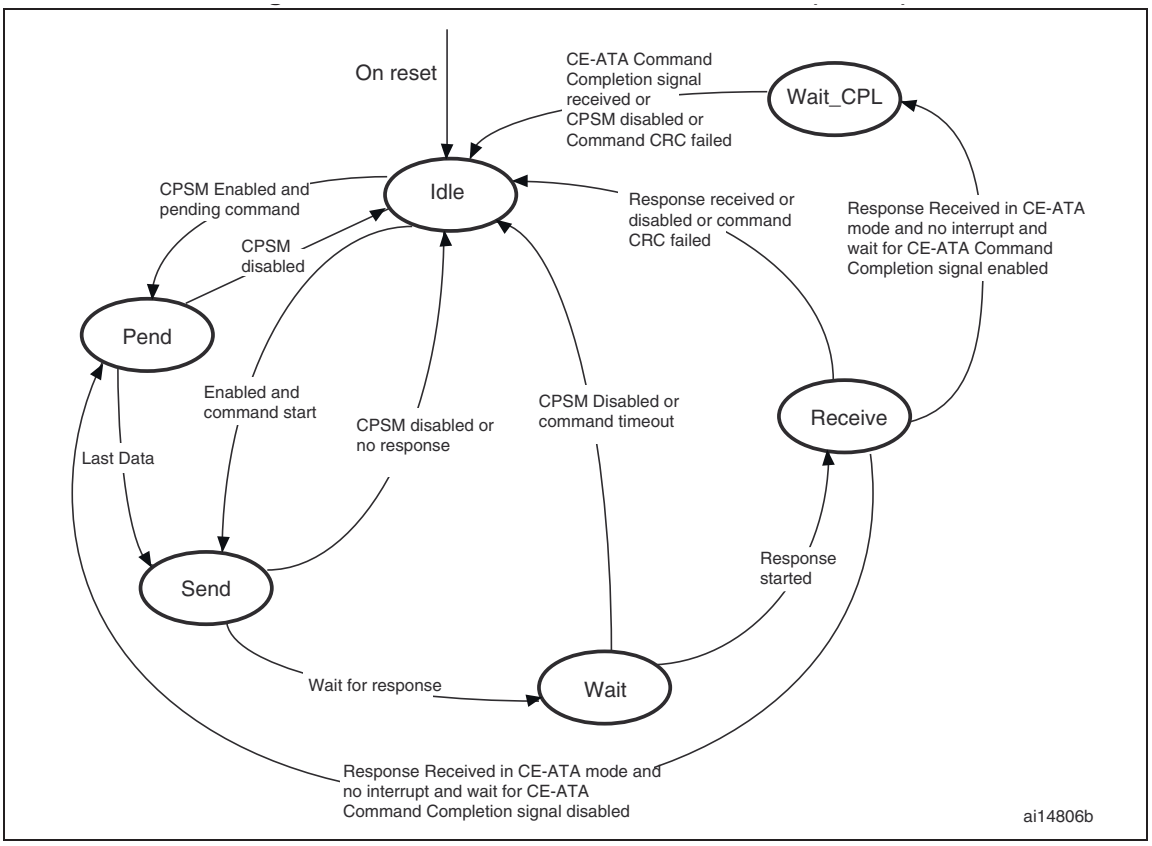
Рис. 329. Путь команд



• Машина состояния пути команд (CPSM)

– Передача команды начинается после записи регистра и стоящем бите разрешения. Если ответ не требуется, то после передачи команды CPSM ставит флаг и переходит в Простой. Иначе она дожидается ответа. По получении ответа полученный CRC сравнивается с полученным внутри и ставится и ставится соответствующий флаг.

Рис. 330. Машина состояния пути команд (CPSM)

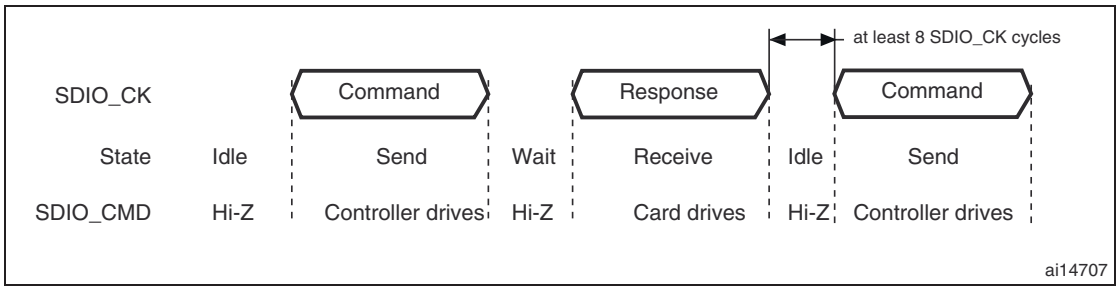


При переходе в Ожидание запускается таймер команды на 64 такта SDIO_CK. Если он истекает до перехода CPSM в Приём, то ставится флаг таймаута и CPSM переходит в Простой.

Если стоит флаг прерывания, то таймер выключается и CPSM ждёт прерывания от одной из карт. Если стоит бит удержания, то CPSM переходит в Удержание и ждёт сигнала CmdPend от пути данных. При обнаружении CmdPend CPSM переходит в состояние Посылки. Этим счётчику данных разрешается остановка передачи команд.

NB: CPSM остаётся в простое не меньше восьми периодов SDIO_CK ради удовлетворения требований NCC и NRC. NCC это минимальная задержка между двумя командами хоста, а NRC это минимальная задержка между командой хоста и ответом карты.

Рис. 331. Передача команды.



• Формат команды

– Команда: это токен пуска операции. Она посылается хостом одной (адресная команда) или всем подключённым картам (широковещательные команды доступны для MMC V3.31 и раньше). Команды передаются последовательно по линии CMD. Они имеют длину 48 бит. Общий формат токена команд для MultiMediaCards, карт памяти SD и карт SDIO приведён в таблице 138. Команды CE-ATA это расширение команд MMC V4.2 с тем же форматом.

Путь команд работает в полудуплексном режиме, так что команды и ответы можно посылать и принимать. При CPSM не в режиме Посылки выход SDIO_CMD уходит в состояние Hi-Z. Данные на SDIO_CMD синхронны с передним фронтом SDIO_CK.

– Ответ: это токен ответа адресованной карты (или синхронно от всех подключённых карт для MMC V3.31 или раньше) на переданную хостом команду. Ответы передаются последовательно по линии CMD.

SDIO поддерживает два типа ответов, оба с контролем CRC:

- короткий 48-бит
- длинный 136-бит

NB: Если ответ не содержит CRC (ответ CMD1), то драйвер должен игнорировать сбой CRC.

Таблица 151. Формат команды.

Позиция бита	Ширина	Значение	Описание
47	1	0	Стартовый бит
46	1	1	Бит передачи
[45:40]	6	-	Индекс команды
[39:8]	32	-	Аргумент
[7:1]	7	-	CRC7
0	1	1	Концевой бит

Таблица 152. Формат короткого ответа.

Позиция бита	Ширина	Значение	Описание
47	1	0	Стартовый бит
46	1	0	Бит передачи
[45:40]	6	-	Индекс команды
[39:8]	32	-	Аргумент
[7:1]	7	-	CRC7(или 1111111)
0	1	1	Концевой бит

Таблица 153. Формат длинного ответа.

Позиция бита	Ширина	Значение	Описание
135	1	0	Стартовый бит
134	1	0	Бит передачи
[133:128]	6	111111	Резерв
[127:1]	127	-	CID или CSD (включая внутренний CRC7)
0	1	1	Концевой бит

Регистр команд содержит индекс команды (посылаемые карте шесть бит) и её тип. Они определяют необходимость ответа и его длину. Путь команд реализует флаги состояния:

Таблица 154. Флаги состояния пути команд.

Флаг	Описание
CMDREND	Нормальный CRC ответа.
CCRCFAIL	Сбойный CRC ответа.
CMDSENT	Команда послана (ответа не надо)
CTIMEOUT	Таймаут ответа.
CMDACT	Команда передаётся.

Генератор CRC вычисляет контрольную сумму всех битов перед кодом CRC. Включаются стартовый бит, бит передачи, индекс команды и её аргумент (или состояние карты). В длинном ответе контрольная сумма вычисляется для первых 120 бит `CID` или `CSD`. Стартовый бит, бит передачи и шесть резервных бит в вычислении CRC не участвуют.

Контрольная сумма CRC это 7-бит значение:

$CRC[6:0] = Остаток\ [(M(x) * x^7) / G(x)]$

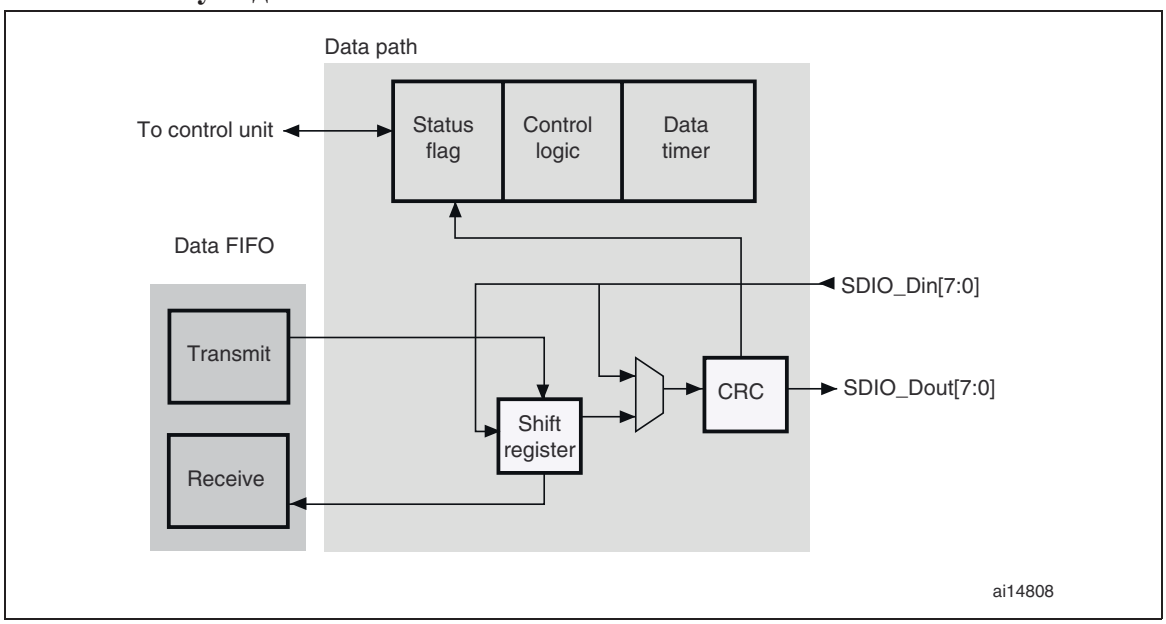
$G(x)=x^7 +x^3 +1$

$M(x) = (начальный\ бит) * x^{39} + ... + (последний\ бит\ перед\ CRC) * x^0$, или

$M(x) = (начальный\ бит) * x^{119} + ... + (последний\ бит\ перед\ CRC) * x^0$

Путь данных.

Рис. 332. Путь данных.



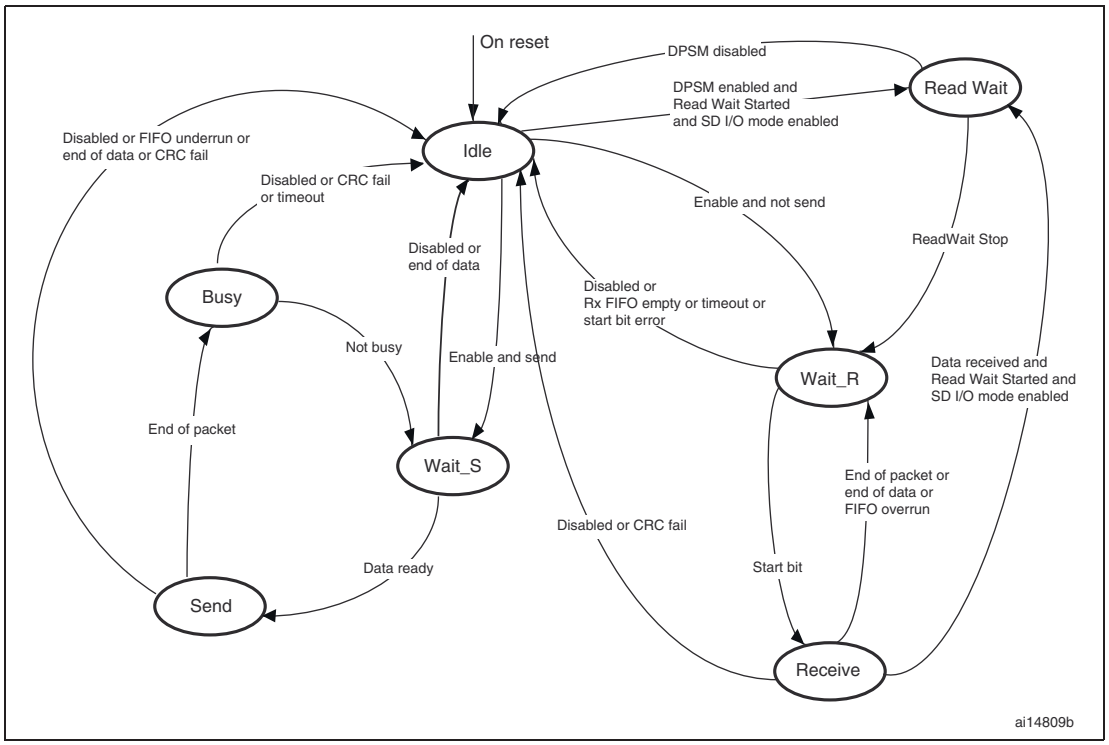
Ширина шины данных карты определяется в регистре управления тактами. При 4-бит шине данные передаются по четыре бита на такт по сигналам `SDIO_D[3:0]`. При 8-бит шине данные передаются по восемь битов на такт по сигналам `SDIO_D[7:0]`. При отключённой ширине шины данные передаются по одному биту на такт по `SDIO_D0`.

В зависимости от направления передачи машина состояния пути данных (DPSM) переходит в состояние Wait_S или Wait_R, если можно:

- Передача: DPSM идёт в Wait_S. Если в FIFO есть данные, то DPSM переходит в состояние Посылки и начинает посылать карте данные.
- Приём: DPSM идёт в Wait_R и ждёт стартового бита. При его появлении DPSM переходит в состояние Приёма и начинает принимать данные от карты.

DPSM работает с частотой `SDIO_CK`. Сигналы данных на шине карты синхронны с передним фронтом `SDIO_CK`. DPSM имеет шесть состояний.

Рис. 333. Машина состояния пути данных (DPSM)



- **Простой**: путь данных неактивен, выходы `SDIO_D[7:0]` в Hi-Z. После записи регистра управления данных и установки бита разрешения DPSM пишет новое число в счётчик данных и в зависимости от направления передачи переходит в Wait_S или Wait_R.
- **Wait_R**: если счётчик данных нулевой, то при пустом FIFO DPSM идёт в Простой. Если счётчик данных не нулевой, то DPSM ждёт стартового бита на `SDIO_D`. При получении его до таймаута DPSM уходит в Приём и заполняет счётчик данных блока. При достижении таймаута до стартового бита или его ошибке он уходит в Простой и ставит бит таймаута.
- **Приём**: принятые последовательные данные пишутся в FIFO данных. В зависимости от бита в регистре управления режим передачи может быть блоками или потоком:
  - В режиме блоков, при исчерпании счётчика данных блока DPSM дожидается приёма CRC. При его совпадении с вычисленным DPSM идёт в Wait_R. Иначе ставится флаг сбоя CRC с уходом в Простой.
  - В режиме потока DPSM принимает данные до исчерпания счётчика. Тогда остаток данных в регистре сдвига пишется в FIFO и DPSM идёт в Wait_R.
  - При переполнении FIFO DPSM ставит флаг ошибки FIFO и идёт в Простой.
- **Wait_S**: При нулевом счётчике данных DPSM уходит в Простой. Иначе он ждёт снятия флага пустого FIFO данных и идёт в Посылку.

  NB: DPSM остаётся в Wait_S не менее двух тактов для соблюдения N_{WR}, где N_{WR} это число тактов от приёма ответа карты и стартом передачи данных от хоста.
- **Посылка**: DPSM начинает передачу данных карте. В зависимости от бита в регистре управления режим передачи может быть блоками или потоком:
  - В режиме блоков, при исчерпании счётчика данных блока DPSM посылает вычисленный CRC с битом конца и уходит в Занят.
  - В режиме потока DPSM посылает данные карте пока стоит бит разрешения и счётчик не нулевой. Затем спокойно уходит в Простой.

  При исчерпании FIFO DPSM ставит флаг ошибки FIFO и идёт в Простой.

- **Занят**: DPSM ждёт флага состояния CRC:
  - Если он появляется плохим, то уходит в Простой и ставит флаг сбоя CRC.
  - При хорошем CRC и высоком `SDIO_D0` (карта не занята) идёт в Wait_S.

  При появлении таймаута в этом состоянии DPSM ставит флаг таймаута данных и уходит в Простой.

В режимах DPSM Wait_R и Занят таймер данных включён и выдаёт ошибку таймаута данных когда:

- При передаче данных DPSM остаётся Занятым дольше периода таймаута
- При приёме данных не достигнут конец данных и DPSM стоит в Wait_R дольше периода таймаута.

- **Данные:** передача по линиям данных работает. Они хранятся в FIFO из 32 слов по 32 бита.

Таблица 155. Формат токена данных.

Описание	Стартовый бит	Данные	CRC16	Бит конца
Блок данных	0	-	Да	1
Поток	0	-	Нет	1

FIFO данных

Это буфер данных с блоками приёма и передачи из 32-х 32-бит слов. Тактируется от домена AHB (HCLK/2), так что сигналы в домен SDIO (SDIOCLK) ресинхронизируются.

Битами `TXACT` и `RXACT` FIFO может отключаться, включаться приём или передача.Взаимно исключающие `TXACT` и `RXACT` управляются путём данных:

• FIFO передачи:

- При стоящем `TXACT` FIFO является логикой и буфером передачи
- При стоящем `RXACT` FIFO является логикой и буфером приёма

В передающий FIFO данные пишутся по интерфейсу AHB при разрешении передач SDIO.

Передающий FIFO доступен по 32 последовательным адресам с указателями чтения и записи. Указатель чтения инкрементируется при загрузке путём данных своего регистра сдвига.

При выключенном FIFO все флаги состояния снимаются. Путь данных ставит `TXACT` во время передачи.

Таблица 156. Флаги состояния FIFO передачи.

Флаг	Описание
TXFIFOF	Все 32 слова FIFO имеют данные.
TXFIFOE	Данных в FIFO нет.
TXFIFOHE	Половина или больше слов FIFO пусты. Может стать запросом DMA.
TXDAVL	Данные в FIFO есть. Инверсия флага TXFIFOE.
TXUNDERR	Ошибка исчерпания. Снимается записью в регистр очистки SDIO.

• FIFO приёма

После получения слова данных путь данных выставляет его на на шину записи. Указатель записи инкрементируется после завершения операции. На стороне чтения слово FIFO по текущему указателю чтения выставляется на шину чтения. При выключенном FIFO чтения все флаги снимаются и указатели чтения и записи сбрасываются. Путь данных выставляет `RXACT` во время приёма. Доступен приёмный FIFO по 32 последовательным адресам.

Таблица 157. Флаги состояния FIFO приёма.

Флаг	Описание
RXFIFOF	Все 32 слова FIFO имеют данные.
RXFIFOE	Данных в FIFO нет.
RXFIFOHF	Половина или больше слов FIFO полны. Может стать запросом DMA.
RXDAVL	Данные в FIFO есть. Инверсия флага RXFIFOE.
RXOVERR	Ошибка переполнения. Снимается записью в регистр очистки SDIO.

31.3.2. AHB интерфейс SDIO

Интерфейс AHB выдаёт прерывания и запросы DMA, обращается к регистрам адаптера SDIO и FIFO. Состоит из пути данных, декодера регистров и логики прерываний/DMA.

Прерывания SDIO

Прерывания вызываются стоящим флагом состояния, при стоящем соответствующем бите в регистре маски.

Интерфейс SDIO/DMA: процедура передачи данных

Ниже приведена передача из хост контроллера SDIO в MMC (512 байт с использованием CMD24 (WRITE_BLOCK)). FIFO SDIO заполняется данными из памяти контроллером DMA.

1. Идентифицировать карту
2. Увеличить частоту SDIO_CK
3. Выбрать карту посылкой CMD7
4. Сконфигурировать DMA2:
   a) Включить контроллер DMA2 и очистить удерживаемые прерывания
   b) Поставить адрес источника DMA2_Channel4 на память и адрес получателя на адрес регистра SDIO_FIFO
   c) Установить регистр управления DMA2_Channel4 (инкремент памяти, периферия без инкремента, ширина источника и получателя - слово)
   d) Включить DMA2_Channel4
5. Послать CMD24 (WRITE_BLOCK):
   a) Записать регистр длины данных SDIO (Регистр таймера данных SDIO надо писать перед идентификацией карты)
   b) Записать в регистр аргумента SDIO адрес в памяти карты (куда писать)
   c) Установить регистр команд SDIO: "24" в CmdIndex (WRITE_BLOCK); "1" в WaitResp (хост SDIO карты ждёт ответа); "1" в CPSMEN (хост SDIO может посылать команду). Другие поля остаются в состоянии сброса.
   d) Дождаться прерывания SDIO_STA[6] = CMDREND, записать регистр управления данными SDIO: '1' в DTEN (хост SDIO может посылать данные); '0' в DTDIR (из контроллера в карту); '0' в DTMODE (передача блока); '1' в DMAEN (DMA включён); 0x9 в DBLOCKSIZE (512 байт). Иные поле не волнуют.
   e) Дождаться SDIO_STA[10] = DBCKEND
6. Опросить регистр состояния каналов DMA на предмет ещё включённых каналов.

31.4. Функциональное описание карты

31.4.1. Режим идентификации карт

Тут хост сбрасывает все карты на шине, определяет диапазон рабочего напряжения, идентифицирует карты и устанавливает из относительные адреса (RCA) на шине. Все данные передаются только по линии команд (CMD).

31.4.2. Сброс карт

Команда сброса GO_IDLE_STATE (CMD0) переводит MultiMediaCard и карты памяти SD в Простой. Команда IO_RW_DIRECT (CMD52) сбрасывает карты SD I/O. После подачи питания или CMD0 все драйверы шины переводятся в высокоимпедансное состояние и карты инициализируются с относительным адресом (RCA=0x0001) и установками каскадов драйверов (наименьшая скорость, наибольший ток) по умолчанию.

31.4.3. Определение рабочего напряжения

Все карты беседуют с хостом SDIO на рабочем уровне напряжения V_{DD}, определённом в регистре условий работы (OCR) карты.

Карты, хранящие идентификационный номер (CID) и специфичные данные карты (CSD) в памяти данных, могут обмениваться этой информацией только в условиях V_{DD}. При различных уровнях V_{DD} хоста SDIO и карты она не может закончить цикл идентификации и послать данные CSD. Для целей идентификации и извлечения карт, не попадающих в требуемое окно V_{DD}, придуманы специальные команды: SEND_OP_COND (CMD1), SD_APP_OP_COND (ACMD41 для памяти SD) и IO_SEND_OP_COND (CMD5 для SD I/O). Хост SDIO посылает требуемое окно V_{DD} в операндах этих

команд. Карты, которые не могут этого выполнить, отключаются от шины и переходят в неактивное состояние.

Этими командами без диапазона напряжений в параметрах хост SDIO может опросить все карты для определения общего диапазона и перевода неподходящих карт в неактивное состояние. Кроме того, этим можно по требованию пользователя определить неиспользуемые карты.

31.4.4. Процесс идентификации карт

Он различен для MultiMediaCard и SD карт.

Для MultiMediaCard процесс стартует на частоте F_{od}. Выходной каскад линии SDIO_CMD стоит в режиме открытого стока, разрешая параллельную работу карт. Регистрация идёт так:

1. Активируется шина.
2. Хост SDIO широковещает SEND_OP_COND (CMD1) для приёма условий работы.
3. Ответ это проводное AND регистров условий работы.
4. Несовместимые карты уходят в неактивное состояние.
5. Хост SDIO широковещает ALL_SEND_CID (CMD2) всем активным картам.
6. Активные карты одновременно шлют свои CID номера по последовательной линии. Карты с отдельными битами CID, не совпадающими с битами на командной линии, останавливают передачу и должны ждать следующего цикла идентификации. Одна карта, успешно передавшая полный CID, идёт в состояние Идентификации.
7. Хост SDIO посылает ей SET_RELATIVE_ADDR (CMD3) с адресом. Это относительный адрес карты (RCA); он короче CID. Обозначенная карта уходит в Дежурство (Standby) и не реагирует на следующие циклы идентификации. Выходной каскад переключается из открытого стока в двухтактный режим.
8. Хост SDIO повторяет шаги 5 — 7 до получения таймаута.

У SD карт процесс стартует на частоте F_{od} и выходной каскад линии SDIO_CMD стоит в двухтактном режиме вместо режиме открытого стока. Регистрация идёт так:

1. Активируется шина.
2. Хост SDIO широковещает SD_APP_OP_COND (ACMD41).
3. Карты отвечают содержимым их регистров условий работы.
4. Несовместимые карты уходят в неактивное состояние.
5. Хост SDIO широковещает ALL_SEND_CID (CMD2) всем активным картам.
6. Карты шлют свои CID и уходят в Идентификацию.
7. Хост SDIO посылает активной карте SET_RELATIVE_ADDR (CMD3) с адресом. Это относительный адрес карты (RCA); он короче CID. Обозначенная карта уходит в Дежурство (Standby). Этой командой хост SDIO может изменять RCA карт.
8. Хост SDIO повторяет шаги 5 — 7 со всеми активными картами.

Для карт SD I/O регистрация идёт так:

1. Активируется шина.
2. Хост SDIO посылает IO_SEND_OP_COND (CMD5).
3. Карты отвечают содержимым их регистров условий работы.
4. Несовместимые карты уходят в неактивное состояние.
5. Хост SDIO посылает активной карте SET_RELATIVE_ADDR (CMD3) с адресом. Это относительный адрес карты (RCA); он короче CID. Обозначенная карта уходит в Дежурство (Standby). Этой командой хост SDIO может изменять RCA карт.

31.4.5. Запись блоками

Во время блоковой записи (CMD24 – 27) хост посылает карте, поддерживающей такой режим, один или несколько блоков данных с добавленным в конце CRC блока. Длина блока определена в WRITE_BL_LEN. При сбое CRC карта показывает его на линии SDIO_D, отбрасывает передаваемые данные и игнорирует остальные передаваемые блоки многоблоковой передачи.

Если хост использует неполные блоки с общей длиной не кратной длиной блока и невыравненные блоки не разрешены (параметр CSD WRITE_BLK_MISALIGN не стоит), то карта обнаружит ошибку

перед началом первого невыравненного блока (бит ADDRESS_ERROR в регистре состояния. Также операция записи будет прервана при попытки записи в защищённую область. При этом ставится бит WP_VIOLATION.

Запись регистров CID и CSD не требует предварительной установки длины блока. Передаваемые данные также защищены CRC. Если часть регистра CSD или CID хранится в ROM, то эта часть сравнивается с соответствующими данными в приёмном буфере. При несовпадении карта сообщает от ошибке и не меняет остальную часть регистров. Некоторые карты требуют долгого и непредсказуемого времени для записи блока данных. После приёма блока и проверки CRC карта начинает запись и держит линию SDIO_D низкой вплоть до готовности приёма новой команды WRITE_BLOCK. Хост может в любое время опросить состояние карт командой SEND_STATUS (CMD13). Бит READY_FOR_DATA показывает готовность к приёму данных. Задумчивую карту можно перевести в режим Отключки (Disconnect) командой CMD7, не прерывая её записи. При этом линии SDIO_D освобождаются для других карт. При повторном выборе карты она показывает свою занятость состоянием SDIO_D (низкий при продолжающейся записи и недоступном буфере).

31.4.6. Чтение блоками

Максимальный размер блока передачи определён в CSD (READ_BL_LEN). При стоящем бите READ_BL_PARTIAL можно передавать блоки меньшего размера, чьи адреса начала и конца лежат внутри физического блока. В конце каждого блока добавляется CRC. Команда CMD17 (READ_SINGLE_BLOCK) инициирует чтение одного блока с возвращением состояние Передачи.

Команда CMD18 (READ_MULTIPLE_BLOCK) запускает передачу нескольких последовательных блоков. Командой остановки передачи хост может прервать такое чтение в любое время, независимо от типа.

Если во время множественной передачи (обоих типов) возникает ошибка (выход за границы, невыравненный адрес или внутренняя ошибка), то карта прекращает передачу и возвращается в состояние Данных. Хост должен послать команду остановки передачи. Ошибка чтения возвращается в ответ на эту команду.

После передачи предопределённого числа блоков карта выходит из состояния Данных и присылаемая команда остановки передачи является незаконной, о чём и сообщается хосту. При неразрешённой передаче невыравненных блоков ошибка выявляется в начале первого невыравненного блока (бит ошибки ADDRESS_ERROR в регистре состояния).

31.4.7. Потоковые чтение и запись (только MultiMediaCard)

В потоковом режиме данные передаются байтами без добавления CRC в конце каждого блока.

Потоковая запись (только MultiMediaCard)

WRITE_DAT_UNTIL_STOP (CMD20) пускает передачу данных от хоста SDIO карте, начиная с заданного адреса, вплоть до команды останова от хоста. Если разрешены неполные блоки (параметр CSD WRITE_BLK_MISALIGN стоит), то поток данных может начинаться и заканчиваться на любом адресе внутри адресного пространства карты, иначе только на границе блока. Поскольку количество данных заранее определить невозможно, то CRC использовать невозможно. Если во время передачи достигнут конец диапазона памяти, то все поступающие данные выбрасываются вплоть до получения команды останова.

Максимальная частота тактов потока использует регистры карты и вычисляется по формуле:

$$\text{Maximumspeed} = \text{MIN}(\text{TRANSPEED}, \frac{(8 \times 2^{\text{writebllen}})(-\text{NSAC})}{\text{TAAC} \times \text{R2WFACTOR}})$$

- Maximumspeed = максимальная частота записи
- TRANSPEED = максимальная частота передачи данных
- writebllen = максимальный размер блока записи
- NSAC = время 2 чтения в тактах CLK
- TAAC = время 1 чтения
- R2WFACTOR = множитель скорости записи

Попытка хоста использовать частоту, большую, чем доступная карте, то она останавливает запись, ставит в регистре состояния ошибку OVERRUN и плюёт на все поступающие данные (в состоянии

приёма данных) вплоть до получения команды останова. Операция записи прерывается также при записи в защищённую область. В этом случае ставится бит WP_VIOLATION.

Потоковое чтение (только MultiMediaCard)

READ_DAT_UNTIL_STOP (CMD11) запускает чтение данных карты начиная с указанного адреса вплоть до посылки хостом команды STOP_TRANSMISSION (CMD12). Команда остановки передаётся последовательно и передача данных останавливается после получения конечного бита команды. Если во время передачи достигнут конец диапазона памяти, то вплоть до получения команды остановки посылается всякая ерунда.

Максимальная частота тактов потока использует регистры карты и вычисляется по формуле:

$$\text{Maximumspeed} = \text{MIN}(\text{TRANSPEED}, \frac{(8 \times 2^{\text{readbllen}})(-\text{NSAC})}{\text{TAAC} \times \text{R2WFACTOR}})$$

- Maximumspeed = максимальная частота чтения
- TRANSPEED = максимальная частота передачи данных
- readbllen = максимальный размер блока чтения
- NSAC = время 2 чтения в тактах CLK
- TAAC = время 1 чтения
- R2WFACTOR = множитель скорости записи

Попытка хоста использовать частоту, большую, чем доступная карте, то она останавливает передачу, ставит в регистре состояния ошибку UNDERRUN и в состоянии Данных ждёт получения команды останова.

31.4.8. Стирание: групповое и сектора

Группа Стирания MultiMediaCard измеряется в блоках записи, основной единицы записи карты. Размер группы определён в CSD. Хост может стирать непрерывный диапазон групп стирания. Процесс запускается последовательностью из трёх шагов.

Сначала хост командой ERASE_GROUP_START (CMD35) указывает адрес начала диапазона, затем командой ERASE_GROUP_END (CMD36) задаёт последний адрес диапазона и наконец запускает стирание командой ERASE (CMD38). Поле адреса в команде стирания содержит адрес группы стирания в байтах. Карта выравнивает этот адрес на границу группы, игнорируя его младшие биты.

При подаче команды стирания вне последовательности в регистре состояния ставится бит ERASE_SEQ_ERROR и сбрасывается вся последовательность.

Если внутри последовательности появляется неположенная команда (кроме SEND_STATUS), то карта ставит в регистре состояния бит ERASE_RESET, сбрасывает последовательность и выполняет последнюю команду.

Попадающие в диапазон стирания защищённые блоки остаются нетронутыми и ставится бит WP_ERASE_SKIP.

Во время стирания линия SDIO_D удерживается низкой. Хост может освободить её, отменив выбор карты командой CMD7.

31.4.9. Широкая шина

Режим широкой шины (4 бита) включается и выключается командой SET_BUS_WIDTH (ACMD6). По включению питания и команде GO_IDLE_STATE (CMD0) используется 1 бит. SET_BUS_WIDTH (ACMD6) допустима только в состоянии передачи, то есть после её выбора командой SELECT/DESELECT_CARD (CMD7).

31.4.10.Управление защитой записи

Хост SDIO поддерживает три метода защиты записи:

1. Внутренняя защита (ответственность карты)
2. Механический переключатель (ответственность хоста SDIO)
3. Защита карты паролём

Внутренняя защита

Биты в CSD включают постоянную или временную защиту записи всей карты. Некоторые карты битом WP_GRP_ENABLE в CSD позволяют включать защиту групп секторов памяти и изменять её

программно. Объём защищённой памяти измеряется в единицах WP_GRP_SIZE секторов из CSD. Команды SET_WRITE_PROT и CLR_WRITE_PROT управляют защитой адресованных групп. Команда SEND_WRITE_PROT подобна чтению одного блока. Карта посылает блок данных с 32 битами защиты записи, представляющими 32 защищённые группы, начиная с указанного адреса, сопроводив это 16 битами CRC. Поле адреса в команде защиты это байтовый адрес группы.

Карта игнорирует биты, младше размера группы.

Механический переключатель

Он расположен сбоку карты и доступен только хосту. Так что вся ответственность за запись возлагается на хост.

Защита паролём

Пароль, замыкающий доступ к карте, хранится в 128-бит регистре PWD, а его длина в 8-бит регистре PWD_LEN. Они энергонезависимые. Замкнутая карта отвечает и выполняет сброс инициализацию, выбор и выдаёт состояние, но без доступа к данным карты. При установленном пароле (ненулевая PWD_LEN) после подачи питания карта замыкается автоматически. Как и команды записи регистров CSD и CID, команды замыкания/размыкания доступны только в состоянии Передачи. Здесь они у них нет аргумента адреса и выбрать карту нужно заранее. Они имеют структуру и тип передачи по шине регулярной команды записи блока. Вся требуемая информация (режим установки пароля, сам пароль и замыкание/размыкание) передаётся в блоке данных. Размер блока данных команды определяется хостом до посылки команды. См. *Таблицу 158*.

Установки битов:

- ERASE: стирание. остальные биты должны быть нулевыми, посылается только бит команды
- LOCK_UNLOCK: замок карты. LOCK_UNLOCK может стоять одновременно с SET_PWD, но не с CLR_PWD
- CLR_PWD: очистка пароля
- SET_PWD: пишет пароль в память
- PWD_LEN: длина пароля в байтах
- PWD: пароль (новый или старый)

Установка пароля

1. Выбрать карту (SELECT/DESELECT_CARD, CMD7), если ещё нет.
2. Задать длину блока посылки (SET_BLOCKLEN, CMD16). Включает 8-бит режима замыкания, 8-бит PWD_LEN, число байтов нового пароля и, при замене пароля, длину старого.
3. Послать LOCK/UNLOCK (CMD42) с блоком данных и 16-бит CRC. Блок данных задаёт режим (SET_PWD = 1), длину (PWD_LEN) и сам пароль (PWD). При замене пароля PWD_LEN задаёт длину обоих паролей, а поле PWD старый и затем новый.
4. При совпадении паролей, новый пароль и его длина пишутся в поля PWD и PWD_LEN. Иначе просто ставится бит ошибки LOCK_UNLOCK_FAILED в регистре состояния.

Замкнуть карту можно сразу при записи пароля, установив бит LOCK_UNLOCK или послав команду замыкания.

Сброс пароля

1. Выбрать карту (SELECT/DESELECT_CARD, CMD7), если ещё нет.
2. Задать длину блока посылки (SET_BLOCKLEN, CMD16). Включает 8-бит режима замыкания и 8-бит PWD_LEN.
3. Послать LOCK/UNLOCK (CMD42) с блоком данных и 16-бит CRC. Блок данных задаёт режим (CLR_PWD = 1), длину (PWD_LEN) и сам пароль (PWD). Бит LOCK_UNLOCK игнорируется.
4. При совпадении паролей, поля PWD и PWD_LEN обнуляются. Иначе просто ставится бит ошибки LOCK_UNLOCK_FAILED в регистре состояния.

Блокировка карты

1. Выбрать карту (SELECT/DESELECT_CARD, CMD7), если ещё нет.
2. Задать длину блока посылки (SET_BLOCKLEN, CMD16). Включает 8-бит режима замыкания (байт 0 в *Таблице 158*.), 8-бит PWD_LEN и число байтов текущего пароля.
3. Послать LOCK/UNLOCK (CMD42) с блоком данных и 16-бит CRC. Блок данных задаёт режим (LOCK_UNLOCK = 1), длину (PWD_LEN) и сам пароль (PWD).

4. При совпадении паролей, карта замыкается и ставится бит CARD_IS_LOCKED в регистре состояния. Иначе просто ставится бит ошибки LOCK_UNLOCK_FAILED в регистре состояния.

Замкнуть карту можно сразу при записи нового пароля, дополнительно установив бит LOCK_UNLOCK режима.

При стоящем пароле (PWD_LEN не 0), карта автоматически замыкается по сбросу питания. Попытка замкнуть уже замкнутую карту или карту без пароля ставит бит ошибки LOCK_UNLOCK_FAILED в регистре состояния.

Разблокировка карты

1. Выбрать карту (SELECT/DESELECT_CARD, CMD7), если ещё нет.
2. Задать длину блока посылки (SET_BLOCKLEN, CMD16). Включает 8-бит режима замыкания (байт 0 в *Таблице 158*.), 8-бит PWD_LEN и число байтов текущего пароля.
3. Послать LOCK/UNLOCK (CMD42) с блоком данных и 16-бит CRC. Блок данных задаёт режим (LOCK_UNLOCK = 1), длину (PWD_LEN) и сам пароль (PWD).
4. При совпадении паролей, карта размыкается и снимется бит CARD_IS_LOCKED в регистре состояния. Иначе просто ставится бит ошибки LOCK_UNLOCK_FAILED в регистре состояния.

Разблокировка действует только в текущем включении питания. При стоящем пароле (PWD_LEN не 0), карта автоматически замыкается по сбросу питания. Попытка разомкнуть уже открытую карту ставит бит ошибки LOCK_UNLOCK_FAILED в регистре состояния.

Насильственное стирание

Если пароль забыт напрочь, то использовать карту можно только после полного стирания её содержимого. Для этого надо:

1. Выбрать карту (SELECT/DESELECT_CARD, CMD7), если ещё нет.
2. Задать длину блока посылки (SET_BLOCKLEN, CMD16) 1 байт для команды замыкания 8-бит карт (байт 0 в *Таблице 158*.).
3. Послать LOCK/UNLOCK (CMD42) с байтом данных режима (ERASE = 1) и 16-бит CRC. Остальные биты байта равны нулю.
4. При единственном стоящем бите ERASE в поле данных стирается вся информация карты, включая регистры PWD и PWD_LEN, и она разблокируется. Если стоят и другие биты, то просто ставится бит ошибки LOCK_UNLOCK_FAILED в регистре состояния, карта остаётся нетронутой и блокированной.

Попытка стирания открытой карты ставит бит ошибки LOCK_UNLOCK_FAILED в регистре состояния.

31.4.11.Регистр состояния карты

32-бит поле состояния карты формата R1 относится к предыдущей команде (если не сказано иное).

В *Таблице 145*. используются следующие аббревиатуры:

Тип:

- E: бит ошибки
- S: бит состояния
- R: обнаружено и установлено для этого ответа
- X: обнаружено и установлено при исполнении команды. Для чтения этих битов хост SDIO должен опрашивать карту командой.

Чистка:

- A: в соответствии с текущим состоянием карты
- B: всегда относится к предыдущей команде. Приём правильной команды чистит их (с задержкой в одну команду)
- C: чистка чтением

Таблица 158. Состояние карты

Биты	Имя	Тип	Значение	Описание	Чистка
31	ADDRESS_ OUT_OF_RANGE	ERX	0= чисто 1= ошибка	Аргумент адреса в команде не годится для этой карты. Очередное чтение/запись блока или потока выходит за допустимые пределы для карты.	C
30	ADDRESS_MISALIGN	-	0= чисто 1= ошибка	Адрес первого блока не выравнен на границу физического блока карты. Очередное чтение/запись блока не выравнено на границу физического блока карты.	C
29	BLOCK_LEN_ERROR	-	0= чисто 1= ошибка	Один из аргументов команды SET_BLOCKLEN превышает допустимое значение для карты или ранее заданная длина блока недопустима для команды (например, при запрещённой записи частями, длина блока меньше длины блока карты)	C
28	ERASE_SEQ_ERROR	-	0= чисто 1= ошибка	Ошибка последовательности команд стирания.	C
27	ERASE_PARAM	EX	0= чисто 1= ошибка	Недопустимый выбор групп стирания.	C
26	WP_VIOLATION	EX	0= чисто 1= ошибка	Попытка записи в защищённый блок.	C
25	CARD_IS_LOCKED	SR	0 =открыто 1 =закрыто	Карта замкнута хостом.	A
24	LOCK_UNLOCK_ FAILED	EX	0= чисто 1= ошибка	Ошибка последовательности или пароля команд блокировки/разблокировки карты.	C
23	COM_CRC_ERROR	ER	0= чисто 1= ошибка	Сбой CRC предыдущей команды.	B
22	ILLEGAL_COMMAND	ER	0= чисто 1= ошибка	Команда недопустима с таким состоянием карты.	B
21	CARD_ECC_FAILED	EX	0= успех 1= сбой	Неисправимая ошибка ECC данных карты.	C
20	CC_ERROR	ER	0= чисто 1= ошибка	Ошибка карты, не относящаяся к командам (в стандарт нет).	C
19	ERROR	EX	0= чисто 1= ошибка	Ошибка карты, относящаяся к командам или при выполнении последней команды (в стандарт нет).	C
18	Резерв				
17	Резерв				
16	CID/ CSD_OVERWRITE	EX	0= чисто 1= ошибка	Одна из ошибок: – Регистр CID уже записан – Неизменяемая часть CSD не соответствует данным карты – Попытка реверса копии или постоянно защищённых битов	C
15	WP_ERASE_SKIP	EX	0=всё 1=часть	Стирание только незащищённых частей памяти.	C
14	CARD_ECC_DISABLE D	SX	0= выкл. 1= вкл.	Команда выполнилась без внутренней коррекции данных (ECC).	A
13	ERASE_RESET	-	0= нету 1= стоит	Сброс последовательности стирания из-за неверной последовательности команд (не CMD35, CMD36, CMD38 или CMD13)	C
12:9	CURRENT_STATE	SR	0 = Idle 1 = Ready 2 = Ident 3 = Stby 4 = Tran 5 = Data 6 = Rcv 7 = Prg 8 = Dis 9 = Btst 10-15 = резерв	Состояние карты при приёме команды. Хост увидит изменение состояния на следующей команде.	B
8	READY_FOR_DATA	SR	0= не готов 1 = готов	Соответствует пустоте буфера.	-
7	SWITCH_ERROR	EX	0= чисто 1= ошибка	По команде SWITCH карта не перешла в требуемый режим	B

Биты	Имя	Тип	Значение	Описание	Чистка
6	Резерв				
5	APP_CMD	SR	0 = Выкл. 1 = Вкл.	Карта подождёт ACMD или команда воспринималась как ACMD.	C
4	Резерв для карт SD I/O				
3	AKE_SEQ_ERROR	ER	0= чисто 1= ошибка	Ошибка последовательности аутентификации.	C
2	Резерв для особых команд.				
1	Резерв для тестов производителя.				
0					

31.4.12.Регистр состояния SD

Содержит собственную информацию SD карт памяти одним блоком из 512 бит. Передаётся хосту SDIO по команде ACMD13 (CMD55 и за ней CMD13). ACMD13 посылается только в режиме Передачи (карта выбрана).

В *Таблице 146*. используются следующие аббревиатуры:

Тип:

- E: бит ошибки
- S: бит состояния
- R: обнаружено и установлено для этого ответа
- X: обнаружено и установлено при исполнении команды. Для чтения этих битов хост SDIO должен опрашивать карту командой.

Чистка:

- A: в соответствии с текущим состоянием карты
- B: всегда относится к предыдущей команде. Приём правильной команды чистит их (с задержкой в одну команду)
- C: чистка чтением

Таблица 159. Состояние карты SD

Биты	Имя	Тип	Значение	Описание	Чистка
511: 510	DAT_BUS_WIDTH	SR	00’= 1 бит (по умолчанию) ‘01’= резерв ‘10’= 4 бита ‘11’= резерв	Shows the currently defined databus width that was defined by SET_BUS_WIDTH command	A
509	SECURED_MODE	SR	0’= Не в режиме ’1’= В безопасном режиме	Card is in Secured Mode of operation (refer to the “SD Security Specification”).	A
508: 496	Резерв.				
495: 480	SD_CARD_TYPE	SR	’00xxh’= SD Memory Card (’x’= не волнуют). Сейчас есть: ’0000’= Регулярная SD RD/WR. ’0001’= SD ROM	In the future, the 8 LSBs will be used to define different variations of an SD memory card (each bit will define different SD types). The 8 MSBs will be used to define SD Cards that do not comply with current SD physical layer specification.	A
479: 448	SIZE_OF_PROTE CT ED_AREA	SR	Размер защищённой области (См. ниже)	(См. ниже)	A
447: 440	SPEED_CLASS	SR	Класс скорости. (См. ниже)	(См. ниже)	A
439: 432	PERFORMANCE_ MOVE	SR	Скорость перемещения с шагом [MB/s]. (См. ниже)	(См. ниже)	A
431: 428	AU_SIZE	SR	Размер AU (См. ниже)	(См. ниже)	A
427: 424	Резерв.				
423: 408	ERASE_SIZE	SR	Число стираемых за раз AU	(См. ниже)	A
407: 402	ERASE_TIMEOUT	SR	Таймаут стирания областей из UNIT_OF_ERASE_AU	(См. ниже)	A
401: 400	ERASE_OFFSET	SR	Фиксированная добавка к времени стирания.	(См. ниже)	A

Биты	Имя	Тип	Значение	Описание	Чистка
399:312	Резерв.				
311:0	Резерв для производителя.				

SIZE_OF_PROTECTED_AREA

У карт стандартной плотности ёмкость защищённой области задаётся как:

Защищённая область = SIZE_OF_PROTECTED_AREA * MULT*BLOCK_LEN.

SIZE_OF_PROTECTED_AREA определяется в единицах MULT*BLOCK_LEN.

У карт высокой плотности ёмкость защищённой области задаётся как:

Защищённая область = SIZE_OF_PROTECTED_AREA

SIZE_OF_PROTECTED_AREA определяется в байтах.

SPEED_CLASS

Это 8-бит поле, указывающее класс скорости устройства и определяемое как $P_W/2$ (где P_W это производительность записи).

Таблица 160. Класс скорости

SPEED_CLASS	Определение
00h	Class 0
01h	Class 2
02h	Class 4
03h	Class 6
04h – FFh	Резерв

PERFORMANCE_MOVE

Это 8-бит поле показывает P_M (производительность пересылки) с шагом 1 [MB/sec]. Если карта не использует блоки записи (RU), то P_M должно рассматриваться как бесконечность.

PERFORMANCE_MOVE	Определение
00h	Не определено
01h	1 [MB/sec]
02h	02h 2 [MB/sec]
---------	---------
FEh	254 [MB/sec]
FFh	Бесконечность

AU_SIZE

Это 4-бит поле показывает размер AU как степень 2 от 16 KB.

AU_SIZE	Определение
00h	Not defined
01h	16 KB
02h	32 KB
03h	64 KB
04h	128 KB
05h	256 KB
06h	512 KB
07h	1 MB
08h	2 MB
09h	4 MB
Ah – Fh	Резерв

Максимальный размер AU зависит от ёмкости карты. Карта может содержать любой размер AU от размера RU до максимального AU.

Ёмкость	16-64 MB	128-256 MB	512 MB	1-32 GB
Макс. размер AU	512 KB	1 MB	2 MB	4 MB

ERASE_SIZE

Это 16-бит поле с N_{ERASE}. После стирания N_{ERASE} числа AU берётся таймаут из ERASE_TIMEOUT. Хост должен определять правильное число стираемых за одну операцию AU, чтобы правильно показывать прогресс стирания. Если поле равно 0, то таймаут не вычисляется.

ERASE_SIZE	Определение
0000h	Таймаут не вычисляется.
0001h	1 AU
0002h	2 AU
0003h	3 AU
---------	---------
FFFFh	65535 AU

ERASE_TIMEOUT

Это 6-поле содержит T_{ERASE}, значение таймаута от смещения после стирания ERASE_SIZE AU. Диапазон ERASE_TIMEOUT определяет до 63 секунд. Производитель может выбрать любую комбинацию ERASE_SIZE и ERASE_TIMEOUT. Определение ERASE_TIMEOUT определяет ERASE_SIZE.

ERASE_TIMEOUT	Определение
0	Таймаут не вычисляется.
1	1 [sec]
2	2 [sec]
3	3 [sec]
---------	---------
63	63 [sec]

ERASE_OFFSET

Это 2-бит поле содержит T_{OFFSET}. Бессмысленно при нулевых ERASE_SIZE и ERASE_TIMEOUT.

ERASE_OFFSET	Определение
0h	0 [sec]
1h	1 [sec]
2h	2 [sec]
3h	3 [sec]

31.4.13.Режим SD I/O

Прерывание SD I/O

Карта SD I/O может прерывать модуль MultiMediaCard/SD с помощью ножки 8 (SDIO_D1) SD интерфейса при работе в 4-бит режиме SD. Эта функция необязательная для карт и их функций. Прерывание SD I/O чувствительно по уровню, то есть линия прерывания должна удерживаться низкой до её распознавания о обработки модулем MultiMediaCard/SD или истечения периода прерывания. После обработки прерывания модуль MultiMediaCard/SD операцией записи снимает соответствующий бит состояния внутреннем регистре карты SD I/O. Все линии данных (SDIO_D[3:0]) должны иметь внешние резисторы подпорки. Модуль MultiMediaCard/SD передаёт уровень 8 ножки (SDIO_D/IRQ) на детектор прерывания только во время периода прерывания, всё остальное время он на неё плюёт.

Период прерывания работает при работе памяти и I/O. При работе с одним блоком и множественными передачами период прерывания определяется по разному.

Задержка и восстановление SD I/O

При работе с некоторыми многофункциональными картами модуль MMC/SD может временно приостанавливать передачу данных и освобождать шину для обслуживания более приоритетных функций. Работа возобновляется с прерванного места. Для этого модуль MMC/SD:

1. Определяет функцию, использующую линии SDIO_D [3:0]
2. Требует задержки низкоприоритетной или медленной передачи
3. Ждёт завершения операции задержки

4. Начинает высокоприоритетную передачу
5. Ждёт её завершения
6. Восстанавливает задержанную передачу

SD I/O ReadWait

Необязательная операция ReadWait (RW) (IO_RW_EXTENDED, CMD53) определена только в 1-бит и 4-бит режимах SD. Ей модуль MMC/SD говорит карте, что это чтение нескольких регистров должно приостановить передачу данных карты для посылки команд другим функциям карты SD I/O. Поддержка протокола ReadWait отмечена в регистрах карты. Времянка ReadWait основана на периоде прерывания.

31.4.14.Команды и ответы

Общие и специфичные команды

В стандарте определены два типа команд: общие (GEN_CMD) и специфичные для конкретной системы (ACMD).

Команда ACMD предваряется посылкой команды APP_CMD (CMD55) и имеет структуру регулярной команды MultiMediaCard и может иметь тот же номер CMD. Если после APP_CMD (CMD55) приходит не ACMD, то выполняется обычная команда. Например, если у карты есть определение SD_STATUS (ACMD13), и она приходит сразу после APP_CMD (CMD55), то выполняется SD_STATUS (ACMD13). Но, если после APP_CMD (CMD55) приходит CMD7, а определения для ACMD7 нет, то выполняется стандартная (SELECT/DESELECT_CARD) CMD7.

Для выполнения ACMD карты SD хост:

1. Шлёт APP_CMD (CMD55)

   Карта отвечает, что бит APP_CMD стоит и ожидается ACMD.

2. Шлёт нужную ACMD

   Карта отвечает, что бит APP_CMD стоит и принята ACMD. Если послана не ACMD, то она выполняется как обычная и бит APP_CMD остаётся чистым.

   Присланная недопустимая команда (ни ACMD, ни CMD) обрабатывается обычным путём.

Передача по шине для GEN_CMD соответствует чтению или записи одного блока (WRITE_BLOCK, CMD24 или READ_SINGLE_BLOCK,CMD17). В этом случае, аргумент определяет направление передачи, а не адрес, и блок данных имеет формат и назначение от производителя.

Перед посылкой GEN_CMD (CMD56) карта должна быть выбрана (в состоянии Передачи state). Размер блока данных определён командой SET_BLOCKLEN (CMD16). Ответ на GEN_CMD (CMD56) выдаётся в формате R1b.

Типы команд

Оба вида команд разделены на 4 типа:

- **широковещательная (BC)**: всем картам, отвечать не нужно.
- **широковещательная с ответом (BCR):** всем картам, ответы принимаются одновременно от всех карт.
- **адресная (point-to-point) (AC):** выбранной карте без передачи данных по линиям SDIO_D.
- **адресная (point-to-point) с передачей данных (ADTC):** выбранной карте с передачей данных по линиям SDIO_D.

Форматы команд

См. *Таблицу 151*.

Команды модуля MultiMediaCard/SD

Таблица 167. Блоковая запись

CMD	Тип	Аргумент	Формат ответа	Имя	Описание
CMD23	ac	[31:16] = 0 [15:0] число блоков	R1	SET_BLOCK_COUNT	Число блоков для передачи по следующей команде.
CMD24	adtc	[31:0] адрес данных	R1	WRITE_BLOCK	Пишет блок длиной из команды SET_BLOCKLEN.
CMD25	adtc	[31:0] адрес данных	R1	WRITE_MULTIPLE_BLOCK	Постоянно пишет блоки до получения команды STOP_TRANSMISSION или исчерпания заданного числа блоков.
CMD26	adtc	[31:0] заполн.	R1	PROGRAM_CID	Пишет регистр идентификации карты. Выдаётся единожды и обычно производителем.
CMD27	adtc	[31:0] заполн.	R1	PROGRAM_CSD	Пишет изменяемые биты CSD.

Таблица 168. Блоковая запись защиты (если есть у карты)

CMD	Тип	Аргумент	Формат ответа	Имя	Описание
CMD28	ac	[31:0] адрес данных	R1b	SET_WRITE_PROT	Ставит бит защиты записи адресованной группы. Свойства защиты описаны в специфичных данных (WP_GRP_SIZE).
CMD29	ac	[31:0] адрес данных	R1b	CLR_WRITE_PROT	Снимает бит защиты записи адресованной группы.
CMD30	adtc	[31:0] адрес данных защиты	R1	SEND_WRITE_PROT	Запрос на отсылку состояния битов защиты.
CMD31	Резерв				

Таблица 169. Команды стирания

CMD	Тип	Аргумент	Формат ответа	Имя	Описание
CMD32 ... CMD34	Резерв. Это команды старых версий MultiMediaCard.				
CMD35	ac	[31:0] адрес данных	R1	ERASE_GROUP_START	Ставит адрес первой группы диапазона стирания.
CMD36	ac	[31:0] адрес данных	R1	ERASE_GROUP_END	Ставит адрес последней группы диапазона стирания.
CMD37	Резерв. Это команды старых версий MultiMediaCard.				
CMD38	ac	[31:0] заполн.	R1	ERASE	Стереть выбранные блоки.

Таблица 170. Команды режима I/O

CMD	Тип	Аргумент	Формат ответа	Имя	Описание
CMD39	ac	[31:16] RCA [15:15] флаг записи [14:8] адрес регистра [7:0] данные	R4	FAST_IO	Чтение/запись 8-бит регистра карты. При стоящем флаге записи пишет в адресованный регистр данные из команды, при чтении данные из адресованного регистра выдаются в ответе R4. В стандарте MultiMediaCard этой команды нет.
CMD40	bcr	[31:0] заполн.	R5	GO_IRQ_STATE	Переключает систему в режим прерывания.
CMD41	Резерв				

Таблица 171. Замок карты

CMD	Тип	Аргумент	Формат ответа	Имя	Описание
CMD42	adtc	[31:0] заполн.	R1b	LOCK_UNLOCK	Ставит/снимает пароль или замок карты. Размер блока данных из команды SET_BLOCK_LEN.
CMD43 ... CMD54	Резерв				

Таблица 159. Специфичные команды

CMD	Тип	Аргумент	Формат ответа	Имя	Описание
CMD55	ac	[31:16] RCA [15:0] заполн.	R1	APP_CMD	Говорит карте, что следующая команда специфичная
CMD56	adtc	[31:1] заполн. [0]: RD/WR	-	-	Чтение/запись блока из/в карту для специфичных команд или общего назначения. Размер блока данных из команды SET_BLOCK_LEN.
CMD57 ... CMD59	Резерв				
CMD60 ... CMD63	Резерв для производителя				

31.5. Форматы ответа

Все ответы посылаются по линии команд SDIO_CMD MCCMD. Передача начинается с левого бита кодового слова ответа. Длина кода зависит от типа ответа.

Ответ начинается с стартового бита (всегда 0) и бита направления передачи (карта = 0). Переменное значение в таблицах обозначено как X. Все ответы, кроме R3, сопровождаются CRC. Командное слово команд завершается конечным битом (всегда 1).

Есть пять типов команд:

31.5.1. R1 (нормальный ответ)

Длина = 48 бит.

Позиция бита	Ширина	Значение	Описание
47	1	0	Бит старта
46	1	0	Бит передачи
[45:40]	6	X	Индекс команды
[39:8]	32	X	Состояние карты
[7:1]	7	X	CRC7
0	1	1	Бит конца

31.5.2. R1b

Идентичен R1 с возможной передачей сигнала занятости по линии данных. Карта может стать занятой после приёма команд, в зависимости от своего состояния перед её приёмом.

31.5.3. R2 (регистры CID, CSD)

Длина = 136 бит. Содержимое регистра CID отсылается в ответ на команды CMD2 и CMD10. Содержимое регистра CSD отсылается в ответ на CMD9. Передаются только биты [127...1] CID и CSD, резервный бит [0] заменяется битом конца ответа. Карта показывает процесс стирания, удерживая низким MCDAT. Стирание может быть долгим и карту можно отключить командой CMD7.

Позиция бита	Ширина	Значение	Описание
135	1	0	Бит старта
134	1	0	Бит передачи
[133:128]	6	111111’	Индекс команды
[127:1]	127	X	Состояние карты
0	1	1	Бит конца

31.5.4. R3 (регистр OCR)

Длина: 48 бит. Состояние регистра OCR отсылается в ответ на команду CMD1. Уровень кодирования: ограниченно окно напряжения = низкий, карта занята = низкий.

Позиция бита	Ширина	Значение	Описание
47	1	0	Бит старта
46	1	0	Бит передачи
[45:40]	6	111111’	Резерв
[39:8]	32	X	Регистр OCR
[7:1]	7	1111111’	Резерв
0	1	1	Бит конца

31.5.5. R4 (Быстрый I/O)

Длина: 48 бит. Поле аргумента содержит RCA адресованной карты, адрес регистра и его данные.

Позиция бита		Ширина	Значение	Описание
47		1	0	Бит старта
46		1	0	Бит передачи
[45:40]		6	100111’	CMD39
[39:8] Аргумент	[31:16]	16	X	RCA
	[15:8]	8	X	Адрес регистра
	[7:0]	8	X	Содержимое регистра
[7:1]		7	X	CRC7
0		1	1	Бит конца

31.5.6. R4b

Только SD I/O: карта SDIO после CMD5 ответит одним R4.

Формат:

Позиция бита		Ширина	Значение	Описание
47		1	0	Бит старта
46		1	0	Бит передачи
[45:40]		6	X	Резерв
[39:8] Аргумент	39	16	X	Карта готова
	[38:36]	3	X	Число функций I/O
	35	1	X	Память есть
	[34:32]	3	X	Заполнитель
	[31:8]	24	X	I/O ORC
[7:1]		7	X	Резерв
0		1	1	Бит конца

Команда CMD5 включает I/O часть карты. Она остаётся включённой вплоть до сброса, цикла питания или команды CMD52 с записью в I/O сброс. Причём карты только с памятью SD могут отвечать на CMD5. Они должны ответить *Память есть* = 1 and *Число функций I/O* = 0. Карты памяти SD спецификации 1.0 должны опознавать CMD5 как недопустимую команду и не отвечать на неё. Осведомлённый хост I/O пошлёт CMD5. По полученному ответу R4, если есть, он определит конфигурацию карты.

31.5.7. R5 (запрос прерывания)

Только MultiMediaCard. Длина: 48 бит. Если ответ выдаётся хостом, то поле RCA аргумента будет равно 0x0.

Позиция бита		Ширина	Значение	Описание
47		1	0	Бит старта
46		1	0	Бит передачи
[45:40]		6	101000’	CMD40
[39:8] Аргумент	[31:16]	16	X	RCA [31:16] победившей карты или хоста
	[15:0]	16	X	Не определено. Можно использовать для данных IRQ
[7:1]		7	X	CRC7

Позиция бита	Ширина	Значение	Описание
0	1	1	Бит конца

31.5.8. R6

Только SD I/O. Нормальный ответ карты памяти на CMD3.

Позиция бита		Ширина	Значение	Описание
47		1	0	Бит старта
46		1	0	Бит передачи
[45:40]		6	101000’	CMD40
[39:8] Аргумент	[31:16]	16	X	RCA [31:16] победившей карты или хоста
	[15:0]	16	X	Не определено. Можно использовать для данных IRQ
[7:1]		7	X	CRC7
0		1	1	Бит конца

У карт только I/O биты состояния [23:8] другие, они содержат:

- Бит [15] COM_CRC_ERROR
- Бит [14] ILLEGAL_COMMAND
- Бит [13] ERROR
- Бит [12:0] Резерв.

31.6. Специфичные операции SDIO I/O

Это:

- SDIO ReadWait сигналом SDIO_D2
- SDIO ReadWait остановкой тактов SDIO_CK
- SDIO задержка/восстановление (чтение и запись)
- SDIO прерывания

SDIO поддерживает эти операции только при стоящем бите SDIO_DCTRL[11], кроме задержки чтения, для которой аппаратная поддержка не нужна.

31.6.1. SDIO I/O ReadWait сигналом SDIO_D2

Интервал ReadWait можно запустить перед приёмом первого блока данных: если путь данных включён (стоит бит SDIO_DCTRL[0]), разрешены специфичные операции (бит SDIO_DCTRL[11]), ReadWait стартует (SDI0_DCTRL[10]=0 и SDI_DCTRL[8]=1) и направление от карты к SDIO (SDIO_DCTRL[1] = 1), то DPSM напрямую идёт из Простоя в ReadWait. В ReadWait DPSM уводит SDIO_D2 в 0 после 2 тактов SDIO_CK. В этом состоянии при установке бита RWSTOP (SDIO_DCTRL[9]) DPSM остаётся в Ожидании ещё на 2 такта SDIO_CK для переключения SDIO_D2 в 1 на 1 такт (в соответствии с спецификацией SDIO). Затем DPSM снова ждёт приёма данных от карты.

DPSM не стартует интервал ReadWait во время приёма блока даже при стоящем бите старта: он запустится после приёма CRC. Для запуска новой операции ReadWait бит RWSTOP надо чистить. Во время интервала ReadWait прерывания SDIO ищут на SDIO_D1.

31.6.2. SDIO ReadWait остановкой тактов SDIO_CK

Если карта SDIO не поддерживает предыдущий метод ReadWait, то его можно добиться остановкой SDIO_CK (SDIO_DCTRL должен стоять как в *Секции 22.6.1*, но SDIO_DCTRL[10] =1): DSPM останавливает SDIO_CK на два такта после конечного бита принимаемого блока и пускает его снова после установки бита старта ReadWait.

После такой остановки SDIO_CK карте можно выдавать команды. Во время интервала ReadWait прерывания SDIO ищут на SDIO_D1.

31.6.3. Задержка/ восстановление SDIO

SDIO может задержать работающую операцию записи в карту. Бит SDIO_CMD[11] показывает CPSM, что операция задержана. CPSM анализирует ответ и при получении от карты ACK он извещает DPSM, который после приёма CRC текущего блока уходит в Простой.

Аппаратура не сохраняет число оставшихся блоков для завершения задержаной операции.

Операция записи может быть задержана программно простым отключением DPSM (SDIO_DCTRL[0]=0) при получении ACK задержанной команды от карты. DPSM идёт в Простой.

Задержка чтения: DPSM ждёт в состоянии Wait_r, тогда как задерживаемая функция посылает полный пакет непосредственно перед остановкой передачи данных. Система продолжает чтение RxFIFO до опустения FIF0 и DPSM автоматически уходит в Простой.

31.6.4. Прерывания SDIO

Прерывания SDIO ищут на линии SDIO_D1 при установке бита SDIO_CMD[11].

31.7. Специфичные операции CE-ATA

Это:

- посылка CE-ATA запрещения сигнала завершения команды
- приём от CE-ATA сигнала завершения команды
- извещение CPU о завершении команды CE-ATA битом состояния/прерыванием.

SDIO использует команду CMD61 CE-ATA, если стоит SDIO_CMD[14].

31.7.1. Запрещение сигнала завершения команды

Запрещение сигнала завершения команды посылается через 8 битовых циклов после приёма **короткого** ответа если бита 'Разрешить завершение CMD' , SDIO_CMD[12], нет, а бит 'Без прерывания', SDIO_CMD[13], стоит.

CPSM Идёт в режим Удержания, пишет в регистр сдвига команды последовательность выключения "00001" и 43 в счётчик команды. Через 8 циклов триггер переводит CPSM в состояние Посылки. При достижении счётчиком команды 48 CPSM переходит в Простой не ожидая ответа.

31.7.2. Разрешение сигнала завершения команды

Если стоят биты 'Разрешить завершение CMD' SDIO_CMD[12] и 'Без прерывания' SDIO_CMD[13], то CPSM ждёт сигнала завершения команды в состоянии Waitcpl.

После приёма '0' на линии CMD, CPSM идёт в Простой. Новый команды можно присылать через 7 битовых циклов. Затем, на последние 5 циклов (вне 7) линия CMD уходит в '1' в двухтактном режиме.

31.7.3. Прерывание CE-ATA

Завершение команды отмечается битом состояния SDIO_STA[23]. Снимается он битом очистки SDIO_ICR[23].

Бит SDIO_STA[23] может генерировать прерывание на любой линии, в зависимости от маски SDIO_MASKx[23].

31.7.4. Прекращение CMD61

Если запрещения сигнала завершения команды не посылалось и нужно прервать CMD61, то надо отключить машину состояния команд. Она идёт в Простой и можно посылать CMD12. В это время Запрещение сигнала завершения команды не посылают.

31.8. Аппаратное управление (HW)

Используется для предотвращения исчерпания (режим TX) и переполнения (режим RX) FIFO.

Надо остановить SDIO_CK и стопорнуть машину состояний SDIO. Пока FIFO не может принимать или передавать данные, они останавливаются. Стопорится только машина состояний (SDIOCLK), интерфейс AHB ещё живёт и можно заполнять или опустошать FIFO.

Работа HW разрешается стоящим битом SDIO_CLKCR[14]. После сброса управление HW отключено.

31.9. Регистры SDIO

32-бит регистры доступны через AHB словами.

31.9.1. Управление питанием (SDIO_POWER)

Смещение адреса: 0x00

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																														PWRCTRL	
																														rw	rw

— Биты 31:2 Резерв, не трогать.

— Биты 1:0 **PWRCTRL**: Управление питанием.

Текущее функциональное состояние тактов карты:

00: Выкл: тактов нет.

01: Резерв

10: Резерв, подача

11: Вкл: такты есть.

NB: Между двумя записями в этот регистр должно быть не менее семи тактов HCLK.

31.9.2. Управление тактами (SDIO_CLKCR)

Смещение адреса: 0x04

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																	HWFC_EN	NEGEDGE	WIDBUS		BYPASS	PWRSAV	CLKEN	CLKDIV								
																	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:2 Резерв, не трогать.

— Бит 14 **HWFC_EN**: Разрешение управления HW

0b: Нельзя

1b: Можно

Сигналы прерываний TXFIFOE и RXFIFOF при разрешённом HW описаны в *Секции 22.9.11*.

— Бит 13 **NEGEDGE:** Выбор фазы SDIO_CK

0b: SDIO_CK выдаётся по переднему фронту SDIOCLK

1b: SDIO_CK выдаётся по заднему фронту SDIOCLK

— Биты 12:11 **WIDBUS:** Ширина шины

00: По умолчанию, 1 бит: SDIO_D0

01: 4-бита: SDIO_D[3:0]

10: 8-бит: SDIO_D[7:0]

— Бит 10 **BYPASS:** Шунт делителя тактов

0: Выкл: SDIOCLK делится на CLKDIV для выдачи SDIO_CK.

1: Вкл: SDIOCLK напрямую идёт на SDIO_CK.

— Бит 9 **PWRSAV**: Экономия питания

Такты SDIO_CK при простое шины:

0: SDIO_CK включены всегда

1: SDIO_CK включены при активной шине

— Бит 8 **CLKEN:** Разрешение тактов

0: SDIO_CK выключены

1: SDIO_CK включены

— Биты 7:0 **CLKDIV:** Делитель тактов

Делитель входных (SDIOCLK) для получения выходных тактов (SDIO_CK):

Частота SDIO_CK = SDIOCLK / [CLKDIV + 2].

NB: При идентификации SD/SDIO и MultiMediaCard частота SDIO_CK должна быть меньше 400 kHz.

Повысить частоту шины можно только после назначения относительных адресов всем картам.

Между двумя записями в этот регистр должно пройти не менее 7 тактов HCLK.

Остановить SDIO_CK можно на время интервала ReadWait SD I/O карт: тогда SDIO_CLKCR не влияет на SDIO_CK.

31.9.3. Регистр аргумента (SDIO_ARG)

Смещение адреса: 0x08

По сбросу: 0x0000 0000

32-бит аргумент посылаемой карте команды.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDARG																															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:0 **CMDARG:** Аргумент команды
Аргумент пишется в этот регистр до записи кода команды в регистр команды.

31.9.4. Регистр команды (SDIO_CMD)

Смещение адреса: 0x0C

По сбросу: 0x0000 0000

Регистр SDIO_CMD содержит биты индекса и типа команды. Биты индекса посылаются карте, биты типа управляют машиной состояния пути команды (CPSM).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																	CE-ATACMD	nIEN	ENCMDcompl	SDIOSuspend	CPSMEN	WAITPEND	WAITINT	WAITRESP		CMDINDEX						
																	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:15 Резерв, не трогать.

— Бит 14 **ATACMD:** Команда CE-ATA
Если ATACMD стоит, то CPSM передаёт CMD61.

— Бит 13 **nIEN:** Не-разрешение прерываний
Если равен 0, то прерывания в CE-ATA разрешены.

— Бит 12 **ENCMDcompl:** Разрешение завершения команды
Если стоит, то сигнал завершения команды разрешён.

— Бит 11 **SDIOSuspend:** Задержка SD I/O
Если стоит, то передаётся команда задержки (только карты SDIO).

— Бит 10 **CPSMEN:** Разрешение машины состояния пути команд (CPSM)
Если стоит, то CPSM разрешена.

— Бит 9 **WAITPEND:** CPSM ждёт конца передачи данных (CmdPend внутренний сигнал).
Если стоит, то CPSM ждёт конца передачи данных перед посылкой команды.

— Бит 8 **WAITINT:** CPSM ждёт запроса прерывания
Если стоит, то CPSM выключает таймаут команды и ждёт запроса прерывания.

— Биты 7:6 **WAITRESP**: Ожидание ответа
00: Без ответа, ожидается флаг CMDSENT
01: Короткий ответ, ожидается флаг CMDREND или CCRCFAIL
10: Без ответа, ожидается флаг CMDSENT
11: Длинный ответ, ожидается флаг CMDREND или CCRCFAIL

— Биты 5:0 **CMDINDEX:** Индекс посылаемой команды.

NB: Между двумя записями в этот регистр должно пройти не менее 7 тактов HCLK.

MultiMediaCards посылают два вида ответов: короткий (48 бит) и длинный (136 бит). Карты SD и SD I/O посылают только короткий ответ. Устройства CE-ATA посылают только короткий ответ. Аргумент может меняться по типу ответа: это определяется по типу команды.

31.9.5. Регистр отвечаемой команды (SDIO_RESPCMD)

Смещение адреса: 0x10

По сбросу: 0x0000 0000

Регистр SDIO_RESPCMD содержит поле индекса команды последнего принятого ответа. Если ответ не содержит поле индекса команды (длинный ответ или OCR), то содержимое RESPCMD неизвестно, хоть и должно содержать 111111b (резервное поле из ответа).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																										RESPCMD					
																										r	r	r	r	r	r

— Биты 31:6 Резерв, не трогать.

— Биты 5:0 **RESPCMD:** Индекс отвечаемой команды.
Только чтение.

31.9.6. Регистры 1..4 ответа (SDIO_RESPx)

Смещение адреса: (0x10 + (4 × x)); x = 1..4

По сбросу: 0x0000 0000

Регистры SDIO_RESP1/2/3/4 содержат состояние карты из принятого ответа.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CARDSTATUSx																															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

CARDSTATUSx:

Регистр	Короткий ответ	Длинный ответ
SDIO_RESP1	Биты[31:0]	Биты[127:96]
SDIO_RESP2	Не используется	Биты[95:64]
SDIO_RESP3	Не используется	Биты[63:32]
SDIO_RESP4	Не используется	Биты[31:1]0b

Старший значащий бит состояния принимается первым. Младший бит регистра SDIO_RESP3 всегда 0b.

31.9.7. Регистр таймера данных (SDIO_DTIMER)

Смещение адреса: 0x24

По сбросу: 0x0000 0000

32-бит регистр SDIO_DTIMER содержит период таймаута данных (DATATIME) в единицах тактов шины карты. Декрементный счётчик грузит это значение при переходе DPSM в состояние Wait_R или Занят. При обнулении счётчика ставится флаг таймаута.

NB: Регистры таймера и длины данных надо писать до записи регистра управления данными.

31.9.8. Регистр длины данных (SDIO_DLEN)

Смещение адреса: 0x28

По сбросу: 0x0000 0000

32-бит регистр SDIO_DLEN содержит число байтов передаваемых данных (DATALENGTH) в единицах тактов шины карты. Декрементный счётчик грузит это значение при старте передачи.

NB: При блоковой передаче длина передаваемых данных должна быть кратной размеру блока (см. SDIO_DCTRL). Регистры таймера и длины данных надо писать до записи регистра управления данными.

31.9.9. Регистр управления данными (SDIO_DCTRL)

Смещение адреса: 0x2C

По сбросу: 0x0000 0000

Управляет машиной состояния пути данных (DPSM).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				SDIOEN	RWMOD	RWSTOP	RWSTART	DBLOCKSIZE				DMAEN	DTMODE	DTDIR	DTEN
																				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:12 Резерв, не трогать.

— Бит 11 **SDIOEN:** Разрешение функций SD I/O
Если стоит, то DPSM выполняет функции SD I/O.

— Бит 10 **RWMOD:** Режим ReadWait
0: Останов ReadWait по SDIO_D2
1: Работа ReadWait по SDIO_CK

— Бит 9 **RWSTOP:** Останов ReadWait
0: ReadWait активен, если стоит RWSTART
1: Разрешение останова ReadWait при стоящем RWSTART

— Бит 8 **RWSTART:** Запуск ReadWait
Если стоит, то запускается ReadWait.

— Биты 7:4 **DBLOCKSIZE:** Размер блока данных в байтах.
Размер блока = $2^{DBLOCKSIZE}$ байт, значение 1111: резерв.

— Бит 3 **DMAEN:** Разрешение DMA

0: Нельзя. 1: Можно.

— Бит 2 **DTMODE:** Режим передачи данных

0: Блоковая передача

1: Поток или мультибайт SDIO для устройств STM32F10xxx XL-плотности. Поток для устройств STM32F10xxx высокой плотности.

— Бит 1 **DTDIR:** Направление передачи данных

0: От контроллера карте.

1: От карты контроллеру.

— Бит 0 **DTEN:** Разрешение передачи данных

Передача данных начинается при записи 1 в бит DTEN. В зависимости от бита DTDIR, DPSM переходит состояние Wait_S, Wait_R или Readwait если RW Start ставится сразу после начала передачи. После конца передачи чистить его не надо, но для начала новой передачи SDIO_DCTRL нужно обновить.

NB: Между двумя записями в этот регистр должно пройти не менее 7 тактов HCLK.

31.9.10.Счётчик данных (SDIO_DCOUNT)

Смещение адреса: 0x30

По сбросу: 0x0000 0000

При переходе DPSM из Простоя в состояние Wait_R или Wait_S, регистр SDIO_DCOUNT загружается из SDIO_DLEN. При чтении выдаёт число оставшихся байтов передачи, запись не имеет смысла. При обнулении SDIO_DCOUNT DPSM переходит простой и ставится флаг DATAEND.

31–26	25–0
Reserved	DATACOUNT
r	r

— Биты 31:25 Резерв, не трогать.

— Биты 24:0 **DATACOUNT:** Счётчик данных.

NB: Регистр должно читать при завершённом цикле передачи данных.

31.9.11.Регистр состояния (SDIO_STA)

Смещение адреса: 0x34

По сбросу: 0x0000 0000

Только читаемый регистр SDIO_STA содержит два типа флагов:

- Статические (биты [23:22,10:0]): они снимаются записью в регистр очистки прерываний (SDIO_ICR)
- Динамические (биты [21:11]): они отражают состояние нижележащей логики (например, флаги заполнения и опорожнения FIFO).

31–24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	CEATAEND	SDIOIT	RXDAVL	TXDAVL	RXFIFOE	TXFIFOE	RXFIFOF	TXFIFOF	RXFIFOHF	TXFIFOHE	RXACT	TXACT	CMDACT	DBCKEND	STBITERR	DATAEND	CMDSENT	CMDREND	RXOVERR	TXUNDERR	DTIMEOUT	CTIMEOUT	DCRCFAIL	CCRCFAIL
Res.	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

— Биты 31:24 Резерв, не трогать.

— Бит 23 **CEATAEND:** Принят сигнал CE-ATA завершения команды CMD61

— Бит 22 **SDIOIT:** Принято прерывание SDIO

— Бит 21 **RXDAVL:** В приёмном FIFO есть данные

— Бит 20 **TXDAVL:** В передающем FIFO есть данные

— Бит 19 **RXFIFOE:** Приёмный FIFO пуст

— Бит 18 **TXFIFOE:** Передающий FIFO пуст

При разрешённом HW Flow Control, TXFIFOE ставится при наличии в FIFO двух слов.

— Бит 17 **RXFIFOF:** Приёмный FIFO полон

При разрешённом HW Flow Control, RXFIFOF ставится за 2 слова до заполнения FIFO.

— Бит 16 **TXFIFOF:** Передающий FIFO полон

— Бит 15 **RXFIFOHF:** Приёмный FIFO наполовину полон, осталось не менее 8 слов

— Бит 14 **TXFIFOHE:** Передающий FIFO наполовину пуст, можно записать до 8 слов

— Бит 13 **RXACT:** Данные принимаются
— Бит 12 **TXACT:** Данные передаются
— Бит 11 **CMDACT:** Команда передаётся
— Бит 10 **DBCKEND:** Блок данных послан/принят (CRC проверен)
— Бит 9 **STBITERR:** В широком режиме шины во всех сигналах данных бит старта не найден
— Бит 8 **DATAEND:** Конец данных (SDIDCOUNT обнулился)
— Бит 7 **CMDSENT:** Команда послана (отвэт нэ нада)
— Бит 6 **CMDREND:** Ответ команды принят (CRC проверен)
— Бит 5 **RXOVERR:** Переполнение приёмного FIFO
— Бит 4 **TXUNDERR:** Исчерпание передающего FIFO
— Бит 3 **DTIMEOUT:** Таймаут данных
— Бит 2 **CTIMEOUT:** Таймаут ответа команды (64 такта SDIO_CK).
— Бит 1 **DCRCFAIL:** Блок данных послан/принят (сбой CRC)
— Бит 0 **CCRCFAIL:** Ответ команды принят (сбой CRC)

31.9.12.Регистр очистки прерываний (SDIO_ICR)

Смещение адреса: 0x38

По сбросу: 0x0000 0000

Только запись. Запись 1 в бит чистит соответствующий бит в регистре SDIO_STA. Запись 0 ничего не изменяет.

31–24	23	22	21–11	10	9	8	7	6	5	4	3	2	1	0
Reserved	CEATAENDC	SDIOITC	Reserved	DBCKENDC	STBITERRC	DATAENDC	CMDSENTC	CMDRENDC	RXOVERRC	TXUNDERRC	DTIMEOUTC	CTIMEOUTC	DCRCFAILC	CCRCFAILC
	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:24 Резерв, не трогать.
— Бит 23 **CEATAENDC:** Принят сигнал CE-ATA завершения команды CMD61
— Бит 22 **SDIOITC:** Принято прерывание SDIO
— Биты 21:11 Резерв, не трогать.
— Бит 10 **DBCKENDC:** Блок данных послан/принят (CRC проверен)
— Бит 9 **STBITERRC:** В широком режиме шины во всех сигналах данных бит старта не найден
— Бит 8 **DATAENDC:** Конец данных (SDIDCOUNT обнулился)
— Бит 7 **CMDSENTC:** Команда послана (отвэт нэ нада)
— Бит 6 **CMDRENDC:** Ответ команды принят (CRC проверен)
— Бит 5 **RXOVERRC:** Переполнение приёмного FIFO
— Бит 4 **TXUNDERRC:** Исчерпание передающего FIFO
— Бит 3 **DTIMEOUTC:** Таймаут данных
— Бит 2 **CTIMEOUTC:** Таймаут ответа команды (64 такта SDIO_CK).
— Бит 1 **DCRCFAILC:** Блок данных послан/принят (сбой CRC)
— Бит 0 **CCRCFAILC:** Ответ команды принят (сбой CRC)

31.9.13.Регистр маски прерываний (SDIO_MASK)

Смещение адреса: 0x3C

По сбросу: 0x0000 0000

Запись 1 в бит разрешает прерывание по стоящему соответствующему биту в регистре SDIO_STA. Запись 0 запрещает прерывание.

31–24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	CEATAENDIE	SDIOITIE	RXDAVLIE	TXDAVLIE	RXFIFOEIE	TXFIFOEIE	RXFIFOFIE	TXFIFOFIE	RXFIFOHFIE	TXFIFOHEIE	RXACTIE	TXACTIE	CMDACTIE	DBCKENDIE	STBITERRIE	DATAENDIE	CMDSENTIE	CMDRENDIE	RXOVERRIE	TXUNDERRIE	DTIMEOUTIE	CTIMEOUTIE	DCRCFAILIE	CCRCFAILIE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:24 Резерв, не трогать.
— Бит 23 **CEATAENDIE:** Принят сигнал CE-ATA завершения команды CMD61

— Бит 22 **SDIOITIE:** Принято прерывание SDIO
— Бит 21 **RXDAVLIE:** В приёмном FIFO есть данные
— Бит 20 **TXDAVLIE:** В передающем FIFO есть данные
— Бит 19 **RXFIFOEIE:** Приёмный FIFO пуст
— Бит 18 **TXFIFOEIE:** Передающий FIFO пуст
— Бит 17 **RXFIFOFIE:** Приёмный FIFO полон
— Бит 16 **TXFIFOFIE:** Передающий FIFO полон
— Бит 15 **RXFIFOHFIE:** Приёмный FIFO наполовину полон, осталось не менее 8 слов
— Бит 14 **TXFIFOHEIE:** Передающий FIFO наполовину пуст, можно записать до 8 слов
— Бит 13 **RXACTIE:** Данные принимаются
— Бит 12 **TXACTIE:** Данные передаются
— Бит 11 **CMDACTIE:** Команда передаётся
— Бит 10 **DBCKENDIE:** Блок данных послан/принят (CRC проверен)
— Бит 9 **STBITERRIE:** В широком режиме шины во всех сигналах данных бит старта не найден
— Бит 8 **DATAENDIE:** Конец данных (SDIDCOUNT обнулился)
— Бит 7 **CMDSENTIE:** Команда послана (отвэт нэ нада)
— Бит 6 **CMDRENDIE:** Ответ команды принят (CRC проверен)
— Бит 5 **RXOVERRIE:** Переполнение приёмного FIFO
— Бит 4 **TXUNDERRIE:** Исчерпание передающего FIFO
— Бит 3 **DTIMEOUTIE:** Таймаут данных
— Бит 2 **CTIMEOUTIE:** Таймаут ответа команды (64 такта SDIO_CK).
— Бит 1 **DCRCFAILIE:** Блок данных послан/принят (сбой CRC)
— Бит 0 **CCRCFAILIE:** Ответ команды принят (сбой CRC)

31.9.14.Счётчик FIFO SDIO (SDIO_FIFOCNT)

Смещение адреса: 0x48

По сбросу: 0x0000 0000

Регистр `SDIO_FIFOCNT` содержит количество оставшихся слов для записи или чтения из FIFO. Загружается из регистра `SDIO_DLEN` при установке бита `DTEN` в регистре `SDIO_DCTRL` и DPSM в режиме Простоя. Если длина данных не кратна 4, то остаток (1-3 байта) признаётся словом.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								FIFOCOUNT																							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

— Биты 31:24 Резерв, не трогать.
— Биты 23:0 **FIFOCOUNT:** Число оставшихся слов для записи или чтения из FIFO.

31.9.15.Регистры данных FIFO SDIO (SDIO_FIFO)

Смещение адреса: 0x80

По сбросу: 0x0000 0000

Оба FIFO (приёмный и передающий) доступны как 32 смежных регистра по 32 бита по адресу от SDIO base + 0x080 до SDIO base + 0xFC. Так CPU может обращаться к ним командами с множеством операндов.

31.9.16.Карта регистров SDIO

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	SDIO_POWER	Reserved																														PWRCTRL	
0x04	SDIO_CLKCR	Reserved																	HWFC_EN	NEGEDGE	WIDBUS		BYPASS	PWRSAV	CLKEN	CLKDIV							
0x08	SDIO_ARG	CMDARG																															
0x0C	SDIO_CMD	Reserved																	CE-ATACMD	nIEN	ENCMDcompl	SDIOSuspend	CPSMEN	WAITPEND	WAITINT	WAITRESP	CMDINDEX						
0x10	SDIO_RESPC-MD	Reserved																									RESPCMD						
0x14	SDIO_RESP1	CARDSTATUS1																															
0x18	SDIO_RESP2	CARDSTATUS2																															
0x1C	SDIO_RESP3	CARDSTATUS3																															
0x20	SDIO_RESP4	CARDSTATUS4																															
0x24	SDIO_DTIM-ER	DATATIME																															
0x28	SDIO_DLEN	Reserved																	DATALENGTH														
0x2C	SDIO_DCTRL	Reserved																						SDIOEN	RWMOD	RWSTOP	RWSTART	DBLOCKSIZE			DMAEN	DTMODE	DTDIR	DTEN
0x30	SDIO_D-COUNT	Reserved																	DATACOUNT														
0x34	SDIO_STA	Reserved																		CEATAEND	SDIOIT	RXDAVL	TXDAVL	RXFIFOE	TXFIFOE	RXFIFOF	TXFIFOF	RXFIFOHF	TXFIFOHE	RXACT	TXACT	CMDACT	DBCKEND	STBITERR	DATAEND	CMDSENT	CMDREND	RXOVERR	TXUNDERR	DTIMEOUT	CTIMEOUT	DCRCFAIL	CCRCFAIL
0x38	SDIO_ICR	Reserved																		CEATAENDC	SDIOITC	Reserved											DBCKENDC	STBITERRC	DATAENDC	CMDSENTC	CMDRENDC	RXOVERRC	TXUNDERRC	DTIMEOUTC	CTIMEOUTC	DCRCFAILC	CCRCFAILC
0x3C	SDIO_MASK	Reserved																		CEATAENDIE	SDIOITIE	RXDAVLIE	TXDAVLIE	RXFIFOEIE	TXFIFOEIE	RXFIFOFIE	TXFIFOFIE	RXFIFOHFIE	TXFIFOHEIE	RXACTIE	TXACTIE	CMDACTIE	DBCKENDIE	STBITERRIE	DATAENDIE	CMDSENTIE	CMDRENDIE	RXOVERRIE	TXUNDERRIE	DTIMEOUTIE	CTIMEOUTIE	DCRCFAILIE	CCRCFAILIE
0x48	SDIO_FIFO-CNT	Reserved																	FIFOCOUNT														
0x80	SDIO_FIFO	FIF0Data																															

32. Локальная сеть (bxCAN)

32.1. Введение в bxCAN

Basic Extended CAN (**bxCAN**) поддерживает протоколы CAN версий 2.0A и B. Обрабатывает большое число входных сообщений не нагружая CPU и соответствует требованиям приоритетов выходных сообщений. Контроллер CAN имеет все аппаратные средства поддержки временного разделения каналов связи.

32.2. Основные свойства bxCAN

- Поддерживает протокол CAN версии 2.0 A, B Active
- Скорость до 1 Mbit/s
- Поддерживает временное разделение каналов

Передача

- Три передающих почтовых ящика
- Конфигурируемый приоритет передачи
- Метки времени при передаче SOF

Приём

- Два приёмных FIFO с тремя стадиями
- Масштабируемые банки фильтров:
  - 28 общих банков для CAN1 и CAN2 в сетевых устройствах
- Список идентификаторов
- Конфигурируемое переполнение FIFO
- Метки времени при приёме SOF

Временное разделение каналов

- Выключение повторной передачи
- 16-бит независимый таймер
- Метка времени в последних двух байтах передачи

Управление

- Маскируемые прерывания
- Удобное размещение ящиков в адресном пространстве

Парный CAN

- CAN1: Ведущий bxCAN для связи ведомого bxCAN с 512-байт SRAM памятью
- CAN2: Ведомый bxCAN, прямой связи с SRAM памятью нет.
- Две ячейки bxCAN имеют общую 512-байт SRAM память

В устройствах разной плотности USB и CAN используют одну 512-байт SRAM память и не могут работать параллельно. USB и CAN могут работать в одной системе, но не одновременно.

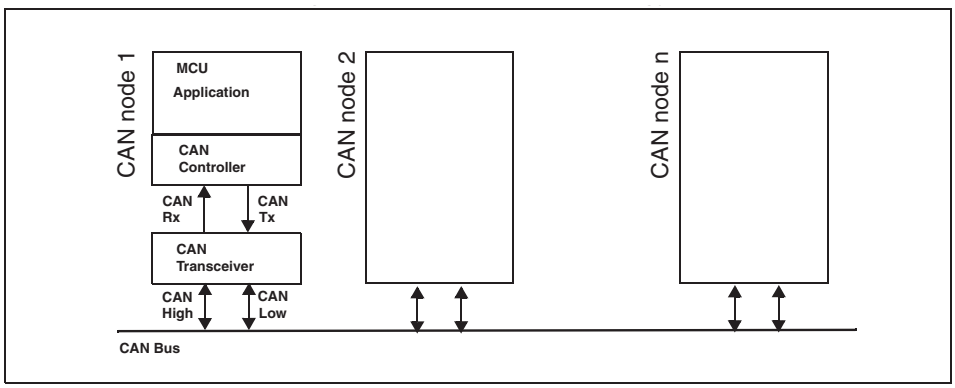
32.3. Общее описание bxCAN

Сегодняшние системы CAN содержат всё увеличивающееся число узлов в сети и часто несколько сетей, объединённых через шлюзы. Число сообщений в системе значительно возросло, и к ним добавились сообщения Диагностики и Управления сетью.

- Нужен улучшенный механизм фильтрации всех типов сообщений. Более того, прикладные задачи требуют всё больше времени CPU, так что надо сокращать затраты на приём сообщений.
- Схема приёмного FIFO отпускает CPU больше времени на прикладные задачи без потери сообщений.

Стандартный HLP (Протокол Верхнего уровня), основанный на стандартных драйверах CAN, требует эффективной связи с контроллером CAN.

Рис. 334. Топология сети CAN.



32.3.1. Активное ядро CAN 2.0B

Модуль bxCAN обрабатывает приём и передачу сообщений CAN полностью автономно, полностью поддерживает стандартные (11-бит) и расширенные (29-бит) идентификаторы.

32.3.2. Регистры управления, состояния и конфигурации

Используются для конфигурации параметров CAN, запроса передач, обработки приёма, управления прерываниями и диагностики.

32.3.3. Почтовые ящики Tx

Программа складывает сообщения в три почтовых ящика, порядок отправки определяет Планировщик.

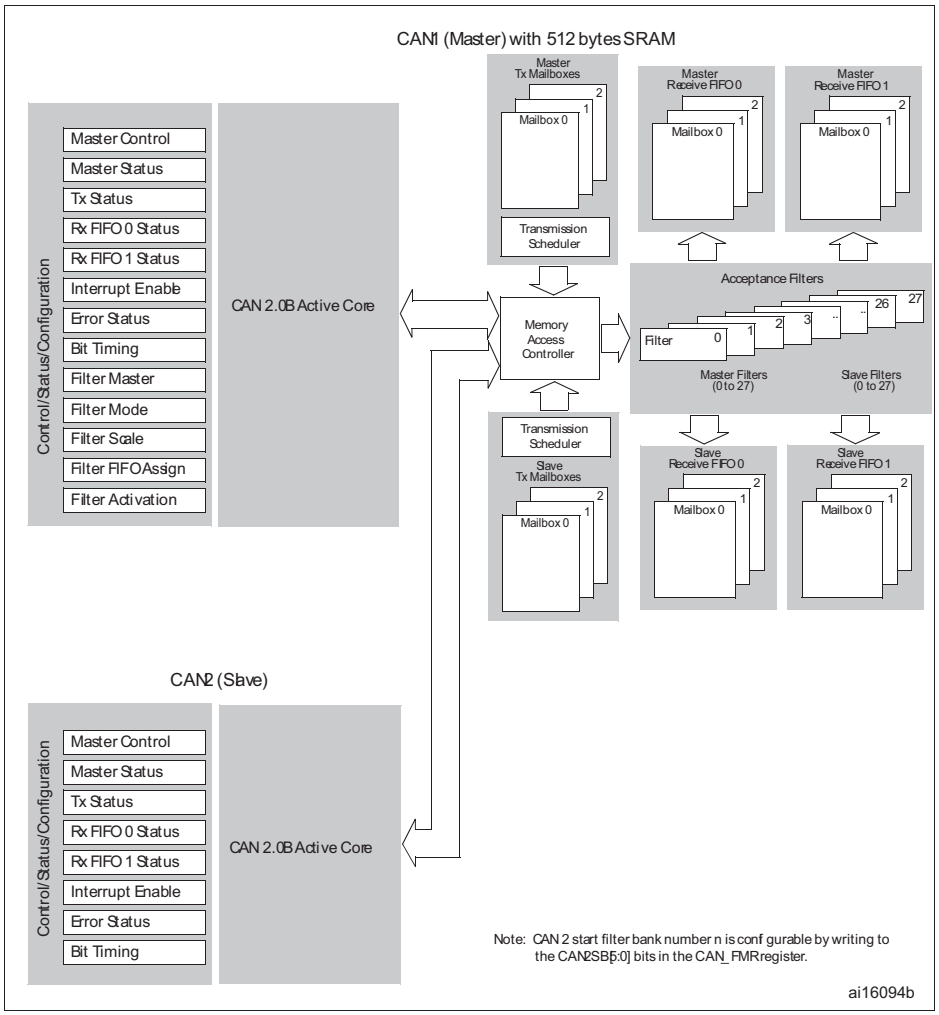
32.3.4. Приёмные фильтры

bxCAN сетевых устройств имеет 28 масштабируемых/конфигурируемых банков фильтров входных сообщений, отбирающих нужные и отбрасывающих чужие. В других устройствах содержится 14 банков фильтров. Аппаратура использует два приёмных FIFO на три сообщения каждый.

Приёмные FIFO

Их два, управляются полностью аппаратно и могут хранить по три входных сообщения.

Рис. 335. Блок-схема парного CAN



32.4. Режимы работы bxCAN

bxCAN работает в трёх режимах: **инициализация**, **нормальный** и **Сон**. После аппаратного сброса bxCAN Спит с активной внутренней подпоркой на `CANTX`. Программа переводит bxCAN в **инициализацию** или **Сон** установкой битов `INRQ` или `SLEEP` в регистре `CAN_MCR`. После установки режима bxCAN ставит биты `INAK` или `SLAK` в регистре `CAN_MSR` и отключает внутреннюю подпорку. В **нормальном** режиме биты `INAK` и `SLAK` чисты. Перед переходом в **нормальный** режим bxCAN всегда **синхронизируется** с шиной CAN. Для этого bxCAN ждёт простоя шины CAN, то есть на `CANRX` появилось 11 последовательных рецессивных битов.

32.4.1. Инициализация

Программная инициализация выполняется в режиме инициализации аппаратуры. Для этого программа ставит бит INRQ в регистре CAN_MCR и ждёт подтверждения от bxCAN (стоящий бит INAK в регистре CAN_MSR). Для выхода из инициализации программа снимает бит INRQ. bxCAN выходит из инициализации после аппаратного снятия бита INAK.

В режиме инициализации все обмены с шиной CAN остановлены и выход CANTX стоит рецессивным (высоким). Переход в инициализацию не изменяет регистры конфигурации.

Для инициализации контроллера CAN программа должна установить регистры времянки битов (CAN_BTR) и опций CAN (CAN_MCR).

Для инициализации регистров, связанных с банками фильтров CAN (режим, масштаб, назначения FIFO, активация и значения фильтров), программа должна поставить бит FINIT в CAN_FMR. Фильтры можно инициализировать и вне режима инициализации.

NB: При FINIT=1, приём CAN деактивирован. Значения фильтров можно изменить снятием битов активации (в регистре CAN_FA1R). Неиспользуемый фильтр лучше держать неактивным (соответствующий бит FACT чистый).

32.4.2. Нормальный режим

Запрос на выход в нормальный режим выдаётся очисткой бита INRQ в регистре CAN_MCR. Теперь надо синхронизироваться с шиной, дождавшись 11 последовательных рецессивных бит (Простой шины). Переключение в нормальный режим подтверждается аппаратным снятием бита INAK в регистре CAN_MSR.

Инициализация значений фильтров не зависит от режима инициализации и делается с неактивными фильтрами (чистый бит FACTx). Масштаб фильтра и конфигурация выполняются до входа в нормальный режим.

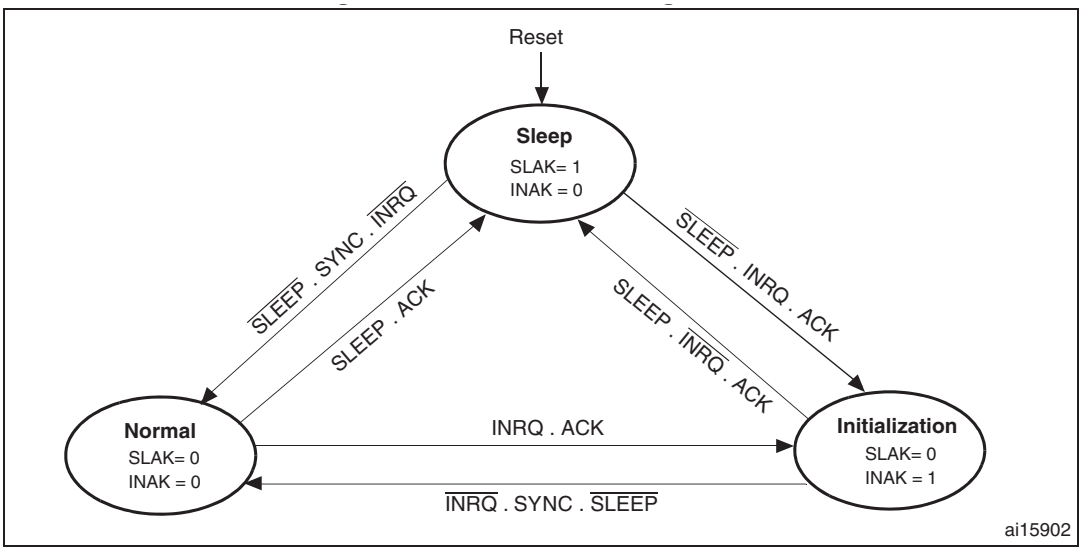
32.4.3. Режим сна (экономный)

Режим сна включается программной установкой бита SLEEP в регистре CAN_MCR. Тогда останавливаются такты bxCAN, но почтовые ящики bxCAN остаются программно доступными. При выдаче запроса на **инициализацию** установкой бита INRQ для спящего bxCAN бит SLEEP надо чистить. bxCAN пробуждается (выходит из Сна) программным снятием бита SLEEP или при появлении активности на шине CAN. Если бит AWUM регистра CAN_MCR стоит, то бит SLEEP снимается аппаратно, иначе его надо чистить программно.

NB: Если прерывание побудки разрешено, (стоит бит WKUIE регистра CAN_IER), то оно выдаётся даже при обнаружении активности шины CAN.

После снятия бита SLEEP bxCAN выходит из Сна после синхронизации с шиной CAN. Бит SLAK снимается аппаратно в конце последовательности выхода has.

Рис. 336. Режимы работы bxCAN.



1. ACK = Состояние ожидания аппаратного подтверждения запросов INAK или SLAK в регистре CAN_MSR
2. SYNC = bxCAN ждёт Простоя шины CAN (11 последовательных рецессивных бит на CANRX)

32.5. Тестовый режим

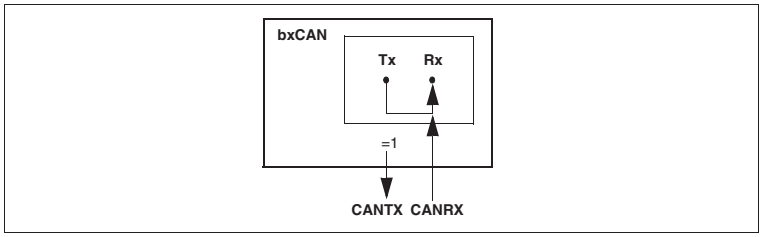
Он выбирается битами SILM и LBKM в регистре CAN_BTR. Их ставят в режиме инициализации bxCAN. После выбора тестового режима нужно отпустить bxCAN в нормальный режим снятием бита INRQ в регистре CAN_MCR.

32.5.1. Режим тишины

bxCAN уходит в Тишину при установке бита SILM в регистре CAN_BTR.

В этом режиме bxCAN может принимать верные фреймы данных и удалённые фреймы, но на шину отсылает только рецессивные биты и не может запустить передачу. Если bxCAN надо послать доминантный бит (ACK, флаг перегрузки, флаг активной ошибки), то он внутренне перенаправляется так, чтобы Ядро CAN Core отследило его, шина CAN может остаться в рецессивном состоянии. Режим тишины может использоваться для анализа передач по шине CAN, не мешая передачей доминантных битов (Биты подтверждения, Фреймы Ошибки).

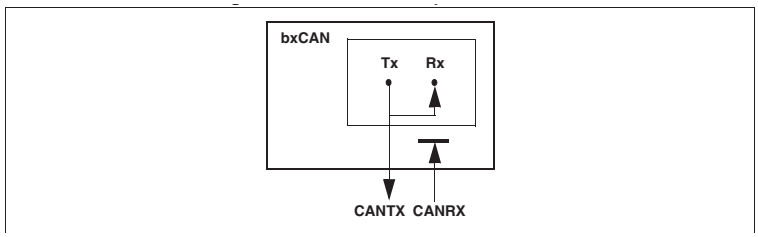
Рис. 337. bxCAN в режиме тишины.



32.5.2. Режим перемычки (loopback)

bxCAN уходит сюда при установке бита LBKM в регистре CAN_BTR. В этом режиме bxCAN воспринимает свои передачи как входные и после фильтрации сохраняется в Приёмном почтовом ящике.

Рис. 338. bxCAN с перемычкой.

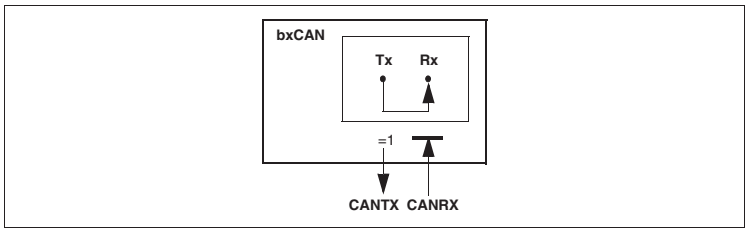


Это само-тестирование. В этом режиме Ядро CAN Core игнорирует ошибки подтверждения (нет доминантного бита в слоте подтверждения фреймов данных/удалённых фреймов). Сигнал Tx передаётся на Rx внутри, реальное значение ножки CANRX отбрасывается. Передаваемые сообщения видны на ножке CANTX.

32.5.3. Тишина + перемычка

Режимы Тишины и Перемычки можно объединить одновременной установкой битов LBKM и SILM в регистре CAN_BTR. Можно проверять себя не влияя на шину CAN. Ножка CANRX отключается от bxCAN, ножка CANTX удерживается рецессивной.

Рис. 339. Тишина + перемычка



32.6. Режим отладки

В режиме отладки (ядро Cortex®-M3 остановлено), bxCAN может работать или стоять в зависимости от:

- Бита DBG_CAN1_STOP для CAN1 и бита DBG_CAN2_STOP для CAN2 в модуле DBG.
- Бита DBF в регистре CAN_MCR.

32.7. Функциональное описание bxCAN

32.7.1. Обработка передачи

Перед запросом передачи битом TXRQ в регистре CAN_TIxR программа должна выбрать **пустой** передающий почтовый ящик, поставить идентификатор, код длины данных (DLC) и сами данные. После выхода ящика из **пустого** состояния, программа лишается права записи в его регистры. Сразу после установки бита TXRQ уходит в состояние удержания и ждёт в очереди приоритетов. Ящик с высшим приоритетом становится **запланированным** для передачи. Передача сообщения из запланированного ящика начинается (состояние **передачи**) при Простое шины CAN. После успешной передачи ящика он снова становится **пустым**. Аппаратура обозначает успешную передачу установкой битов RQCP и TXOK в регистре CAN_TSR.

Причина сбойной передачи показывается битом ALST регистра CAN_TSR в случае Потери Арбитража, и/или битом TERR в случает ошибки передачи.

Приоритет передачи

- ***По идентификатору***

  Если удерживается не один ящик, то порядок передачи определяется по идентификаторам хранящихся сообщений. Сообщение с меньшим приоритетом отправляется первым в соответствии арбитражем протокола CAN. При одинаковых идентификаторах первым планируется ящик с меньшим номером.

- ***По порядку запросов передачи***

  Битом TXFP в регистре CAN_MCR передающие ящики можно превратить в FIFO. При этом порядок приоритетов задаётся порядком запросов. Весьма полезно при сегментированных передачах.

Отмена

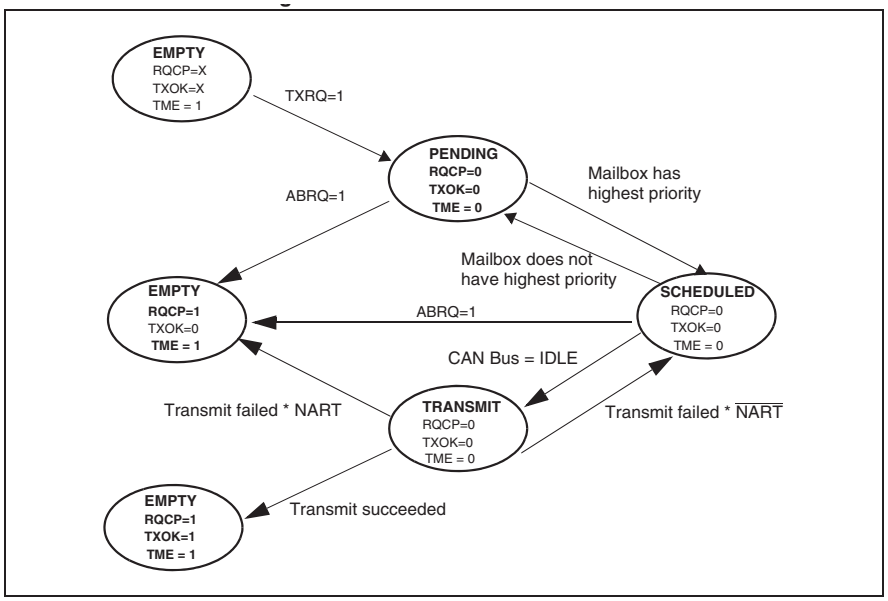
Запрос передачи может быть отменён установкой бита ABRQ в регистре CAN_TSR. В состояниях **удержания** или **планирования** ящик освобождается немедленно. Аборт запроса ящика в состоянии **передачи** может иметь два итога. При успешном завершении передачи ящик становится **пустым** с установкой бита TXOK в регистре CAN_TSR. При сбойной передаче ящик становится **планируемым,** передача прекращается и становится **пустым** с очисткой TXOK. Во всех случаях ящик становится **пустым** не раньше конца текущей передачи.

Не-автоматический повтор передачи

Так исполняются требования стандарта CAN временного разделения каналов. Режим включается установкой бита NART в регистре CAN_MCR. В этом режиме все передачи запускаются единожды.

По концу попытки запрос считается завершённым и ставится бит RQCP в регистре CAN_TSR. Результат передачи указывается в регистре CAN_TSR битами TXOK, ALST и TERR.

Рис. 340. Состояния передающих ящиков.



32.7.2. Временное разделение каналов

В этом режиме запускается внутренний счётчик, создающий Метки Времени, хранящиеся в регистрах CAN_RDTxR/CAN_TDTxR, (для ящиков Rx и Tx). Счётчик инкрементируется по каждому биту CAN (см. *Секцию 24.7.7*.). Считывается по биту Старта Фрейма при приёме и передаче.

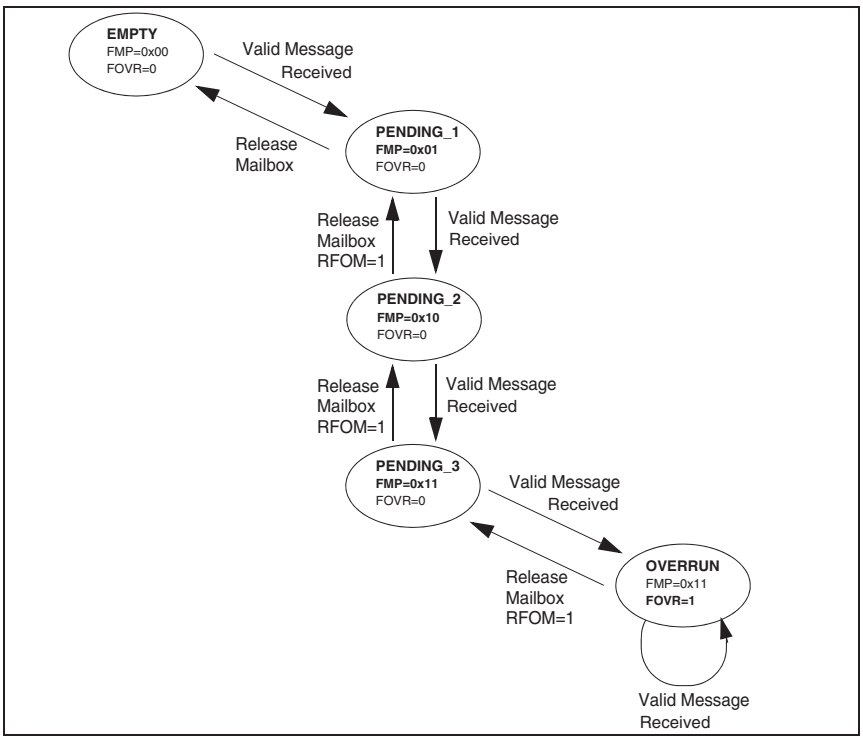
32.7.3. Обработка приёма

Сообщения принимаются в три ящика, организованных как FIFO. Он управляется аппаратно. Программа получает сообщения из выходного ящика FIFO.

Правильное сообщение

Действительным считается безошибочно принятое сообщение, прошедшее через фильтр.

Рис. 341. Состояния приёмного FIFO.



Управление FIFO

После приёма верного сообщения **пустой** FIFO становится **задержанным_1**. Аппаратура сообщает об этом записью 01b в биты FMP[1:0] регистра CAN_RFR. Сообщение доступно в выходном ящике FIFO. Программа читает его и освобождает ящик установкой бита RFOM в регистре CAN_RFR. FIFO снова становится **пустым**. Если в то же время принято новое действительное сообщение, то FIFO остаётся **задержанным_1**, новое сообщение можно достать из выходного ящика.

Следующее верное сообщение при неосвобождённом ящике сохранится в FIFO, который перейдёт в состояние **задержанный_2** (FMP[1:0] = 10b). Запись следующего сообщения в FIFO переведёт его в состояние **задержанный_3** (FMP[1:0] = 11b). Вот тут надо бы наконец освобождать ящик установкой бита RFOM, дабы не терять сообщения. См. *Секцию 24.7.5*.

Переполнение

Следующее верное сообщение при FIFO в состоянии **задержанный_3** переведёт его в **переполнение** и одно сообщение будет потеряно. Аппаратура сообщает об этом установкой бита FOVR в регистре CAN_RFR. Теряемое сообщение определяется конфигурацией FIFO:

- Если замок FIFO отключён (бит RFLM в CAN_MCR сброшен), то заменяется самое старое сообщение в FIFO и доступны самые поздние.
- Если замок FIFO включён (бит RFLM в CAN_MCR стоит), то заменяется самое последнее сообщение в FIFO и доступны самые ранние.

Прерывания приёма

После сохранения сообщения в FIFO изменяются биты FMP[1:0] и при стоящем бите FMPIE в регистре CAN_IER выдаётся запрос прерывания.

При заполнении FIFO (записано третье сообщение) ставится бит FULL в регистре CAN_RFR и при стоящем бите FFIE в регистре CAN_IER выдаётся запрос прерывания.

При переполнении ставится бит FOVR и при стоящем бите FOVIE в регистре CAN_IER выдаётся запрос прерывания.

32.7.4. Фильтрация идентификаторов

Протокол CAN относит идентификатор к содержанию сообщения. Следовательно передатчик вещает для всех приёмников. А приёмник по идентификатору решает, нужно ли ему это сообщение. Нужное копируется в SRAM, ненужное теряется.

Нужность определяется 28 (27-0) конфигурируемыми и масштабируемыми банками фильтров. В иных устройствах их всего 14 (13-0). Каждый банк фильтров состоит из двух 32-бит регистров, CAN_FxR0 CAN_FxR1.

Масштаб ширины

Банки масштабируются независимо. В зависимости от масштаба банки имеют:

- Один 32-бит фильтр для битов STDID[10:0], EXTID[17:0], IDE и RTR.
- Два 16-бит фильтра для битов STDID[10:0], RTR, IDE и EXTID[17:15].

Более того, фильтры можно сконфигурировать в режим маски или списка идентификаторов.

Режим маски

Регистр маски определяет значимые и бестолковые биты регистра идентификатора.

Список идентификаторов

Регистры маски тоже становятся регистрами идентификаторов, удваивая их число. Сравниваются все биты идентификаторов.

Конфигурация масштаба и режима банка фильтров

Для этого есть регистр CAN_FMR, но сначала нужно очистить бит FACT регистра CAN_FAR. Масштаб фильтра определяется соответствующим битом FSCx регистра CAN_FS1R. Режим **списка** или **маски** регистров определяется битами FBMx регистра CAN_FMR.

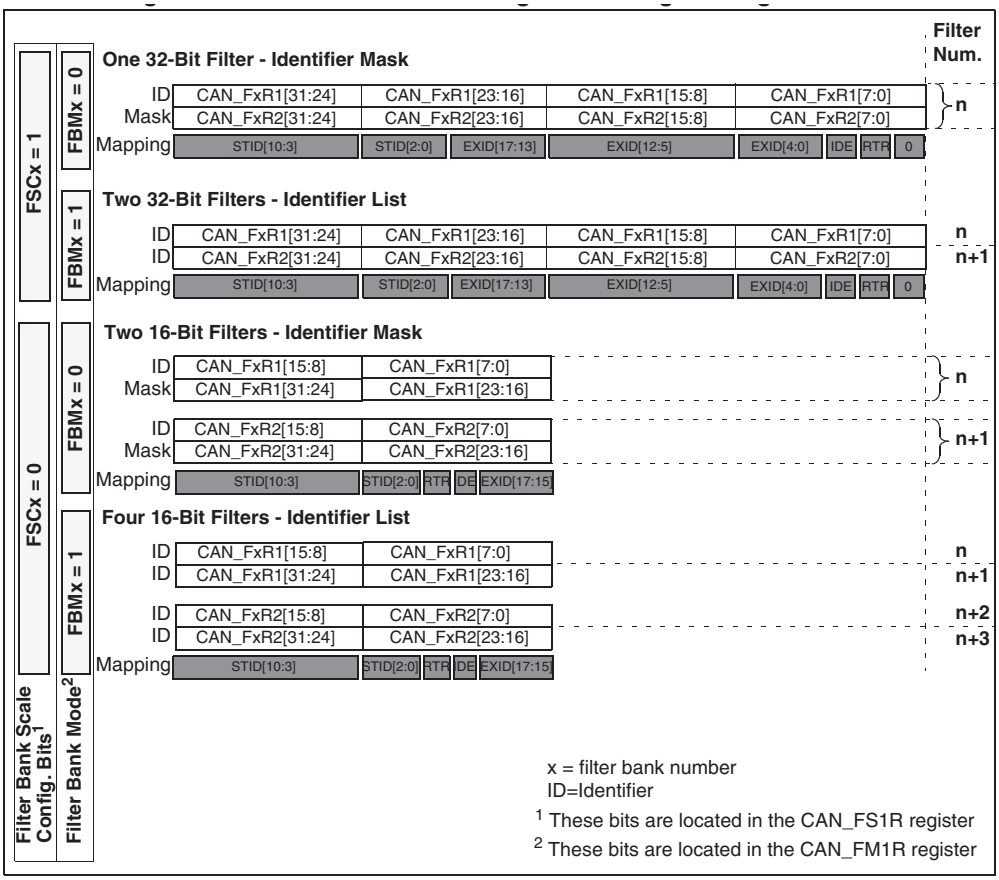
Режим маски выбирает группу идентификаторов.

Режим списка отбирает один идентификатор.

Неиспользуемые фильтры не активируются.

Фильтры внутри банка нумеруются, начиная с 0 до максимума при установленных режиме и масштабе фильтров.

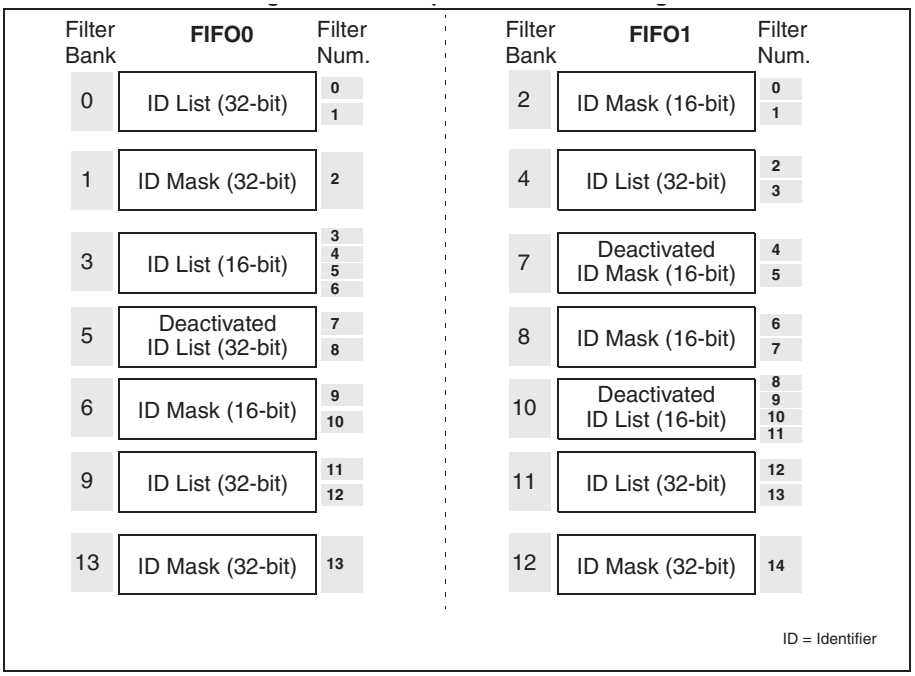
Рис. 342. Организация регистров - масштабирование



Индекс фильтра

Вместе с записью в FIFO полученного сообщения с ним сохраняется индекс пропустившего его фильтра (FMI). Доступен FMI в регистре CAN_RDTxR. Это может существенно облегчить последующую обработку сообщения. Индекс вычисляется в соответствии с правилами приоритета фильтров и без учёта их активности. Кроме того, есть две независимые схемы нумерации фильтров по одному для каждого FIFO.

Рис. 343. Пример нумерации фильтров

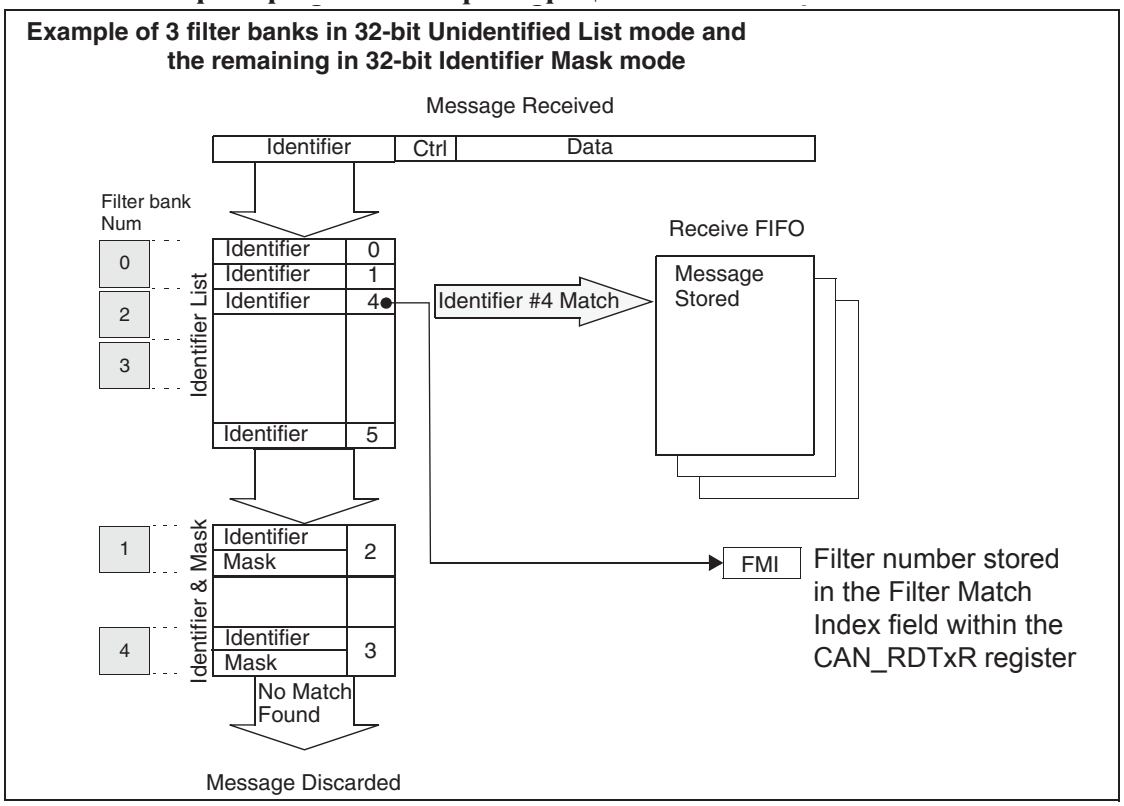


Правила приоритета фильтров

Идентификатор может пройти сразу через несколько фильтров. Тогда FMI выбирается так:

- 32-бит фильтр приоритетнее 16-бит фильтра.
- Из фильтров одинакового масштаба, Список приоритетнее Маски.
- Из фильтров одинакового масштаба и режима, приоритетнее фильтр с меньшим номером.

Рис. 344. Пример механизма фильтрации



В этом примере в FIFO записывается FMI 4.

32.7.5. Память сообщений

Сообщения передаются через почтовые ящики, которые содержат идентификатор, данные, управление и метки времени сообщений.

Передающий ящик

Программа пишет сообщение в пустой ящик. Состояние ящика аппаратно ставится в регистр CAN_TSR.

Таблица 182. Регистры передающего ящика

Смещение от базового адреса ящика	Имя регистра
0	CAN_TIxR
4	CAN_TDTxR
8	CAN_TDLxR
12	CAN_TDHxR

Приёмный ящик

Программа может прочитать полученное сообщение через выходной ящик FIFO. После этого ящик надо освободить битом RFOM в регистре CAN_RFR, разрешая приём следующих сообщений. Индекс фильтра хранится в поле MFMI регистра CAN_RDTxR. 16-бит метка времени хранится в поле TIME[15:0] регистра CAN_RDTxR.

Таблица 183. Регистры приёмного ящика

Смещение от базового адреса ящика	Имя регистра
0	CAN_RIxR
4	CAN_RDTxR
8	CAN_RDLxR
12	CAN_RDHxR

32.7.6. Обработка ошибок

Протокол CAN возлагает это на аппаратуру с использованием Счётчика Ошибок Передачи (поле TEC в регистре CAN_ESR) и Счётчика Ошибок Приёма (поле REC в регистре CAN_ESR), которые инкрементируются и декрементируются в соответствии с условиями ошибки, см. стандарт CAN.

Читая их можно определить стабильность сети. Кроме того, текущее состояние ошибки хранится в регистре CAN_ESR. Битами регистра CAN_IER (бит ERRIE и пр.) можно разрешать выдачу прерываний ошибки.

Отключение шины

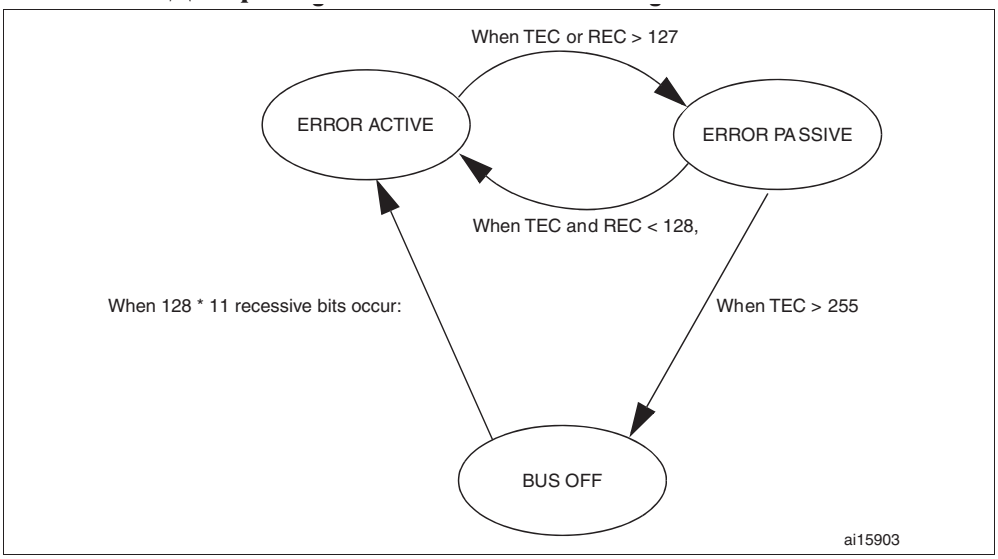
Возникает при TEC больше 255, о чём глаголет бит BOFF регистра CAN_ESR. Всё, шина недоступна для приёма/передачи. bxCAN восстанавливает работу автоматически или по запросу программы, в зависимости от бита ABOM регистра CAN_MCR (ошибка снова активна). Но в любом случае bxCAN должен дождаться последовательности восстановления (128 появлений 11-ти последовательных рецессивных бит на CANRX).

Если ABOM стоит, то bxCAN начнёт последовательность восстановления сразу после отключения шины.

Если ABOM чист, то программа должна запустить последовательность восстановления запросом включения и выхода из инициализации bxCAN.

NB: В режиме инициализации bxCAN на сигнал CANRx не глядит, и завершить последовательность восстановления не может. **Для этого bxCAN должен быть в нормальном режиме**.

Рис. 345. Диаграмма состояний ошибки CAN



32.7.7. Времянка битов

Логика времянки бита следит за последовательной линией шины, синхронизируя точку чтения по фронту стартового бита и ресинхронизируясь по следующим фронтам.

Номинальное время бита разбивается на три сегмента:

- **Сегмент синхронизации** (`SYNC_SEG`): здесь ожидается изменение бита. Фиксированная длительность в один квант времени (1 x t_q).
- **Сегмент бита 1** (`BS1`): точка считывания. Включает `PROP_SEG` и `PHASE_SEG1` стандарта CAN. Длительность программируемая от 1 до 16 квантов времени, но может автоматически удлиняться для компенсации положительного сдвига фазы из-за различия частот разных узлов сети.
- **Сегмент бита 2** (`BS2`): точка передачи или `PHASE_SEG2` стандарта CAN. Длительность программируется от 1 до 8 квантов времени, может укорачиваться для компенсации отрицательного сдвига фазы.

Ширина шага ресинхронизации (`SJW`) определяет верхнюю границу удлинения или сокращения сегментов бита. Программируется от 1 до 4 квантов времени.

Рабочий фронт определяется как первый переход уровня шины из доминантного в рецессивный, а контроллер не передаёт рецессивного бита.

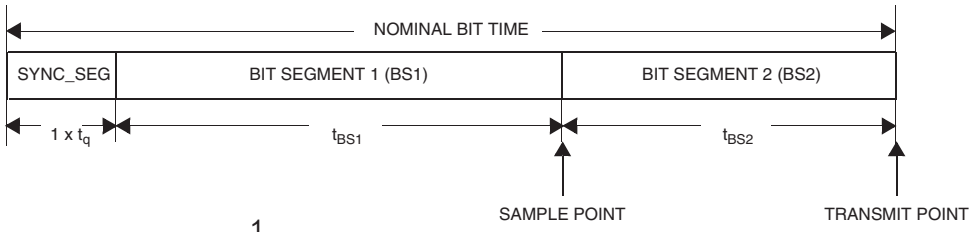
Если рабочий фронт обнаруживается во время `BS1` вместо `SYNC_SEG`, то `BS1` расширяется на `SJW` и точка считывания задерживается.

Если рабочий фронт обнаруживается во время `BS2` вместо `SYNC_SEG`, то `BS2` сокращается на `SJW` точка передачи приближается.

Регистр времянки битов (`CAN_BTR`) доступен для записи только в Дежурном режиме устройства.

См. стандарт ISO 11898.

Рис. 346. Времянка бита



$$BaudRate = \frac{1}{NominalBitTime}$$

$$NominalBitTime = 1 \times t_q + t_{BS1} + t_{BS2}$$

with:

$t_{BS1} = t_q$ x (TS1[3:0] + 1),

$t_{BS2} = t_q$ x (TS2[2:0] + 1),

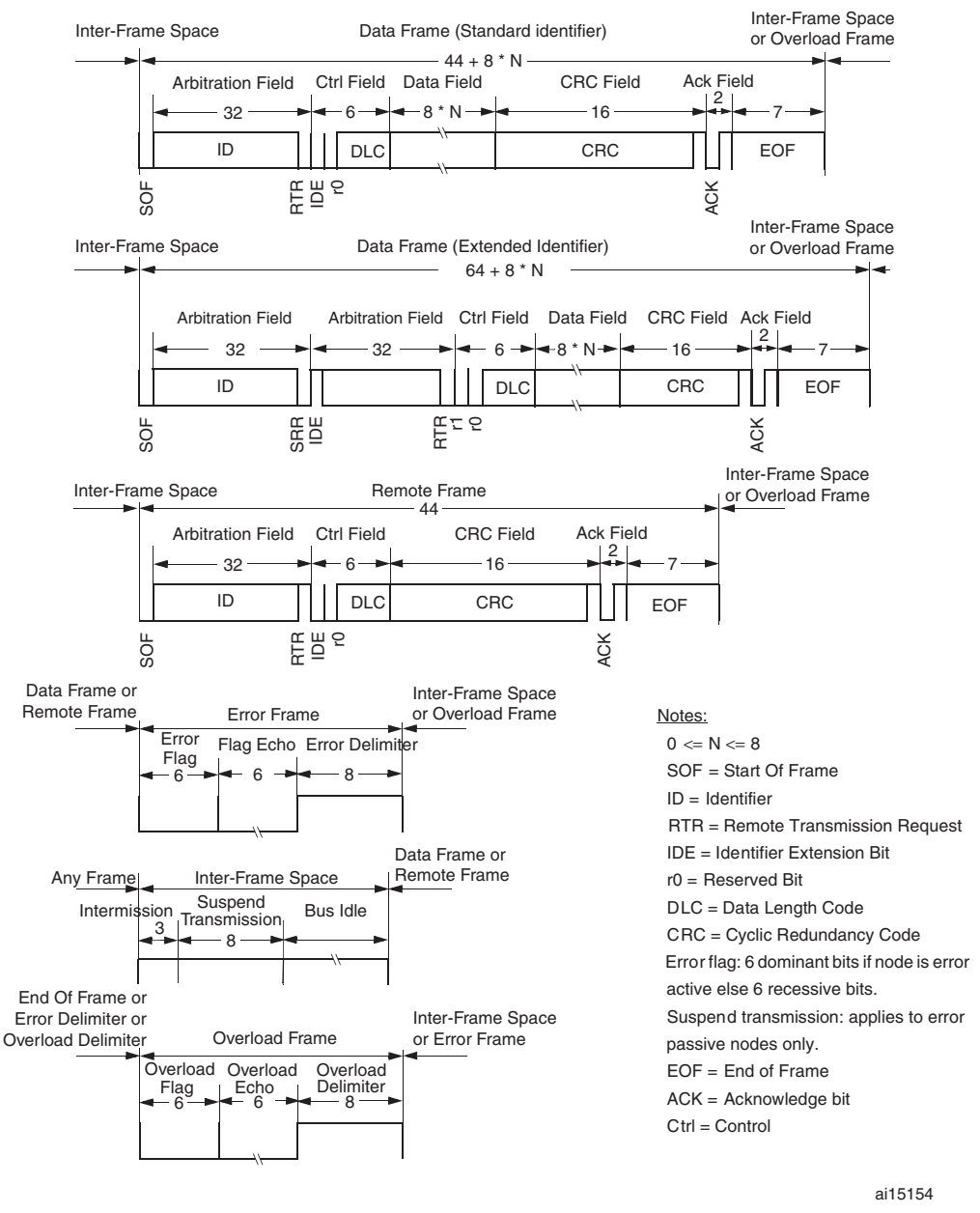
t_q = (BRP[9:0] + 1) x t_{PCLK}

where t_q refers to the Time quantum

t_{PCLK} = time period of the APB clock,

BRP[9:0], TS1[3:0] and TS2[2:0] are defined in the CAN_BTR Register.

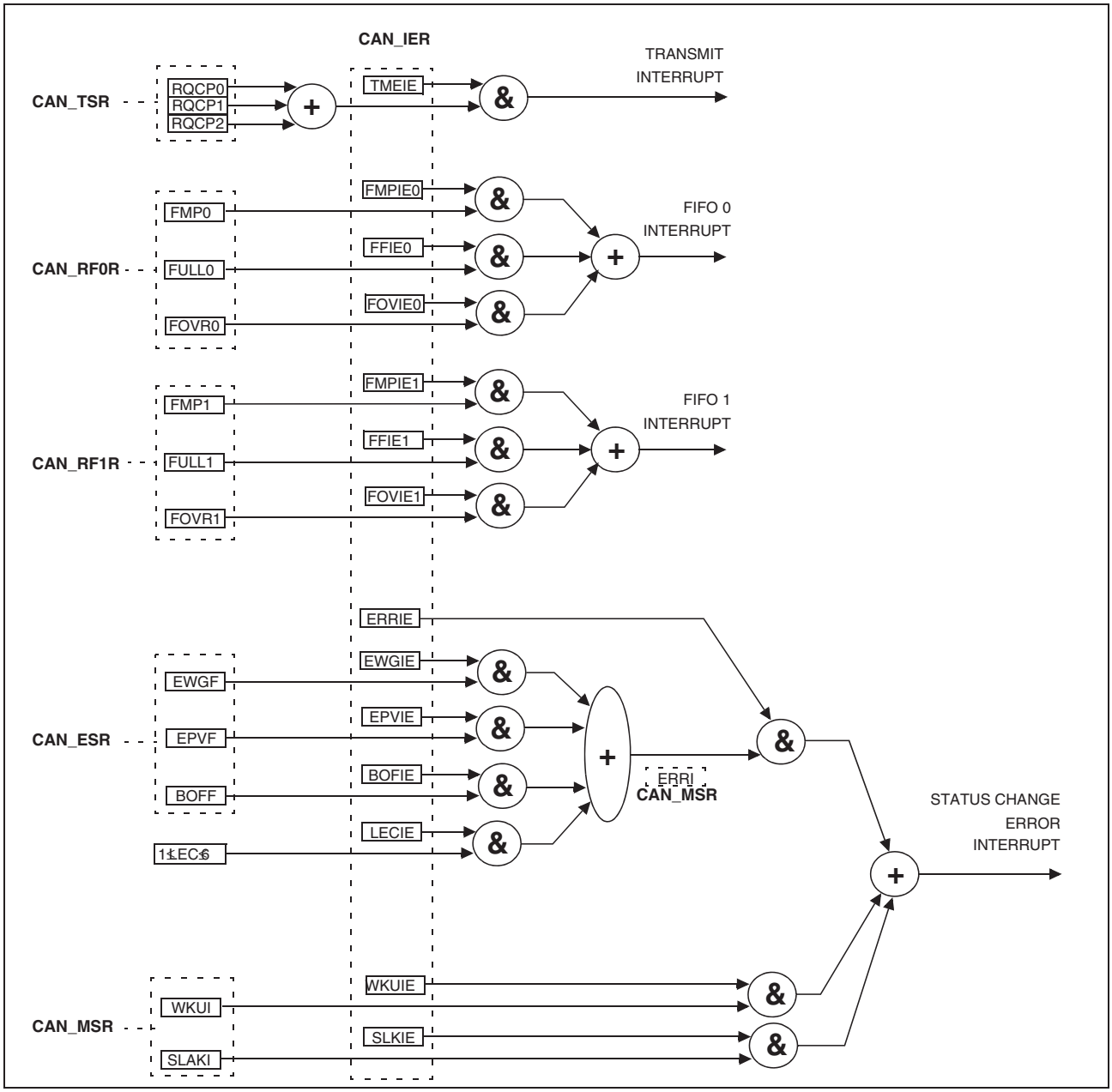
Рис. 347. Фреймы CAN



32.8. Прерывания bxCAN

Для bxCAN выделены 4 вектора. Разрешают их независимо через регистр CAN_IER.

Рис. 348. Флаги событий и прерывания



- **Прерывание передачи** выдаётся при:

– Пустом передающем ящике 0, стоит бит RQCP0 в регистре CAN_TSR.

– Пустом передающем ящике 1, стоит бит RQCP1 в регистре CAN_TSR.

– Пустом передающем ящике 2, стоит бит RQCP2 в регистре CAN_TSR.

- **Прерывание FIFO 0** выдаётся при:

– Приёме нового сообщения, бит FMP0 в CAN_RF0R не равен ’00’.

– Заполнении FIFO0, стоит бит FULL0 в регистре CAN_RF0R.

– Переполнении FIFO0, стоит бит FOVR0 в регистре CAN_RF0R.

- **Прерывание FIFO 1** выдаётся при:

– Приёме нового сообщения, бит FMP1 в CAN_RF1R не равен ’00’.

– Заполнении FIFO1, стоит бит FULL1 в регистре CAN_RF1R.

– Переполнении FIFO1, стоит бит FOVR1 в регистре CAN_RF1R.

- **Прерывание ошибки и смены состояния** выдаётся при:

– Ошибке, см. регистр CAN_ESR.

– Пробуждении, на CANRx обнаружен SOF.

– Вход в Сон.

32.9. Регистры bxCAN

Регистры доступны словами (32 бита).

32.9.1. Защита доступа к регистрам

Неправильная установка регистров конфигурации может обрушить всю сеть CAN. Так что регистр CAN_BTR можно изменить только в режиме инициализации.

Модифицировать можно только пустой передающий ящик.

Значения фильтров можно изменять деактивируя их или установив бит FINIT. Конфигурацию фильтров (масштаб, режим и назначение FIFO) в регистрах CAN_FMxR, CAN_FSxR и CAN_FFAR можно менять только при FINIT=1 в регистре CAN_FMR.

32.9.2. Регистры управления и состояния CAN

Главный регистр управления CAN (CAN_MCR)

Смещение адреса: 0x00

По сбросу: 0x0001 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															DBF
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESET	Reserved							TTCM	ABOM	AWUM	NART	RFLM	TXFP	SLEEP	INRQ
rs								rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:17 Резерв, не трогать.

— Бит 16 **DBF:** Останов при отладке

0: CAN работает

1:Приём/передача CAN стоят. Приёмные FIFO доступны/управляемы.

— Бит 15 **RESET:** Главный программный сброс bxCAN

0: Работайте.

1: Принудительный сброс bxCAN, а после сброса в Сон (биты FMP регистра CAN_MCR получают значения сброса). Бит снимается автоматически.

— Биты 14:8 Резерв, не трогать.

— Бит 7 **TTCM**: Режим временного разделения каналов

0: Выключен.

1: Включён

— Бит 6 **ABOM:** Автоматическое восстановление отключения шины

0: Восстановление по запросу программы, если отслежено 128 появлений 11 рецессивных битов и сначала программа поставила и сняла бит INRQ регистра CAN_MCR.

1: Автоматически, если отслежено 128 появлений 11 рецессивных битов.

— Бит 5 **AWUM**: Автоматическая побудка

0: Просыпаемся по запросу программы очисткой бита SLEEP регистра CAN_MCR.

1: Просыпаемся аппаратно при обнаружении сообщения CAN. Бит SLEEP регистра CAN_MCR и бит SLAK регистра CAN_MSR снимаются аппаратно.

— Бит 4 **NART**: Нет автоматическому повтору сбойной передачи!

0: Аппаратура CAN автоматически повторяет сбойную передачу.

1: Сообщение передаётся единожды, не глядя на результат (успех, ошибка или потеря арбитража).

— Бит 3 **RFLM**: Замок приёмного FIFO

0: Приёмный FIFO не замкнут. Приём в заполненный FIFO замещает предыдущее сообщение.

1: Приёмный FIFO замкнут. Сообщение, поступающее в заполненный FIFO теряется.

— Бит 2 **TXFP**: Приоритет передающего FIFO

0: Приоритет управляется идентификатором сообщения

1: Приоритет управляется порядком запросов (хронологически)

— Бит 1 **SLEEP**: Запрос режима Сна

Ставится программно чтобы CAN уснул. Уснёт сразу после завершения текущей операции CAN (передача или приём фрейма CAN).

Снимается программно для выхода из Сна.

Снимается аппаратно при обнаружении бита SOF на сигнале CANRx при стоящем бите AWUM.

Ставится после сброса - CAN стартует спящим.

— Бит 0 **INRQ**: Запрос инициализации

Программная очистка переключает аппаратуру в нормальный режим. После 11 последовательных рецессивных битов на сигнале Rx аппаратура CAN засинхронизирована и готова к обмену. Об этом сообщается аппаратным снятием бита INAK регистра CAN_MSR.

Программная установка переключает аппаратуру в режим инициализации. Переключится сразу после завершения текущей операции CAN (передача или приём фрейма CAN). Об этом сообщается аппаратной установкой бита INAK регистра CAN_MSR.

Главный регистр состояния CAN (CAN_MSR)

Смещение адреса: 0x04

По сбросу: 0x0000 0C02

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved.				RX	SAMP	RXM	TXM	Reserved			SLAKI	WKUI	ERRI	SLAK	INAK
				r	r	r	r				rc_w1	rc_w1	rc_w1	r	r

— Биты 31:12 Резерв, не трогать.

— Бит 11 **RX**: Сигнал CAN Rx. Текущее состояние ножки **CAN_RX**.

— Бит 10 **SAMP**: Значение RX в последней точке считывания (принятый бит).

— Бит 9 **RXM**: Сейчас CAN принимает.

— Бит 8 **TXM**: Сейчас CAN передаёт.

— Биты 7:5 Резерв, не трогать.

— Бит 4 **SLAKI**: Прерывание подтверждения Сна

При SLKIE=1, этот бит ставится аппаратно, извещая, что bxCAN спит, и генерирует прерывание смены состояния. Снимается программно или аппаратно при очистке SLAK.

NB: Если SLKIE=0, то опрос SLAKI невозможен. В этом случае можно опросить бит SLAK.

— Бит 3 **WKUI**: Прерывание пробуждения

Этот бит ставится аппаратно, извещая, что обнаружен SOF при спящем bxCAN. Прерывание смены режима вызывается при стоящем бите WKUIE в регистре CAN_IER.

Снимается программно.

— Бит 2 **ERRI**: Прерывание ошибки

Этот бит ставится аппаратно при установке бита ошибки в регистре CAN_ESR и разрешении прерывания соответствующим битом в CAN_IER. Вызывает прерывание смены состояния при стоящем бите ERRIE в регистре CAN_IER.

Снимается программно.

— Бит 1 **SLAK**: Подтверждение Сна

Этот бит ставится аппаратно, извещая, что bxCAN спит. Это подтверждение программного запроса (ставит бит SLEEP в регистре CAN_MCR).

Снимается аппаратно при пробуждении bxCAN (для синхронизации с шиной CAN). Для синхронизации отслеживает появление 11 последовательных рецессивных битов на CAN RX.

NB: Выход из Сна запускается очисткой бита SLEEP регистра CAN_MCR.

— Бит 0 **INAK**: Подтверждение инициализации

Этот бит ставится аппаратно в ответ на программный запрос, извещая, что bxCAN инициализируется.

Снимается аппаратно при выходе bxCAN из инициализации (для синхронизации с шиной CAN). Для синхронизации отслеживает появление 11 последовательных рецессивных битов на CAN RX.

Регистр состояния передачи CAN (CAN_TSR)

Смещение адреса: 0x08

По сбросу: 0x1C00 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOW2	LOW1	LOW0	TME2	TME1	TME0	CODE[1:0]		ABRQ2	Reserved			TERR2	ALST2	TXOK2	RQCP2
r	r	r	r	r	r	r	r	rs				rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRQ1	Reserved Res.			TERR1	ALST1	TXOK1	RQCP1	ABRQ0	Reserved			TERR0	ALST0	TXOK0	RQCP0
rs				rc_w1	rc_w1	rc_w1	rc_w1	rs				rc_w1	rc_w1	rc_w1	rc_w1

— Бит 31 **LOW2**: Флаг самого низкого приоритета ящика 2
Ставится аппаратно, из удерживаемых к передаче ящиков ящик 2 имеет самый низкий приоритет.
— Бит 30 **LOW1**: Флаг самого низкого приоритета ящика 1
Ставится аппаратно, из удерживаемых к передаче ящиков ящик 1 имеет самый низкий приоритет.
— Бит 29 **LOW0**: Флаг самого низкого приоритета ящика 0
Ставится аппаратно, из удерживаемых к передаче ящиков ящик 0 имеет самый низкий приоритет.
— Бит 28 **TME2**: Передающий ящик 2 пуст
Ставится аппаратно.
— Бит 27 **TME2**: Передающий ящик 1 пуст
Ставится аппаратно.
— Бит 26 **TME2**: Передающий ящик 0 пуст
Ставится аппаратно.
— Биты 25:24 **CODE[1:0]**: Код ящиков
Если хоть один ящик пуст, то равен номеру следующего освобождаемого передающего ящика.
Если заняты все ящики, то это номер ящика с самым низким приоритетом.
— Бит 23 **ABRQ2**: Запрос аборта ящика 2
Ставится программно, снимается аппаратно при опустении ящика 2.
Установка бита не влияет на не удерживаемый к передаче ящик.
— Биты 22:20 Резерв, не трогать.
— Бит 19 **TERR2**: Ошибка передачи ящика 2
Ставится при сбойной предыдущей передаче.
— Бит 18 **ALST2**: Потеря арбитража ящика 2
Ставится при потере арбитража предыдущей передачей.
— Бит 17 **TXOK2**: Нормальная передача ящика 2
Аппаратно обновляется при каждой попытке передачи.
0: Предыдущая передача сбойная
1: Предыдущая передача успешная
— Бит 16 **RQCP2**: Запрос ящика 2 закончен
Ставится аппаратно при завершении последнего запроса (передача или аборт).
Чистится записью “1” или аппаратно при запросе передачи (установка TXRQ2 в CAN_TMID2R).
Снятие этого бита чистит все биты состояния (TXOK2, ALST2 и TERR2) ящика 2.
— Бит 15 **ABRQ1**: Запрос аборта ящика 1
Ставится программно, снимается аппаратно при опустении ящика 1.
Установка бита не влияет на не удерживаемый к передаче ящик.
— Биты 14:12 Резерв, не трогать.
— Бит 11 **TERR1**: Ошибка передачи ящика 1
Ставится при сбойной предыдущей передаче.
— Бит 10 **ALST1**: Потеря арбитража ящика 1
Ставится при потере арбитража предыдущей передачей.
— Бит 9 **TXOK1**: Нормальная передача ящика 1
Аппаратно обновляется при каждой попытке передачи.
0: Предыдущая передача сбойная
1: Предыдущая передача успешная
— Бит 8 **RQCP1**: Запрос ящика 1 закончен
Ставится аппаратно при завершении последнего запроса (передача или аборт).
Чистится записью “1” или аппаратно при запросе передачи (установка TXRQ1 в CAN_TMID2R).
Снятие этого бита чистит все биты состояния (TXOK1, ALST1 и TERR1) ящика 1.
— Бит 7 **ABRQ0**: Запрос аборта ящика 0
Ставится программно, снимается аппаратно при опустении ящика 0.
Установка бита не влияет на не удерживаемый к передаче ящик.
— Биты 6:4 Резерв, не трогать.
— Бит 3 **TERR0**: Ошибка передачи ящика 0
Ставится при сбойной предыдущей передаче.
— Бит 2 **ALST0**: Потеря арбитража ящика 0
Ставится при потере арбитража предыдущей передачей.

— Бит 1 **TXOK0**: Нормальная передача ящика 0
Аппаратно обновляется при каждой попытке передачи.
0: Предыдущая передача сбойная
1: Предыдущая передача успешная

— Бит 0 **RQCP0**: Запрос ящика 0 закончен
Ставится аппаратно при завершении последнего запроса (передача или аборт).
Чистится записью “1” или аппаратно при запросе передачи (установка TXRQ0 в CAN_TMID2R).
Снятие этого бита чистит все биты состояния (TXOK0, ALST0 и TERR0) ящика 0.

Регистр приёмного FIFO 0 CAN (CAN_RF0R)

Смещение адреса: 0x0C

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										RFOM0	FOVR0	FULL0	Res.	FMP0[1:0]	
										rs	rc_w1	rc_w1		r	r

— Биты 31:6 Резерв, не трогать.

— Бит 5 **RFOM0**: Освободи выходной ящик FIFO 0
Ставится программно. Освобождает выходной ящик непустого FIFO для приёма следующего сообщения, на пустой FIFO не влияет. Снимается аппаратно.

— Бит 4 **FOVR0**: Переполнение FIFO 0
Ставится аппаратно при приёме сообщения для заполненного FIFO. Снимается программно.

— Бит 3 **FULL0**: FIFO0 полон.
Ставится аппаратно при трёх сообщениях в FIFO. Снимается программно.

— Бит 2 Резерв, не трогать

— Биты 1:0 **FMP0[1:0]**: Счёт сообщений в FIFO 0
Увеличивается при приёме сообщения в FIFO. Уменьшается при освобождении выходного ящика установкой бита RFOM0.

Регистр приёмного FIFO 1 CAN (CAN_RF1R)

Смещение адреса: 0x10

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										RFOM1	FOVR1	FULL1	Res.	FMP1[1:0]	
										rs	rc_w1	rc_w1		r	r

— Биты 31:6 Резерв, не трогать.

— Бит 5 **RFOM1**: Освободи выходной ящик FIFO 1
Ставится программно. Освобождает выходной ящик непустого FIFO для приёма следующего сообщения, на пустой FIFO не влияет. Снимается аппаратно.

— Бит 4 **FOVR1**: Переполнение FIFO 1
Ставится аппаратно при приёме сообщения для заполненного FIFO. Снимается программно.

— Бит 3 **FULL1**: FIFO 1 полон.
Ставится аппаратно при трёх сообщениях в FIFO. Снимается программно.

— Бит 2 Резерв, не трогать

— Биты 1:0 **FMP1[1:0]**: Счёт сообщений в FIFO 1
Увеличивается при приёме сообщения в FIFO. Уменьшается при освобождении выходного ящика установкой бита RFOM1.

Регистр разрешения прерываний CAN (CAN_IER)

Смещение адреса: 0x14

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved														SLKIE	WKUIE
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERRIE	Reserved			LEC IE	BOF IE	EPV IE	EWG IE	Res.	FOV IE1	FF IE1	FMP IE1	FOV IE0	FF IE0	FMP IE0	TME IE
rw				rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

— Биты 31:18 Резерв, не трогать.

— Бит 17 **SLKIE**: Разрешение прерывания Сна при стоящем бите SLAKI
0: Нельзя.
1: Можно.

— Бит 16 **WKUIE**: Разрешение прерывания побудки при стоящем бите WKUI
0: Нельзя.
1: Можно.

— Бит 15 **ERRIE**: Разрешение прерывания ошибки при наличии события в CAN_ESR
0: Нельзя.
1: Можно.

— Биты 14:12 Резерв, не трогать.

— Бит 11 **LECIE**: Разрешение прерывания по коду последней ошибки
0: Бит ERRI не ставится при аппаратной записи кода ошибки в LEC[2:0].
1: Бит ERRI ставится при аппаратной записи кода ошибки в LEC[2:0].

— Бит 10 **BOFIE**: Разрешение прерывания по отключению шины
0: Бит ERRI не ставится при стоящем BOFF.
1: Бит ERRI ставится при стоящем BOFF.

— Бит 9 **EPVIE**: Разрешение прерывания пассивной ошибки
0: Бит ERRI не ставится при стоящем EPVF.
1: Бит ERRI ставится при стоящем EPVF.

— Бит 8 **EWGIE**: Разрешение прерывания ошибки уровня предупреждения
0: Бит ERRI не ставится при стоящем EWGF.
1: Бит ERRI ставится при стоящем EWGF.

— Бит 7 Резерв, не трогать.

— Бит 6 **FOVIE1**: Разрешение прерывания переполнения FIFO при стоящем FOVR
0: Нельзя.
1: Можно.

— Бит 5 **FFIE1**: Разрешение прерывания заполнения FIFO при стоящем FULL
0: Нельзя.
1: Можно.

— Бит 4 **FMPIE1**: Разрешение прерывания непустого FIFO (FMP[1:0] ненулевые)
0: Нельзя.
1: Можно.

— Бит 3 **FOVIE0**: Разрешение прерывания переполнения FIFO при стоящем FOVR
0: Нельзя.
1: Можно.

— Бит 2 **FFIE0**: Разрешение прерывания заполнения FIFO при стоящем FULL
0: Нельзя.
1: Можно.

— Бит 1 **FMPIE0**: Разрешение прерывания непустого FIFO (FMP[1:0] ненулевые)
0: Нельзя.
1: Можно.

— Бит 0 **TMEIE**: Разрешение прерывания пустого выходного ящика (RQCPx=1)
0: Нельзя.
1: Можно.

Регистр состояния ошибки CAN (CAN_ESR)

Смещение адреса: 0x18

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REC[7:0]									TEC[7:0]						
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									LEC[2:0]			Res.	BOFF	EPVF	EWGF
									rw	rw	rw		r	r	r

— Биты 31:24 **REC[7:0]**: Счётчик ошибок приёма
Реализованная часть механизма ограничения ошибок протокола CAN. При возникновении ошибки во время приёма этот счётчик увеличивается на вес ошибки от 1 до 8, определённый в стандарте CAN. После успешного приёма счётчик уменьшается на 1 или сбрасывается до 120, если был больше 128. При достижении счётчиком значения 127 наступает режим пассивной ошибки.

— Биты 23:16 **TEC[7:0]**: Младший байт 9-бит счётчика ошибок передачи
Реализованная часть механизма ограничения ошибок протокола CAN.

— Биты 15:7 Резерв, не трогать.

— Биты 6:4 **LEC[2:0]**: Код последней ошибки на шине CAN
Пишется аппаратно. Нормальная передача чистит поле.
Значение LEC[2:0] = 0b111 пишется программно.
000: Ошибки нет
001: Ошибка заполнения
010: Ошибка формы
011: Ошибка подтверждения
100: Ошибка рецессивного бита
101: Ошибка доминантного бита
110: Ошибка CRC
111: Пишется программно

— Бит 3 Резерв, не трогать.

— Биты 2 **BOFF**: Флаг отключения шины
Ставится аппаратно при переполнении TEC, больше 255.

— Биты 1 **EPVF**: Флаг пассивной ошибки
Ставится аппаратно при TEC или REC >127.

— Биты 0 **EWGF**: Флаг ошибки уровня предупреждения
Ставится аппаратно при TEC или REC≥96.

Регистр времянки бита CAN (CAN_BTR)

Смещение адреса: 0x1C

По сбросу: 0x0123 0000

Программно доступен только в режиме Инициализации bxCAN.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SILM	LBKM	Reserved				SJW[1:0]		Res.	TS2[2:0]			TS1[3:0]			
rw	rw					rw	rw		rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						BRP[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Бит 31 **SILM**: Режим тишины
0: Выключен
1: Включён

— Бит 30 **LBKM**: Режим перемычки
0: Выключен
1: Включён

— Биты 29:26 Резерв, не трогать

— Биты 25:24 **SJW[1:0]**: Ширина шага ресинхронизации
Число квантов времени аппаратуры CAN для удлинения или сокращения бита при ресинхронизации.
$t_{RJW} = t_q \times (SJW[1:0] + 1)$

— Бит 23 Резерв, не трогать.
— Биты 22:20 **TS2[2:0]**: Длина сегмента времени 2 в квантах
$t_{BS2} = t_q$ x (TS2[2:0] + 1)
— Биты 19:16 **TS1[3:0]**: Длина сегмента времени 1 в квантах
$t_{BS1} = t_q$ x (TS1[3:0] + 1)
— Биты 19:16 Резерв, не трогать
— Биты 9:0 **BRP[9:0]**: Длина кванта времени
t_q = (BRP[9:0]+1) x t_{PCLK}

32.9.3. Регистры почтовых ящиков CAN

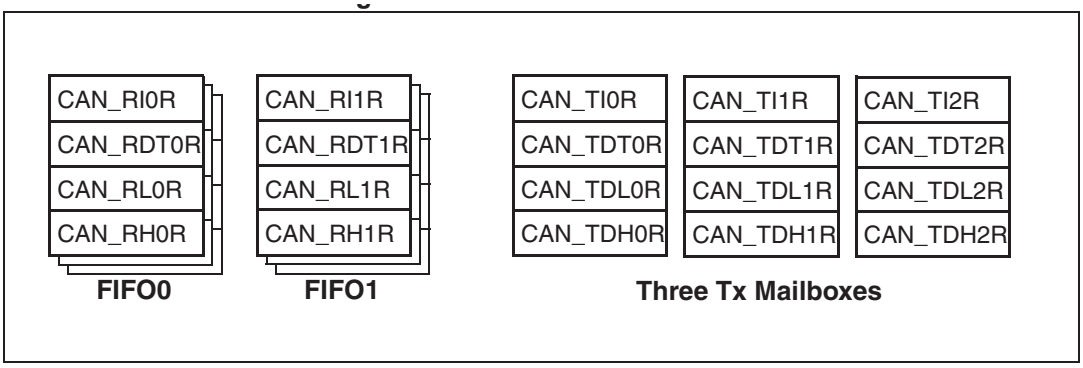
Передающие и приёмные ящики имеют одинаковые регистры, за исключением:

- Поля FMI в регистре CAN_RDTxR.
- Приёмный ящик защищён от записи всегда.
- Запись возможна только в пустой передающий ящик (стоит бит TME в регистре CAN_TSR).

Есть 3 TX ящика и 2 RX ящика. Ящик RX имеет трёхуровневый FIFO, доступно только самое старое сообщение.

Ящик состоит из 4.

Ящики RX и TX



Регистр идентификатора ящика TX (CAN_TIxR) (x=0..2)

Смещение адреса: 0x180, 0x190, 0x1A0

По сбросу: 0xXXXX XXXX (кроме бита 0, TXRQ = 0)

Регистры TX защищены от записи при сброшенном бите TMEx. Есть управление запросом передачи (бит 0).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STID[10:0]/EXID[28:18]											EXID[17:13]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXID[12:0]													IDE	RTR	TXRQ
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:21 **STID[10:0]/EXID[28:18]**: Стандартный или расширенный идентификатор
Стандартный идентификатор или старшие биты расширенного (в зависимости от бита IDE)
— Биты 20:3 **EXID[17:0]**: Младшие биты расширенного идентификатора
— Бит 2 **IDE**: Расширение идентификатора, 0 - стандартный, 1 - расширенный.
— Бит 1 **RTR**: Запрос удалённой передачи, 0 - фрейм данных, 1 - удалённый фрейм
— Бит 0 **TXRQ**: Запрос передачи ящика.
Ставится программно, снимается аппаратно при опустошении ящика.

Регистр длины данных и меток времени (CAN_TDTxR) (x=0..2)

Смещение адреса: 0x184, 0x194, 0x1A4

По сбросу: 0xXXXX XXXX

Регистр защищён от записи при непустом ящике.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIME[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							TGT	Reserved				DLC[3:0]			
							rw					rw	rw	rw	rw

— Биты 31:16 **TIME[15:0]**: Метка времени сообщения
16-бит значение таймера на момент передачи SOF.
— Биты 15:9 Резерв, не трогать
— Бит 8 **TGT**: Глобальное время передачи
Активен только в режиме временного разделения каналов, стоит бит TTCM регистра CAN_MCR.
0: Метка времени TIME[15:0] не посылается.
1: Метка времени TIME[15:0] посылается в двух последних байтах 8-байт сообщения: TIME[7:0] в байте 7 и TIME[15:8] в байте 6 данных, заменяя данные из регистра CAN_TDHxR[31:16] (DATA6[7:0] и DATA7[7:0]). DLC должно содержать число 8 для пересылки этих байтов по шине CAN.
— Биты 7:4 Резерв, не трогать
— Биты 3:0 **DLC[3:0]**: Код длины данных
Содержит число байтов данных в фрейме данных или запросе удалённого фрейма. Сообщение может содержать от 0 до 8 байтов данных, в зависимости от поля DLC.

Младший регистр данных ящика (CAN_TDLxR) (x=0..2)

Смещение адреса: 0x188, 0x198, 0x1A8
По сбросу: 0xXXXX XXXX
Регистр защищён от записи при непустом ящике.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA3[7:0]								DATA2[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA1[7:0]								DATA0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:24 **DATA3[7:0]**: Байт данных 3.
— Биты 23:16 **DATA2[7:0]**: Байт данных 2.
— Биты 15:8 **DATA1[7:0]**: Байт данных 1.
— Биты 7:0 **DATA0[7:0]**: Байт данных 0.

Старший регистр данных ящика (CAN_TDLxR) (x=0..2)

Смещение адреса: 0x18C, 0x19C, 0x1AC
По сбросу: 0xXXXX XXXX
Регистр защищён от записи при непустом ящике. Байты 6 и 7 могут заменяться меткой времени.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA7[7:0]								DATA6[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA5[7:0]								DATA4[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:24 **DATA7[7:0]**: Байт данных 7.
— Биты 23:16 **DATA6[7:0]**: Байт данных 6.
— Биты 15:8 **DATA5[7:0]**: Байт данных 5.
— Биты 7:0 **DATA4[7:0]**: Байт данных 4.

Регистр идентификатора приёмного ящика (CAN_RIxR) (x=0..1)

Смещение адреса: 0x1B0, 0x1C0
По сбросу: 0xXXXX XXXX
Регистры RX защищены от записи.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STID[10:0]/EXID[28:18]											EXID[17:13]				
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXID[12:0]													IDE	RTR	Res.
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	

— Биты 31:21 **STID[10:0]/EXID[28:18]**: Стандартный или расширенный идентификатор
Стандартный идентификатор или старшие биты расширенного (в зависимости от бита IDE)
— Биты 20:3 **EXID[17:0]**: Младшие биты расширенного идентификатора
— Бит 2 **IDE**: Расширение идентификатора, 0 - стандартный, 1 - расширенный.
— Бит 1 **RTR**: Запрос удалённой передачи, 0 - фрейм данных, 1 - удалённый фрейм
— Бит 0 Резерв, не трогать

Регистр длины данных и меток времени (CAN_RDTxR) (x=0..2)

Смещение адреса: 0x1B4, 0x1C4
По сбросу: 0xXXXX XXXX
Регистры RX защищены от записи.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIME[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FMI[7:0]								Reserved				DLC[3:0]			
r	r	r	r	r	r	r	r					r	r	r	r

— Биты 31:16 **TIME[15:0]**: Метка времени сообщения
16-бит значение таймера на момент появления SOF.
— Биты 15:8 **FMI[7:0]**: Индекс фильтра, пропустившего сообщение.
— Биты 7:4 Резерв, не трогать
— Биты 3:0 **DLC[3:0]**: Код длины данных
Содержит число байтов данных в фрейме данных или 0 при запросе удалённого фрейма.

Младший регистр данных ящика (CAN_RDLxR) (x=0..1)

Смещение адреса: 0x1B8, 0x1C8
По сбросу: 0xXXXX XXXX
Регистры RX защищены от записи.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA3[7:0]								DATA2[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA1[7:0]								DATA0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:24 **DATA3[7:0]**: Байт данных 3.
— Биты 23:16 **DATA2[7:0]**: Байт данных 2.
— Биты 15:8 **DATA1[7:0]**: Байт данных 1.
— Биты 7:0 **DATA0[7:0]**: Байт данных 0.

Старший регистр данных ящика (CAN_RDLxR) (x=0..1)

Смещение адреса: 0x1BC, 0x1CC
По сбросу: 0xXXXX XXXX
Регистры RX защищены от записи.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA7[7:0]								DATA6[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA5[7:0]								DATA4[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:24 **DATA7[7:0]**: Байт данных 7.
— Биты 23:16 **DATA6[7:0]**: Байт данных 6.
— Биты 15:8 **DATA5[7:0]**: Байт данных 5.
— Биты 7:0 **DATA4[7:0]**: Байт данных 4.

32.9.4. Регистры фильтров CAN

Главный регистр фильтров (CAN_FMR)

Смещение адреса: 0x200

По сбросу: 0x2A1C 0E01

Все биты ставятся и снимаются программно.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		CAN2SB[5:0]						Reserved							FINIT
		rw	rw	rw	rw	rw	rw								rw

— Биты 31:14 Резерв, не трогать
— Биты 13:8 **CAN2SB[5:0]**: Стартовый банк CAN2
Стартовый банк ведомого CAN2 в диапазоне от 0 до 27.
NB: При CAN2SB[5:0] = 28d можно использовать фильтры CAN1.
При CAN2SB[5:0] = 0, фильтров CAN1 нет.
— Биты 7:1 Резерв, не трогать
— Бит 0 **FINIT**: Режим инициализации фильтров
0: Нормальная работа.
1: Инициализация.

Регистр режима фильтров (CAN_FM1R)

Смещение адреса: 0x204

По сбросу: 0x0000 0000

Писать можно только в режиме инициализации (FINIT=1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				FBM27	FBM26	FBM25	FBM24	FBM23	FBM22	FBM21	FBM20	FBM19	FBM18	FBM17	FBM16
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FBM15	FBM14	FBM13	FBM12	FBM11	FBM10	FBM9	FBM8	FBM7	FBM6	FBM5	FBM4	FBM3	FBM2	FBM1	FBM0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:28 Резерв, не трогать
— Биты 27:0 **FBM*x***: Режим фильтра
0: Два 32-бит регистра банка в режиме Маски.
1: Два 32-бит регистра банка в режиме Списка идентификаторов.
NB: Биты 27:14 доступны только в сетевых устройствах, иначе - резерв.

Регистр масштаба фильтров (CAN_FS1R)

Смещение адреса: 0x20C

По сбросу: 0x0000 0000

Писать можно только в режиме инициализации (FINIT=1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				FSC27	FSC26	FSC25	FSC24	FSC23	FSC22	FSC21	FSC20	FSC19	FSC18	FSC17	FSC16
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FSC15	FSC14	FSC13	FSC12	FSC11	FSC10	FSC9	FSC8	FSC7	FSC6	FSC5	FSC4	FSC3	FSC2	FSC1	FSC0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:28 Резерв, не трогать

— Биты 27:0 **FSC*x***: Конфигурация масштаба фильтров 13 - 0

0: Парный 16-бит.

1: Одинарный 32-бит.

NB: Биты 27:14 доступны только в сетевых устройствах, иначе - резерв.

Регистр назначения фильтров (CAN_FFA1R)

Смещение адреса: 0x214

По сбросу: 0x0000 0000

Писать можно только в режиме инициализации (FINIT=1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				FFA27	FFA26	FFA25	FFA24	FFA23	FFA22	FFA21	FFA20	FFA19	FFA18	FFA17	FFA16
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FFA15	FFA14	FFA13	FFA12	FFA11	FFA10	FFA9	FFA8	FFA7	FFA6	FFA5	FFA4	FFA3	FFA2	FFA1	FFA0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:28 Резерв, не трогать

— Биты 27:0 **FFA*x***: Назначение FIFO фильтру x.

0: FIFO 0.

1: FIFO 1.

NB: Биты 27:14 доступны только в сетевых устройствах, иначе - резерв.

Регистр активации фильтров (CAN_FFA1R)

Смещение адреса: 0x21C

По сбросу: 0x0000 0000

Писать можно только в режиме инициализации (FINIT=1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				FACT27	FACT26	FACT25	FACT24	FACT23	FACT22	FACT21	FACT20	FACT19	FACT18	FACT17	FACT16
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FACT15	FACT14	FACT13	FACT12	FACT11	FACT10	FACT9	FACT8	FACT7	FACT6	FACT5	FACT4	FACT3	FACT2	FACT1	FACT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:28 Резерв, не трогать

— Биты 27:0 **FACT*x***: Активация фильтра x.

0: Не активен.

1: Активен.

NB: Биты 27:14 доступны только в сетевых устройствах, иначе - резерв.

Регистр x банка фильтров i (CAN_FiRx) (i=0..27, x=1, 2)

Смещение адреса: 0x240..0x31C

По сбросу: 0xXXXX XXXX

В сетевых устройствах есть 28 банков фильтров, i=0 .. 27, в прочих устройствах есть 14 банков фильтров i = 0 ..13. Банк состоит из двух 32-бит регистров, CAN_FiR[2:1].

Изменять этот регистр можно только при чистом бите FACTx в регистре CAN_FAxR или стоит бит FINIT регистра CAN_FMR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FB31	FB30	FB29	FB28	FB27	FB26	FB25	FB24	FB23	FB22	FB21	FB20	FB19	FB18	FB17	FB16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FB15	FB14	FB13	FB12	FB11	FB10	FB9	FB8	FB7	FB6	FB5	FB4	FB3	FB2	FB1	FB0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:0 **FB[31:0]**: Биты фильтров

Идентификатор

Каждый бит определяет уровень соответствующего бита ожидаемого идентификатора.

0: Ожидается доминантный бит

1: Ожидается рецессивный бит

Маска

Каждый бит определяет необходимость сравнения соответствующего бита с ожидаемым идентификатором.

0: Бит не нужен

1: Сравниваем. Уровень поступающего бита должен совпадать с уровнем соответствующего бита в регистре идентификатора.

В зависимости от конфигурации масштаба и режима, функции регистров могут различаться. См. Секцию 24.7.4. и Таблицу 181.

32.9.5. Карта регистров CAN

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	CAN_MCR	Reserved															DBF	RESET	Reserved							TTCM	ABOM	AWUM	NART	RFLM	TXFP	SLEEP	INRQ
	Reset value																1	0								0	0	0	0	0	0	1	0
0x004	CAN_MSR	Reserved																				RX	SAMP	RXM	TXM	Res.			SLAKI	WKUI	ERRI	SLAK	INAK
	Reset value																					1	1	0	0				0	0	0	1	0
0x008	CAN_TSR	LOW[2:0]			TME[2:0]			CODE[1:0]		ABRQ2	Res.			TERR2	ALST2	TXOK2	RQCP2	ABRQ1	Res.			TERR1	ALST1	TXOK1	RQCP1	ABRQ0	Res..			TERR0	ALST0	TXOK0	RQCP0
	Reset value	0	0	0	1	1	1	0	0	0				0	0	0	0	0				0	0	0	0	0				0	0	0	0
0x00C	CAN_RF0R	Reserved																										RFOM0	FOVR0	FULL0	Reserved	FMP0[1:0]	
	Reset value																											0	0	0		0	0
0x010	CAN_RF1R	Reserved																										RFOM1	FOVR1	FULL1	Reserved	FMP1[1:0]	
	Reset value																											0	0	0		0	0
0x014	CAN_IER	Reserved													SLKIE	WKUIE	ERRIE	Res.				LECIE	BOFIE	EPVIE	EWGIE	Reserved	FOVIE1	FFIE1	FMPIE1	FOVIE0	FFIE0	FMPIE0	TMEIE
	Reset value														0	0	0					0	0	0	0		0	0	0	0	0	0	0
0x018	CAN_ESR	REC[7:0]								TEC[7:0]									Reserved								LEC[2:0]			Reserved	BOFF	EPVF	EWGF
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									0	0	0		0	0	0
0x01C	CAN_BTR	SILM	LBKM	Reserved				SJW[1:0]		Reserved	TS2[2:0]			TS1[3:0]					Reserved							BRP[9:0]							
	Reset value	0	0					0	0		0	1	0	0	0	1	1									0	0	0	0	0	0	0	0	0	0
0x020-0x17F	Reserved																																
0x180	CAN_TI0R	STID[10:0]/EXID[28:18]											EXID[17:0]																		IDE	RTR	TXRQ
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0
0x184	CAN_TDT0R	TIME[15:0]																	Reserved							TGT	Reserved			DLC[3:0]			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x									x				x	x	x	x
0x188	CAN_TDL0R	DATA3[7:0]								DATA2[7:0]									DATA1[7:0]								DATA0[7:0]						
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x18C	CAN_TDH0R	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x190	CAN_TI1R	STID[10:0]/EXID[28:18]											EXID[17:0]																		IDE	RTR	TXRQ
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0
0x194	CAN_TDT1R	TIME[15:0]																Reserved							TGT	Reserved				DLC[3:0]			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x								x					x	x	x	x
0x198	CAN_TDL1R	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x19C	CAN_TDH1R	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1A0	CAN_TI2R	STID[10:0]/EXID[28:18]											EXID[17:0]																		IDE	RTR	TXRQ
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0
0x1A4	CAN_TDT2R	TIME[15:0]																Reserved							TGT	Reserved				DLC[3:0]			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x								x					x	x	x	x
0x1A8	CAN_TDL2R	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1AC	CAN_TDH2R	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1B0	CAN_RI0R	STID[10:0]/EXID[28:18]											EXID[17:0]																		IDE	RTR	Reserved
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
0x1B4	CAN_RDT0R	TIME[15:0]																FMI[7:0]								Reserved				DLC[3:0]			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x					x	x	x	x
0x1B8	CAN_RDL0R	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1BC	CAN_RDH0R	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1C0	CAN_RI1R	STID[10:0]/EXID[28:18]											EXID[17:0]																		IDE	RTR	Reserved
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1C4	CAN_RDT1R	TIME[15:0]																FMI[7:0]								Reserved				DLC[3:0]			
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x					x	x	x	x
0x1C8	CAN_RDL1R	DATA3[7:0]								DATA2[7:0]								DATA1[7:0]								DATA0[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1CC	CAN_RDH1R	DATA7[7:0]								DATA6[7:0]								DATA5[7:0]								DATA4[7:0]							
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x1D0-0x1FF	Reserved																																
0x200	CAN_FMR	Reserved																		CAN2SB[5:0]						Reserved							FINIT
	Reset value																			0	0	1	1	1	0								1
0x204	CAN_FM1R	Reserved								FBM[27:0]																							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x208	Reserved																																
0x20C	CAN_FS1R	Reserved								FSC[27:0]																							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x210	Reserved																																
0x214	CAN_FFA1R	Reserved								FFA[27:0]																							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x218	Reserved																																
0x21C	CAN_FA1R	Reserved								FACT[27:0]																							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x220	Reserved																																
0x224-0x23F	Reserved																																
0x240	CAN_F0R1	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x244	CAN_F0R2	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x248	CAN_F1R1	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x24C	CAN_F1R2	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
. . . .	. . . .	. . . .																															
0x318	CAN_F27R1	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x31C	CAN_F27R2	FB[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

33. Ethernet (ETH): контроллер доступа к среде (MAC) с контроллером DMA

33.1. Введение в Ethernet

Поддерживаются два промышленных стандарта интерфейса с внешним физическим уровнем (PHY): медиа-независимый интерфейс по умолчанию (MII) спецификации IEEE 802.3 и упрощённый медиа-независимый интерфейс (RMII).

Ethernet совместим со следующими стандартами:

- IEEE 802.3-2002 для Ethernet MAC
- IEEE 1588-2002 для синхронизации тактов точных сетей
- AMBA 2.0 для портов AHB Master/Slave
- Спецификация RMII от консорциума RMII

33.2. Основные свойства Ethernet

33.2.1. Свойства ядра

- Поддерживает 10/100 Мбит/с передачу данных по внешнему интерфейсу PHY
- Совместимо с IEEE 802.3 MII интерфейсом связи с внешним Fast Ethernet PHY
- Поддерживает операции полу- и полного дуплекса
  - Поддерживает протокол CSMA/CD полу-дуплекса
  - Поддерживает управление потоком IEEE 802.3x полного дуплекса
  - Необязательная передача программе принятых фреймов паузы в полном дуплексе
  - Поддерживает сеть с обратным давлением в полу-дуплексе
  - Автоматическая передача кадра паузы нулевых квантов по снятию ввода управления потоком в полном дуплексе
- Вставка Преамбулы и Старта-фрейма (SFD) при Передаче и удаление при Приёме
- Автоматическое вычисление CRC и заполнителя для фрейма
- Возможное удаление Заполнителя/CRC на приёме
- Программируемая длина фрейма до 16 KB
- Программируемый промежуток между фреймами (40-96 бит шагами по 8)
- Поддержка режимов фильтрации адресов:
  - До четырёх 48-бит фильтров полного адреса (DA) с масками для каждого байта
  - До трёх 48-бит сравнений адреса SA с масками для каждого байта
  - 64-бит Хэш фильтр (опция) для широковещательных и полных (DA) адресов
  - Опция передачи всех широковещательных фреймов
  - Беспорядочный режим передачи всех фреймов без фильтрации для контроля сети
  - Передача всех поступающих пакетов с отчётом о состоянии
- Раздельное 32-бит состояние принимаемых и передаваемых пакетов
- Поддержка определения тэга IEEE 802.1Q VLAN в принимаемых фреймах
- Раздельные интерфейсы программы для передачи, приёма и управления
- Поддержка обязательной статистики сети с счётчиками RMON/MIB (RFC2819/RFC2665)
- Интерфейс MDIO для конфигурации и управления устройствами PHY
- Обнаружение фреймов побудки LAN и фреймов AMD Magic Packet™
- Приём дополнительной контрольной суммы принятых пакетов IPv4 и TCP внутри фреймов Ethernet
- Улучшенная проверка контрольной суммы заголовка IPv4 и контрольных сумм TCP, UDP или ICMP внутри дейтаграмм IPv4 и IPv6
- Поддержка меток времени фреймов Ethernet согласно IEEE 1588-2002. 64-бит метки времени в статусе каждого фрейма приёма и передачи
- Два набора FIFO: 2-KB Передающий FIFO с программируемым порогом и 2-KB и Приёмный FIFO с изменяемым порогом (по умолчанию 64 байта)

- Сохраняемые в приёмном FIFO после EOF векторы статуса приёма
- Опция фильтрации сбойных фреймов без участия программы
- Опция выдачи малоразмерных хороших фреймов
- Поддержка статистики выдачей импульсов для отброшенных и сбойных фреймов в FIFO приёма
- Поддержка механизма временного хранения (Store and Forward) передач ядру MAC
- Автоматическая выдача ядру MAC сигнала фрейма PAUSE или обратного давления на основе порога FIFO приёма
- Автоматическая повторная передача фреймов при коллизии
- Отбрасывание фреймов при запаздывании, чрезмерных коллизиях, излишней задержке и исчерпании
- Программное управление сливом TxFIFO
- Вычисление и вставка контрольных сумм заголовков IPv4 и TCP, UDP или ICMP внутри передаваемых фреймов в режиме временного хранения
- Поддержка внутренней перемычки (loopback) в MII для отладки

33.2.2. Свойства DMA

- Поддержка всех типов пакетов ведомого AHB
- Программный выбор типа пакета (фиксированный или неопределённый) ведущего AHB.
- Опция выбора пакетов с выравненными адресами из порта ведущего AHB
- Оптимизация пакетно-ориентированных передач DMA с разделителями фреймов
- Байтовое выравнивание адресов буферов данных
- Двухбуферная (кольцевая) или Цепная (связанным списком) связка дескрипторов
- Архитектура дескрипторов, позволяющая передачу больших блоков данных без участия CPU;
- Один дескриптор может передавать до 8 KB данных
- Исчерпывающий отчёт о нормальных и сбойных передачах
- Раздельный программируемый размер пакетов у машин DMA приёма и передачи
- Программируемые опции прерываний
- Прерывания по концу фреймов Приёма/Передачи
- Кольцевой или приоритетный арбитраж для машин Приёма и Передачи
- Режимы Старт/Стоп
- Текущий указатель буфера Tx/Rx как регистр состояния
- Текущий указатель Дескрипторов Tx/Rx как регистр состояния

33.2.3. Свойства PTP

- Метки времени принимаемых и передаваемых фреймов
- Методы грубой и точной коррекции
- Выдаёт прерывание если системное время превышает целевое
- Импульс секунд

33.3. Ножки Ethernet

Таблица 185. Карта альтернативных функций Ethernet

Port	AF11
	ETH
PA0-WKUP	ETH_MII_CRS
PA1	ETH_MII _RX_CLK / ETH_RMII _REF_CLK
PA2	ETH _MDIO
PA3	ETH _MII_COL
PA7	ETH_MII _RX_DV / ETH_RMII _CRS_DV
PB0	ETH _MII_RXD2
PB1	ETH _MII_RXD3
PB5	ETH _PPS_OUT

PB8	ETH _MII_TXD3
PB10	ETH_ MII_RX_ER
PB11	ETH _MII_TX_EN / ETH _RMII_TX_EN
PB12	ETH _MII_TXD0 / ETH _RMII_TXD0
PB13	ETH _MII_TXD1 / ETH _RMII_TXD1
PC1	ETH _MDC
PC2	ETH _MII_TXD2
PC3	ETH _MII_TX_CLK
PC4	ETH_MII_RXD0 / ETH_RMII_RXD0
PC5	ETH _MII_RXD1/ ETH _RMII_RXD1
PE2	ETH_MII_TXD3
PG8	ETH_PPS_OUT
PG11	ETH _MII_TX_EN / ETH _RMII_TX_EN
PG13	ETH _MII_TXD0 / ETH _RMII_TXD0
PG14	ETH _MII_TXD1 / ETH _RMII_TXD1
PH2	ETH _MII_CRS
PH3	ETH _MII_COL
PH6	ETH _MII_RXD2
PH7	ETH _MII_RXD3
PI10	ETH _MII_RX_ER

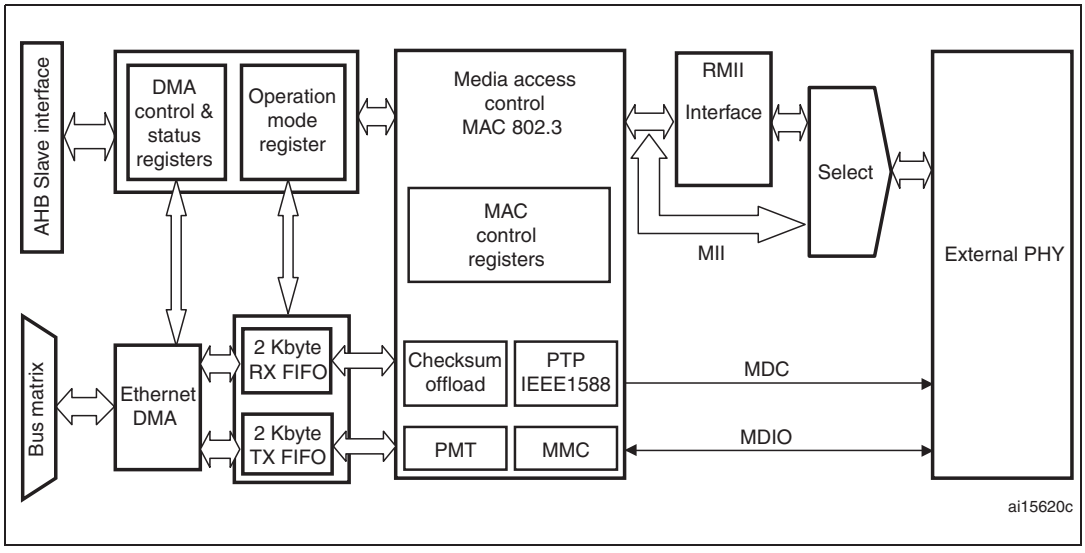
33.4. Функциональное описание Ethernet: SMI, MII и RMII

Блок состоит из MAC 802.3 (контроллер доступа среды) с контроллером DMA. Он поддерживает интерфейсы MII и RMII. Также есть SMI для связи с внешним PHY.

Контроллер DMA работает с ядром и памятью ведущим и ведомым по шине AHB. Ведущим при передаче данных и ведомым при доступе к регистрам (CSR).

NB: Частота AHB должна быть не меньше 25 MHz при работе с Ethernet.

Блок-схема ETH



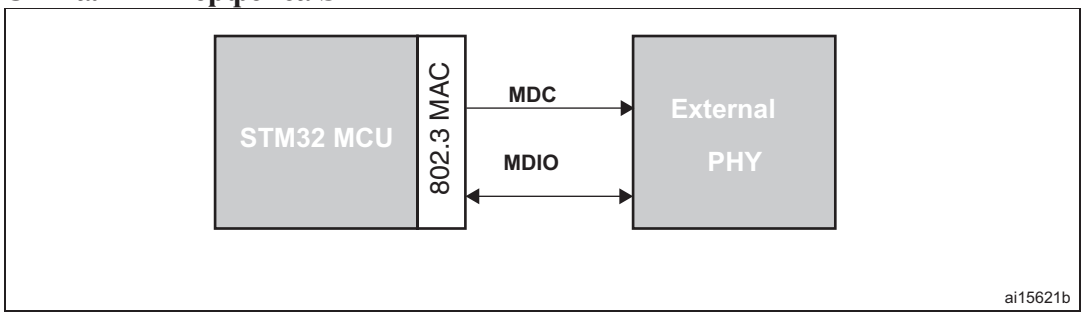
33.4.1. Интерфейс управления станцией: SMI

Обеспечивает доступ к регистрам PHY по 2 линиям тактов (MDC) и данных (MDIO). Поддерживаются до 32 PHY. Единовременно доступен только один регистр PHY.

Обе линии реализованы как альтернативные функции I/O:

- MDC: Такты передачи данных частотой до 2.5 MHz. Минимальная длительность состояний MDC равна 160 ns, минимальный период 400 ns. В состоянии простоя удерживается низким.
- MDIO: Линия данных, синхронна с MDC.

Сигналы интерфейса SMI



Формат фрейма SMI

Биты фрейма передаются начиная с левого.

Таблица 186. Формат фрейма управления

Поля фрейма								
	Преамбула (32 бита)	Старт	Операция	PADDR	RADDR	TA	Данные (16 бит)	Простой
Чтение	1... 1	01	10	ppppp	rrrrr	Z0	dddddddddddddddd	Z
Запись	1... 1	01	01	ppppp	rrrrr	10	dddddddddddddddd	Z

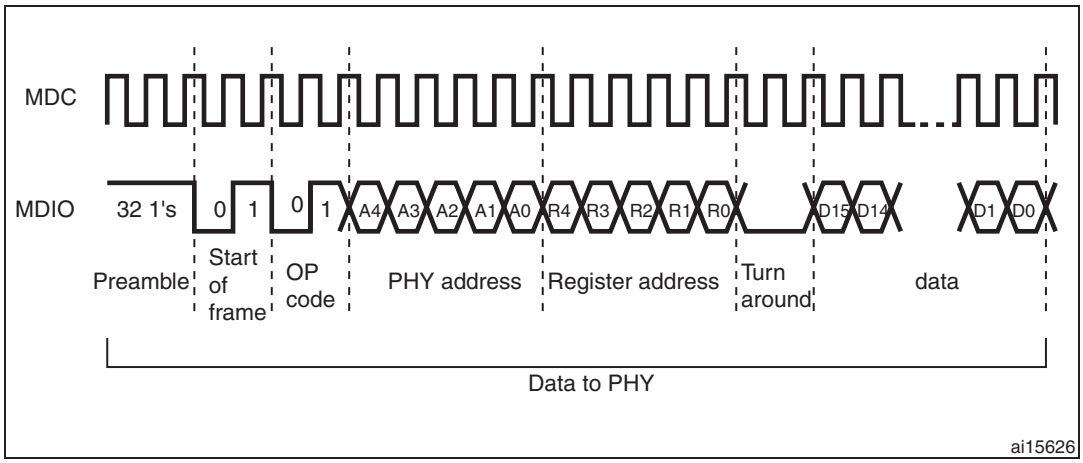
Фрейм состоит из восьми полей:

- **Преамбула**: все передачи начинаются с 32 единиц на MDIO за 32 такта MDC. Это синхронизация с PHY.
- **Старт**: Это биты <01>.
- **Операция**: Это тип операции (чтение или запись).
- **PADDR**: Это 5 бит адреса PHY. Старший бит передаётся первым.
- **RADDR**: Это 5 бит адреса регистра PHY. Старший бит передаётся первым.
- **TA**: Это 2-бит разделитель между полями RADDR и DATA. При чтении контроллер MAC ставит линию MDIO в Z-состояние на 2 бита TA. PHY ставит линию MDIO в Z-состояние на первый бит TA и в 0 во втором бите TA. При записи контроллер MAC выдаёт бита <10>. PHY держит линию в Z-состоянии 2 бита TA.
- **Данные**: Это 16 бит регистра `ETH_MIID`, начиная со старшего.
- **Простой**: Ключи линии MDIO стоят в Z-состоянии, резисторы подпорки PHY держат единицу.

Операция записи SMI

После программной установки битов MII Запись и Занят (`ETH_MACMIIAR`) SMI передаёт в регистры PHY адрес PHY, адрес регистра PHY и данные из регистра `ETH_MACMIIDR`. При работающей передаче регистры адреса и данных изменить нельзя (стоит бит Занят), он снимется после завершения операции.

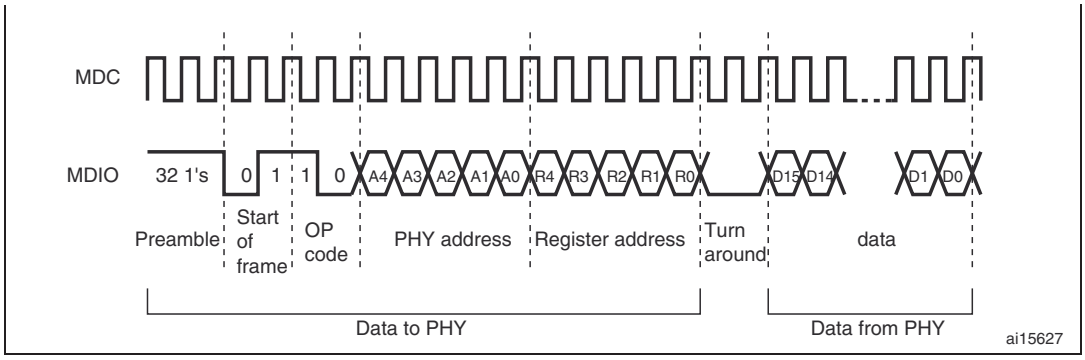
Времянка MDIO при записи



Операция чтения SMI

Запускается установкой бита Занят MII (в ETH_MACMIIAR) при снятом бите Запись. SMI передаёт в регистры PHY адрес PHY и адрес регистра PHY. Т При работающей передаче регистры адреса и данных изменить нельзя (стоит бит Занят). После завершения операции чтения SMI снимет бит Занят и заполнит регистр данных MII принятыми от PHY данными.

Времянка MDIO при чтении



Выбор тактов SMI

MAC запускает операцию Чтения/Записи. Такты SMI получаются делением тактов AHB в соответствии с полем в регистре адреса MII.

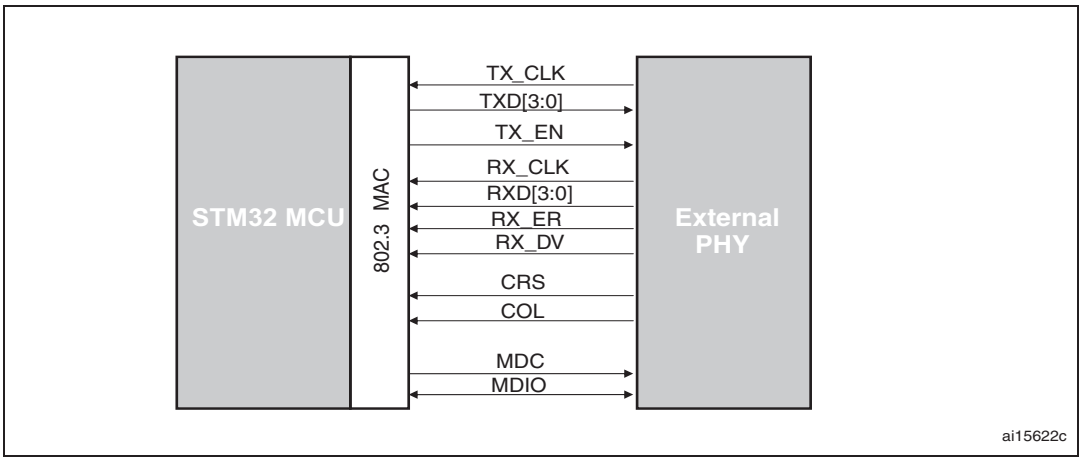
Таблица 209. Диапазон частот

Выбор	Частота HCLK	Частота MDC
000	60-72 MHz	Такты AHB / 42
001	Резерв	—
010	20-35 MHz	Такты AHB / 16
011	35-60 MHz	Такты AHB / 26
0100, 0101, 0110, 0111	Резерв	—

33.4.2. Независимый от среды интерфейс: MII

Определяет соединение подуровня MAC и PHY при передачах на 10 Мбит/с и 100 Мбит/с.

Сигналы интерфейса MII.



- **MII_TX_CLK**: Тактовый сигнал для TX данных. Частота 2.5 MHz для 10 Мбит/с до 25 MHz для 100 Мбит/с.
- **MII_RX_CLK**: Тактовый сигнал для RX данных. Частота 2.5 MHz для 10 Мбит/с до 25 MHz для 100 Мбит/с.
- **MII_TX_EN**: Разрешение передачи. MAC выставляет полубайты передачи. Ставится синхронно (MII_TX_CLK) с выдачей первого полубайта преамбулы и держится до конца передачи.
- **MII_TXD[3:0]**: 4 бита передаваемых данных от подуровня MAC. MII_TXD[0] это младший бит, MII_TXD[3] - старший. Без стоящего MII_TX_EN на PHY не влияет.
- **MII_CRS**: Несущая есть. Выставляется PHY при наличии приёма или передачи. Им же и снимается. В случае коллизии должен стоять. С тактами TX и RX не синхронизируется. В полном дуплексе подуровню MAC не нужен.

- **MII_COL**: Коллизия. Ставится PHY в середине выборки и должен стоять все время существования коллизии. С тактами TX и RX не синхронизируется. В полном дуплексе подуровню MAC не нужен.
- **MII_RXD[3:0]**: 4 бита принимаемых от PHY данных, первый полубайт ставится синхронно с сигналом MII_RX_DV. MII_RXD[0] это младший бит, MII_RXD[3] - старший. При снятом MII_RX_EN и стоящем MII_RX_ER используются для передачи спецданных от PHY (см. *Таблицу 211*).
- **MII_RX_DV**: Разрешение приёма. Ставится PHY синхронно (MII_RX_CLK) с первым полубайтом фрейма и снимается после передачи последнего до начала следующего такта. Должен включать фрейм, начинаясь не позднее поля SFD.
- **MII_RX_ER**: Ошибка приёма. Ставится не менее чем на один период MII_RX_CLK для извещения MAC об ошибке в фрейме. Смысл определяется вместе с состоянием MII_RX_DV

Таблица 188. Сигналы интерфейса TX

MII_TX_EN	MII_TXD[3:0]	Описание
0	0000 до 1111	Нормальный интер-фрейм
1	0000 до 1111	Нормальная передача данных

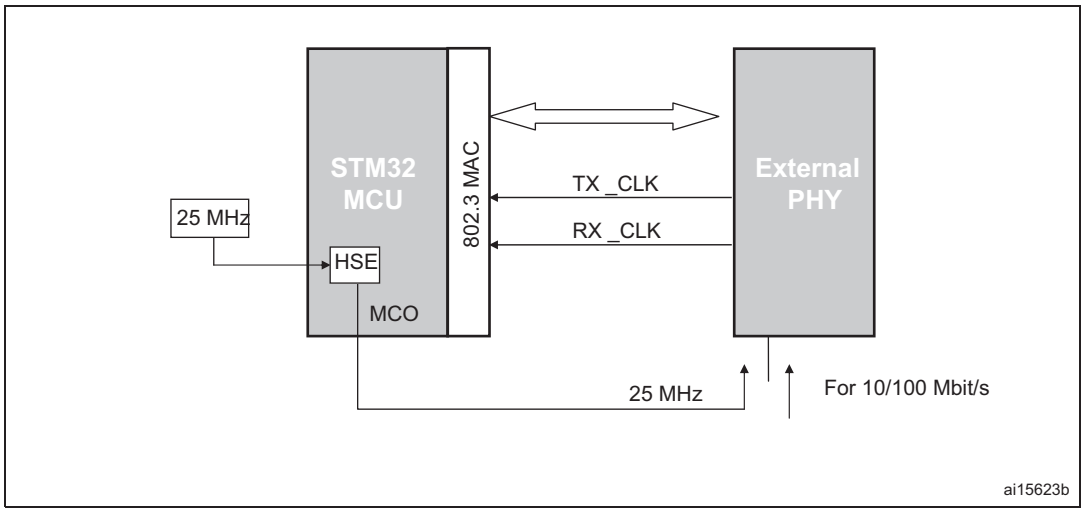
Таблица 189. Сигналы интерфейса RX

MII_RX_DV	MII_RX_ER	MII_RXD[3:0]	Описание
0	0	0000 до 1111	Нормальный интер-фрейм
0	1	0	Нормальный интер-фрейм
0	1	0001 до 1101	Резерв
0	1	1110	Дурная несущая
0	1	1111	Резерв
1	0	0000 до 1111	Нормальный приём данных
1	1	0000 до 1111	Приём данных с ошибками

Источники тактов MII

Для выдачи сигналов TX_CLK и RX_CLK на внешний PHY надо подать внешние 25 MHz. Вместо внешнего 25 MHz кварца можно подать его с ножки MCO микроконтроллера STM32F10xxx. В этом случае надо правильно сконфигурировать PLL с внешним 25 MHz кварцем.

Источники тактов MII



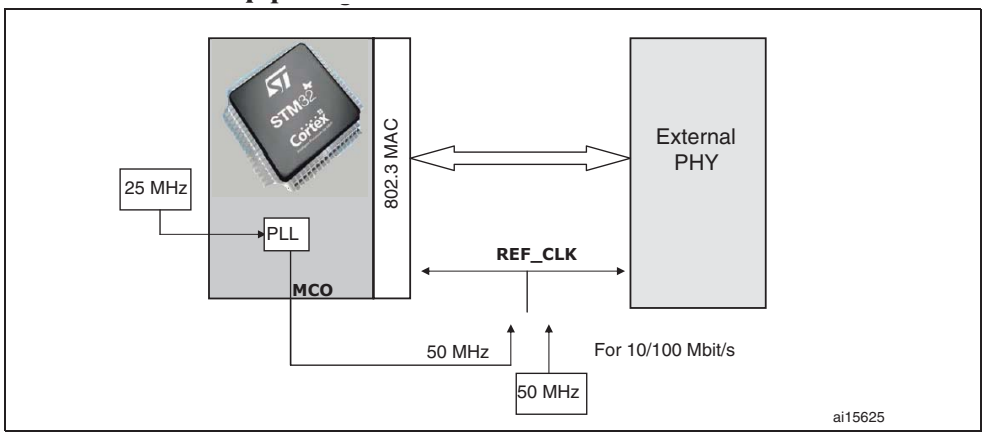
33.4.3. Сокращённый независимый от среды интерфейс: RMII

Он уменьшает число ножек соединения микроконтроллера и PHY с 16 до 7.

RMII стоит между MAC и PHY для трансляции интерфейса MII в RMII. Характеристики такие:

- Поддержка скоростей 10-Mbit/s и 100-Mbit/s
- Опорная частота удваивается до 50 MHz
- Опорная частота подаётся и на MAC и на внешний Ethernet PHY
- Обеспечивает независимые 2-бит (дибит) приём и передачу

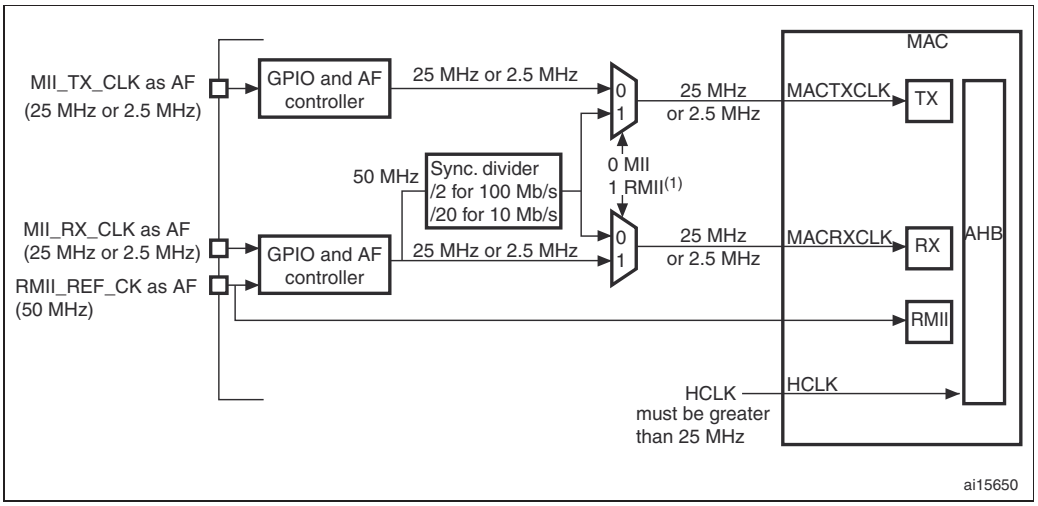
Сигналы интерфейса RMII.



33.4.4. Выбор режима MII/RMII

Режим определяется битом `MII_RMII_SEL` в регистре `AFIO_MAPR`. Писать бит надо при сброшенном контроллере или до подачи тактов.

Схема внутреннего тактирования MII/RMII



Сигналы, RMII_REF_CK и MII_RX_CLK мультиплексируются на одну ножку GPIO.

33.5. Функциональное описание Ethernet: MAC 802.3

Блок MAC реализует подуровень LAN CSMA/CD 10 Mbit/s и 100 Mbit/s узкополосных и широкополосных систем. Поддерживаются полу- и полный дуплекс. Коллизии обнаруживаются только в полудуплексе. Поддерживается подуровень управляющих фреймов MAC.

Он выполняет следующие функции:

- Инкапсуляция данных (передача и приём)
  - Разделение границ и синхронизация фреймов
  - Адресация (источник и получатель)
  - Обнаружение ошибок
- Управление доступом к среде
  - Срединное размещение (предотвращение коллизий)
  - Разрешение конфликтов (обработка коллизий)

Два режима работы подуровня MAC:

- Полудуплекс: доступ к среде по алгоритмам CSMA/CD.
- Полный дуплекс: одновременные приём и передача (CSMA/CD не нужны) когда:
  - Физическая среда поддерживает одновременные приём и передачу
  - К LAN подключены только 2
  - Обе станции работают в полном дуплексе

33.5.1. Формат фрейма MAC 802.3

CSMA/CD MAC определяет два формата фреймов:

- Базовый формат фрейма MAC
- Фрейм MAC с тегами (расширение базового формата)

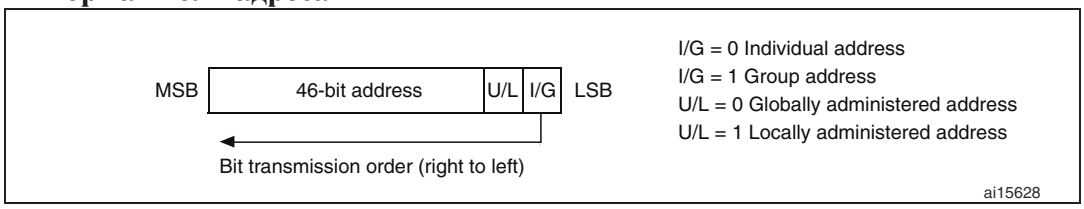
Фреймы содержат следующие поля:

- Преамбула: 7-байтовое поле синхронизации (схема PLS)
  Шестнадцатиричное: 55-55-55-55-55-55-55
  Биты: 01010101 01010101 01010101 01010101 01010101 01010101 01010101 (справа налево)
- Старт фрейма (SFD): 1-байтовое поле.
  Шестнадцатиричное: D5
  Биты: 11010101 (справа налево)
- 6-байтовые поля адресов источника и получателя:
  - Каждый адрес по 48 бит длиной
  - Младший бит адреса получателя (I/G) определяет индивидуальный (I/G = 0) или групповой (I/G = 1) адрес. он может определять все, несколько или отсутствующие станции в LAN. Младший бит адреса источника всегда равен 0.
  - Второй бит (U/L) отличает локальное (U/L = 1) или глобальное (U/L = 0) администрирование адресов. У широковещательных адресов этот бит всегда равен 1.
  - Все байты передаются младшим битом вперёд.

Есть два типа адресов назначения:

- Индивидуальный адрес отдельной станции в сети.
- Групповой адрес двух типов:
  - Групповой адрес для нескольких станций в сети.
  - Широковещательный адрес (все 1 в поле адреса получателя) для всех станций в сети.

Формат поля адреса.



- Префикс QTag: 4-байт поле между полями Адреса Источника и Длины/Типа Клиента MAC. Это расширение базового формата (там его нет). Состоит из:
  - 2-байтовая константа Длины/Типа, совместимая с интерпретацией Типа (больше 0x0600) равная значению 802.1Q Tag Protocol Type (0x8100). Отличает теговые фреймы MAC.
  - 2-байтовое поле Тега, состоящее из: 3- бит приоритета пользователя, бита канонического индикатора формата (CFI) и 12-бит идентификатора VLAN.
- Длина/Тип Клиента MAC: 2-байтовое поле с взаимоисключающими значениями:
  - Если оно меньше или равно maxValidFrame (0d1500), то это число байтов данных клиента MAC в последующем поле данных фрейма 802.3 (интерпретация длины).
  - Если оно больше или равно MinTypeValue (0d1536 или 0x0600), то это тип протокола клиента MAC (интерпретация типа) фрейма Ethernet.

  Независимо от интерпретации поля длины/типа, если длина поля данных меньше требуемого для нормальной работы протокола, то после поля данных перед полем FCS (последовательность проверки фрейма) добавляется расширитель (PAD). Поле длины/типа передаётся начиная со старшего байта.

  Значения поля длины/типа между maxValidLength и minTypeValue (исключая границы) подуровнем MAC могут не передаваться.
- Поля данных и PAD: Любые данные. Общий размер:
  - Максимальная длина = 1500 байт
  - Минимальная длина бестегового фрейма MAC = 46 байт
  - Минимальная длина тегового фрейма MAC = 42 байта

PAD добавляется для получения минимальной общей длины.

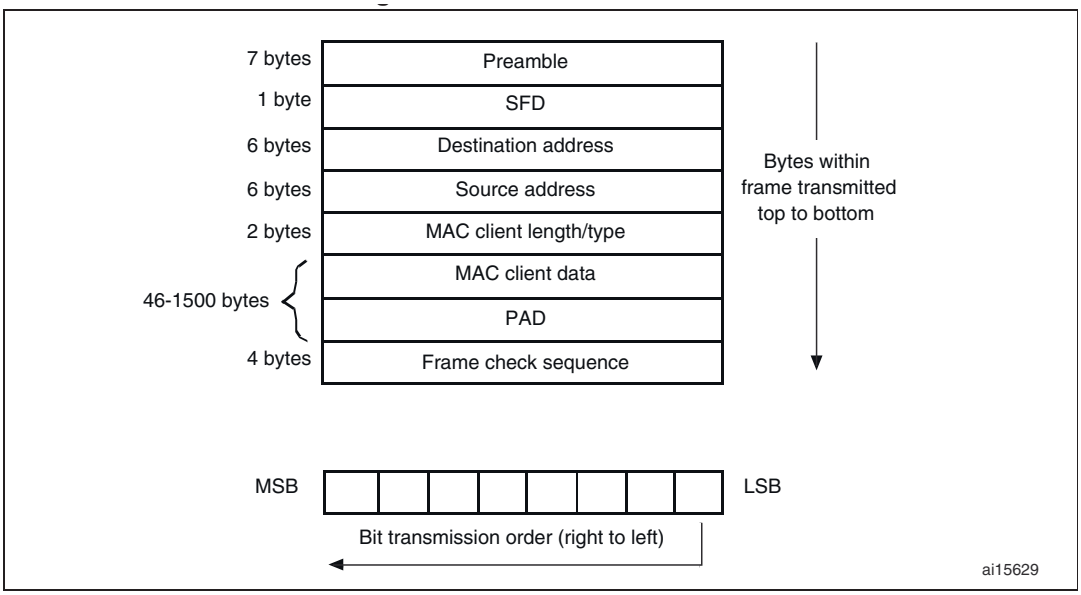
- Последовательность проверки фрейма: 4-байтовое поле с CRC. В вычислении CRC участвуют все поля, кроме преамбулы и SFD. Полином такой:

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

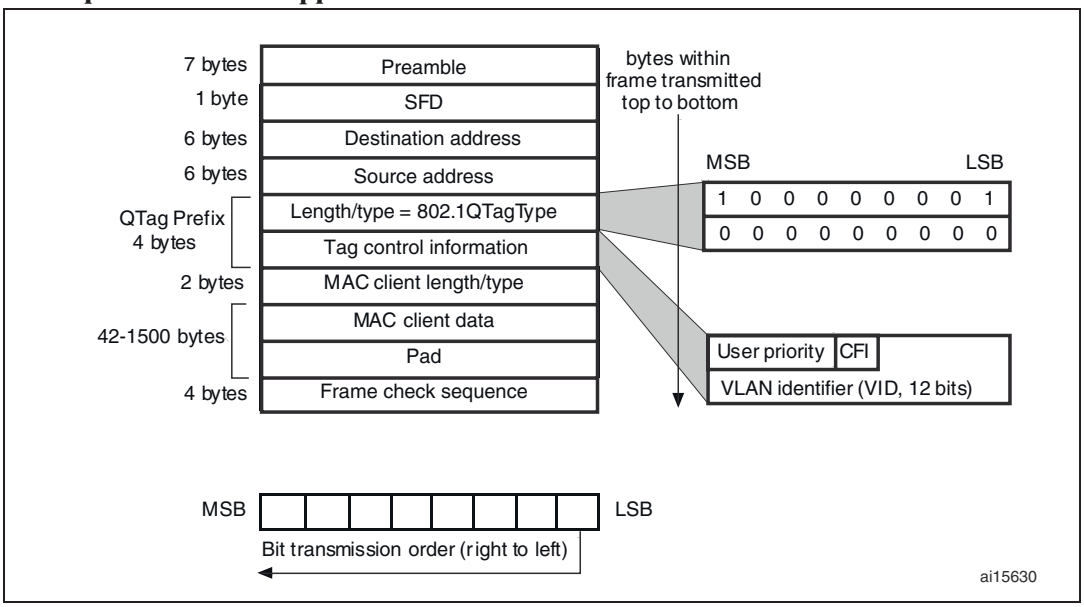
CRC фрейма вычисляется так:

- Первые 2 бита фрейма комплементируются
- n-битов фрейма это коэффициенты полинома $M(x)$ степени $(n - 1)$. Первый бит адреса получателя соответствует x^{n-1} и последний бит поля данных соответствует x^0
- $M(x)$ умножается на x^{32} и делится на $G(x)$, выдавая остаток $R(x)$ степени ≤31
- Коэффициенты $R(x)$ рассматриваются как 32-бит последовательность
- CRC получается комплементом этой последовательности
- 32-бит CRC помещается в последовательности проверки фрейма. x^{32} передаётся первым

Формат фрейма MAC.



Формат тегового фрейма MAC.



Все байты фрейма, кроме поля FCS, передаются младшим битом вперёд. Недостойный фрейм MAC определяется так:

- Длина фрейма не соответствует указанной в поле длины/типа. Если указан тип, то должна быть соответствующая ему длина
- Не целое число байтов (лишние биты)
- Вычисленный CRC не равен переданному в FCS

33.5.2. Передача фрейма MAC

Фреймы из системной памяти в FIFO передаются через DMA, оттуда их выталкивает ядро MAC. После передачи конца фрейма состояние передачи возвращается DMA. Глубина передающего FIFO равна 2 КБ. При передаче данных DMA использует уровень заполнения FIFO.

При появлении SOF, MAC принимает данные и начинает передачу в MII. Время передачи зависит от задержки IFG, времени передачи преамбулы/SFD и задержки подтверждения полу-дуплекса. После получения EOF ядро MAC прекращает передачу и возвращает DMA статус передачи. При появлении коллизии (в полу-дуплексе) ядро MAC отмечает это в статусе и отбрасывает дальнейшие данные до получения SOF. Этот фрейм надо передать повторно начиная с SOF. Если передаваемые данные не поступают вовремя, то MAC выдаёт исчерпание. Если новый SOF поступает до получения EOF предыдущего фрейма, то SOF игнорируется и новый фрейм рассматривается как продолжение предыдущего.

Есть два режима выдачи данных ядру MAC:

- В режиме Порога данные передаются ядру MAC после заполнения FIFO выше порога, заданного битами TTC в регистре ETH_DMABMR, или при получении EOF.
- В режиме Накопления (Store-and-forward) данные передаются ядру MAC только после записи в FIFO всего фрейма. Если TxFIFO меньше передаваемого фрейма, то данные передаются ядру MAC при почти полном TxFIFO.

Передающий FIFO сливается установкой бита FTF в регистре ETH_DMAOMR. По окончанию слива бит аппаратно снимается и FIFO возвращается в исходное состояние. Если бит FTF ставится во время передачи данных ядру MAC, то передача останавливается и FIFO становится пустым. Возникает Исчерпание и DMA передаётся соответствующий статус.

Автоматическое формирование CRC и расширения

Если число байтов для фрейма (DA+SA+LT+Data) меньше 60, то в данные добавляются нули до получения 46 байт. Это действие MAC можно запретить. К передаваемым данным в поле FCS добавляется CRC, что тоже можно запретить. Если расширение фрейма разрешено, то обязательно добавится и CRC. CRC вычисляется по полиному:

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x + 1$$

Протокол передачи

Он соответствует спецификации IEEE 802.3/802.3z, это:

- генерация преамбулы и SFD
- генерация джем-последовательности (jam pattern) в полу-дуплексе
- управление длительностью трёпа (Jabber timeout)
- управление потоком в полу-дуплексе (обратное давление)
- генерация статуса передачи фрейма
- логика меток времени в соответствии с IEEE 1588

При запросе передачи нового фрейма MAC перед данными посылает преамбулу и SFD. Преамбула это 7 байтов 0b10101010, а SFD это 1 байт 0b10101011. Окно коллизии определено как 1 слот (время 512 бит для 10/100 Mbit/s Ethernet).

В режиме MII, при появлении коллизии во время передачи от начала фрейма до конца поля CRC, MAC высылает по MII 32-бит джем-последовательность 0x5555 5555 для извещения других станций о возникновении коллизии. Если коллизия возникла при передаче преамбулы, то MAC прекращает передачу преамбулы и SFD и высылает джем.

Таймер трёпа отслеживает, чтобы передача фрейма не превышала 2048 байт. В полу-дуплексе MAC использует механизм отсрочки (обратного давления). По запросу остановки приёма MAC в любом месте приёма отсылает 32 байта Джема, разрешая передачу. Возникает коллизия и удалённая станция стопорится. Программа запрашивает управление передачей установкой бита BPA в регистре ETH_MACFCR. Запрошенный фрейм планируется и передаётся даже при активном обратном давлении. Оно сохраняется активным длительное время (более 16 последовательных коллизий) и удалённая станция стопорит передачу по причине излишних коллизий. В метках времени отмечается момент передачи SFD на шину MII.

Планирование передачи

В полу-дуплексе MAC управляет промежутками между передаваемыми фреймами по усечённому двоичному экспоненциальному алгоритму возврата. Он разрешает передачу нового фрейма после задержек IFG (биты IFG в регистре ETH_MACCR) и возврата. Если передаваемые фреймы появляются раньше отведённого времени IFG, то MII перед запуском передачи ждёт сигнала разрешения от MAC. Счётчик IFG запускается при отключении несущей от MII. В полном дуплексе MAC разрешает передачу по истечении времени IFG. В полу-дуплексе и при IFG равном времени 96 бит MAC следует правилу задержки из спецификации IEEE 802.3. MAC сбрасывает свой счётчик IFG если несущая появляется в течение первых двух третей интервала IFG (время 64-бит). Если несущая появляется во время последней трети интервала IFG, то счётчик не останавливается и передача разрешается после истечения всего интервала IFG.

Управление потоком передачи

В полном дуплексе при стоящем бите TFE в регистре ETH_MACFCR MAC создаёт фреймы Паузы с добавленным CRC и выдаёт их при установке бита FCB в регистре ETH_MACFCR или при заполненном FIFO приёма.

- По установке бита FCB в ETH_MACFCR MAC выдаёт один фрейм Паузы длительностью, заданной в битах PT в регистре ETH_MACFCR. Для окончания или расширения паузы надо выдать новый фрейм с изменённой длительностью паузы.
- При запросе передачи в заполненный FIFO MAC выдаёт фрейм Паузы длительностью, определённой в регистре ETH_MACFCR. Если FIFO остаётся заполненным в течении заданного числа слотов времени (биты PLT в ETH_MACFCR) до истечения этой паузы, то выдаётся новый фрейм паузы, иначе MAC передаёт фрейм с нулевым временем паузы, извещая о готовности принимать новые фреймы.

Передача одного пакета

Общая последовательность такова:

1. Если системе есть что передавать, то контроллер DMA переносит данные из памяти в FIFO вплоть до конца фрейма.
2. При достижении уровня порога или полном пакете в FIFO контроллер DMA начинает передавать данные фрейма ядру MAC. И так продолжается до выдачи из FIFO полного пакета. По завершению фрейма контроллер DMA получает извещение от MAC.

Передача—два пакета в буфере

1. Поскольку перед возвращением дескриптора хосту DMA должен обновить его статус, то в FIFO можно держать два фрейма. DMA переносит второй фрейм в FIFO только при стоящем бите OSF, иначе следующий фрейм извлекается из памяти только после завершения обработки фрейма MAC и освобождения дескрипторов контроллером DMA.
2. При стоящем бите OSF DMA начинает переносить второй фрейм в FIFO сразу после записи первого и не ждёт изменения статуса. Сразу после передачи первого пакета MAC выдаёт DMA статус передачи и, если второй пакет уже в FIFO, то вторая передача должна дождаться статуса первого пакета.

Повторная передача при коллизии в полу-дуплексе

В случае коллизии MAC должен показать это выдачей статуса даже до получения конца фрейма. Затем разрешается повторная передача и фрейм выдаётся из FIFO. После выдачи ядру MAC более 96 байт контроллер FIFO освобождает место и отдаёт его DMA. То есть при превышении порога или коллизии опоздания повторная передача невозможна.

Слив передающего FIFO

Выполняется битом 20 Регистра режима работы. TxFIFO и соответствующие указатели сбрасывается в начальное состояние даже при незавершённой выдаче данных ядру MAC, возникает исчерпание и передача прекращается. В статусе фрейма отмечаются и исчерпание и слив (биты 13 и 1 в TDES0). Во время слива данные в FIFO не поступают. На каждый слитый фрейм (включая незавершённые) выдаются слова статуса со стоящим битом (TDES0 13) для полностью слитых фреймов. Операция слива завершается после приёма слов состояния всех слитых фреймов. Бит слива снимается. Можно принимать начинающиеся с SOF новые фреймы.

Слово статуса передачи

Оно передаётся программе после того как ядро MAC завершит приём или передачу фрейма по сети. Это биты[23:0] в TDES0. Если разрешены метки времени, то вместе со словом состояния выдаётся и 46-битная метка времени.

Контрольная сумма передачи

Протоколы TCP и UDP используют передачу контрольных сумм по сети. Стало быть и контроллер занимается этим.

NB: Контрольная сумма для TCP, UDP и ICMP вычисляется для всего фрейма и вставляется в нужное поле заголовка. Это разрешается для передающего FIFO только в режиме Накопления (Store-and-forward) установкой бита TSF в регистре ETH_ETH_DMAOMR.

Tx FIFO должен уметь хранить передаваемый фрейм целиком, иначе в любом режиме обрабатывается только контрольная сумма заголовка фрейма IPv4.

Два режима вычисления и вставки контрольной суммы определяются битами CIC в TDES1.

• **Контрольная сумма заголовка IP**

В дейтаграммах IPv4 16-бит контрольная сумма лежит в 11-м и 12-м байтах. Она проверяется когда поле Типа равно 0x0800, а поле Версии равно 0x4. Во входящем фрейме она вычисляется заново и пишется в это поле даже при ошибке CRC. Заголовки IPv6 контрольную сумму не хранят. Результат проверки указывается в бите 16 регистра состояния передачи. От ставится если поля Типа и Версии не те или длина фрейма не соответствует заявленной в заголовке. То есть:

a) В дейтаграммах IPv4:

- Принятый тип Ethernet равен 0x0800, а Версия не равна 0x4
- Поле длины заголовка IPv4 содержит число меньше 0x5 (20 байт)
- Общая длина фрейма меньше заявленной в заголовке

b) В дейтаграммах IPv6:

- Принятый тип Ethernet равен 0x86DD, а Версия не равна 0x6
- Принят конец фрейма, а заголовок IPv6 (40 байт) или заголовок расширения ещё не закончился.

• **Контрольная сумма TCP/UDP/ICMP**

Контрольная сумма TCP/UDP/ICMP обслуживает заголовки IPv4 и IPv6 (включая заголовки расширения) и определяет, относятся ли вложенные данные к TCP, UDP или ICMP.

NB:

a) В не-TCP, -UDP или -ICMP/ICMPv6 фреймах контрольная сумма не проверяется и ничего не изменяется.

b) Фрагментированные фреймы IP (IPv4 или IPv6), фреймы IP с обеспечением безопасности (вроде заголовков аутентификации или вложенными данными безопасности) и фреймы IPv6 с заголовками маршрута на контрольную сумму не проверяются.

Контрольная сумма данных TCP, UDP или ICMP вычисляется и вставляется в заголовок двумя методами:

- Первый, псевдо-заголовки TCP, UDP и ICMPv6 не учитываются, они уже есть в поле контрольной суммы принятого фрейма. Принятая контрольная сумма учитывается и заменяется новым значением.
- Второй, поле контрольной суммы игнорируется, данные TCP, UDP и псевдо-заголовков ICMPv6 учитываются и результат пишется в поле контрольной суммы.

NB: Поскольку в пакетах ICMP-по-IPv4 псевдо-заголовков нет, то поле контрольной суммы пакета ICMP всегда должно быть равно 0x0000, иначе пакет получит неверную контрольную сумму.

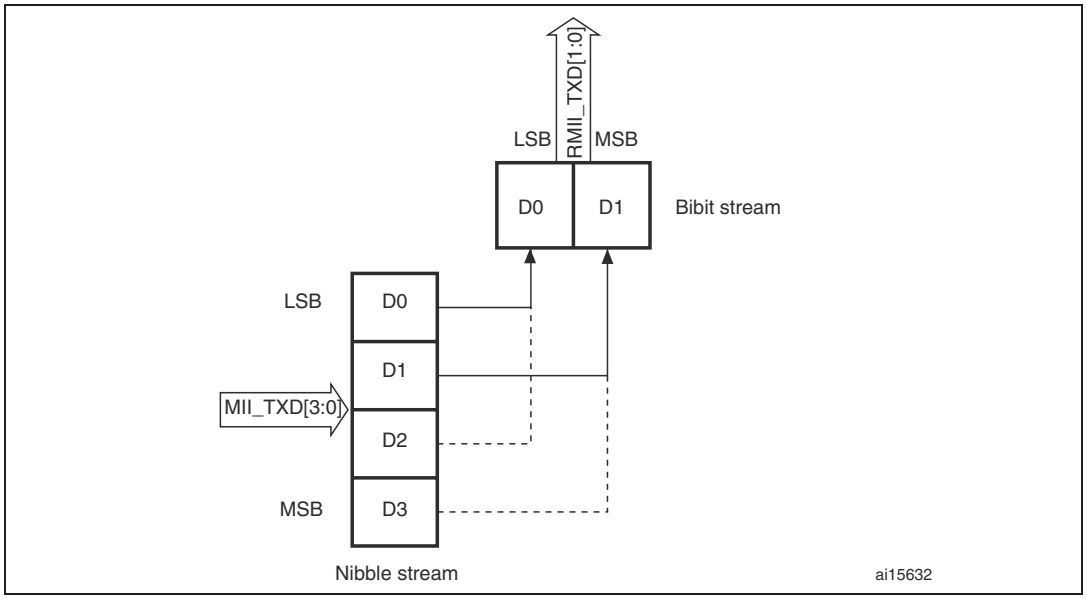
Ошибка контрольной суммы ставится в статусе передачи когда:

- в режиме Накопления на передачу подаётся пакет без Конца фрейма в FIFO
- принятый пакет короче заявленного в принятом заголовке IP.

Если пакет длиннее заявленного, то это байты заполнителя. Они игнорируются и ошибки не возникает. При ошибке первого типа заголовки t TCP, UDP и ICMP не изменяются. При ошибке второго типа новая контрольная сумма всё же записывается.

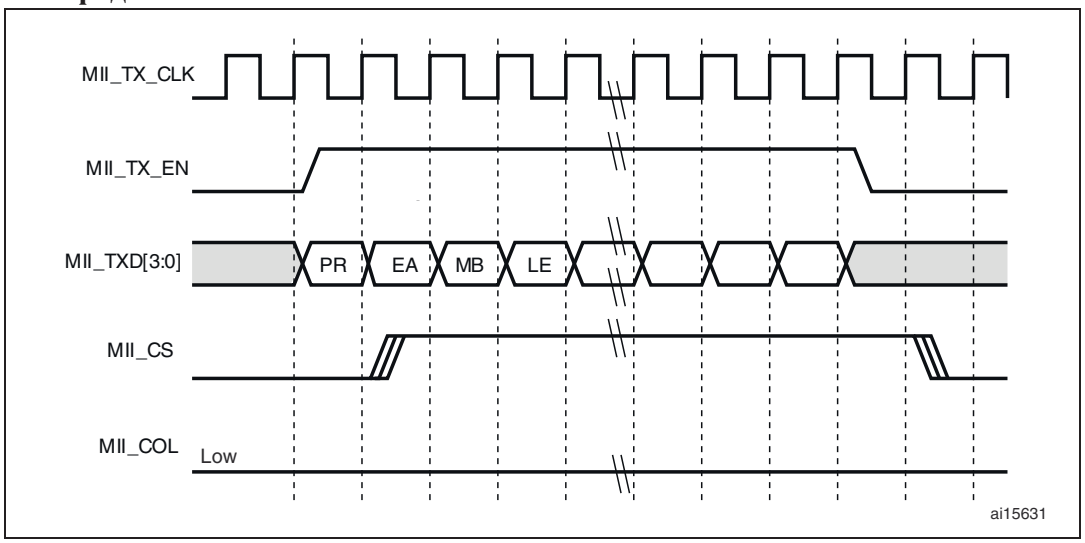
Порядок передачи битов MII/RMII

Полубайты от MII к RMII передаются парами, сначала младшие (D1 и D0) и затем D2 и D3.

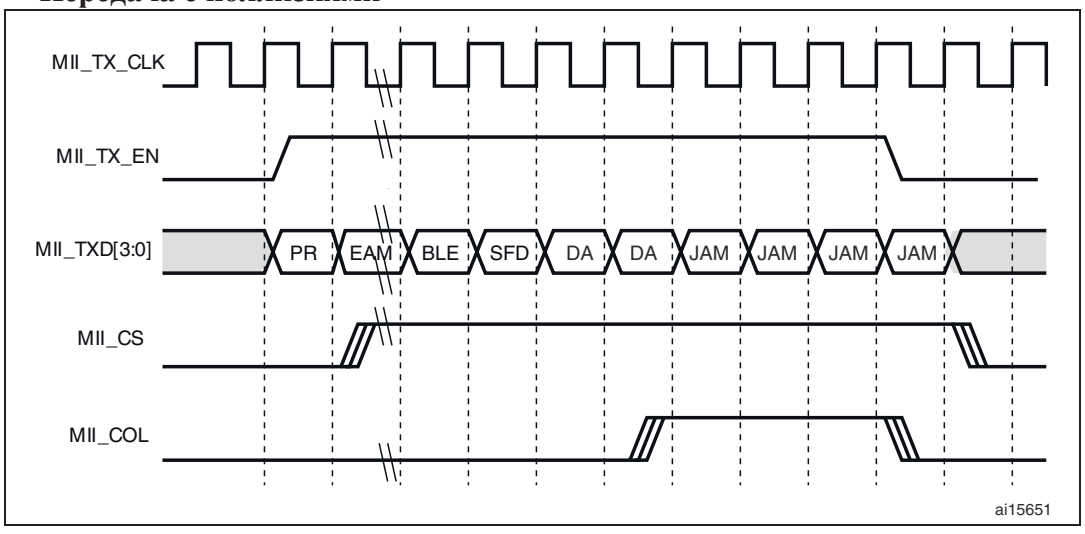


Времянки передач MII/RMII

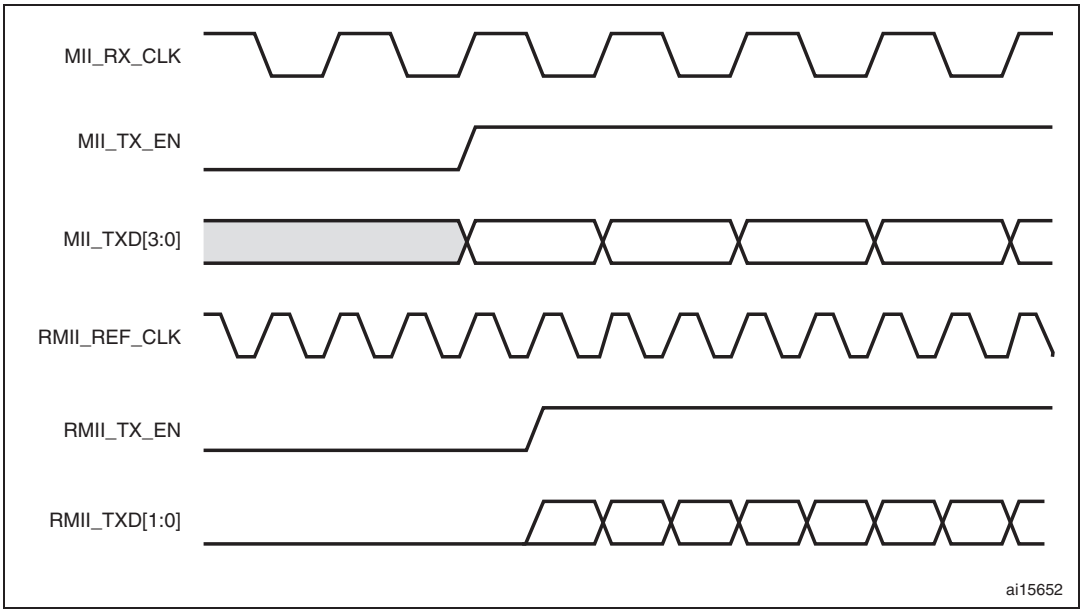
Передача без коллизий



Передача с коллизиями



Передача фрейма в режимах MII и RMII



33.5.3. Приём фрейма MAC

MAC пихает принятые фреймы в RxFIFO. DMA может начать передачи из FIFO в память после получения извещения о достижении порога заполнения (RTC в регистре ETH_DMAOMR).

В режиме по умолчанию DMA получает уведомление о доступности данных при записи в FIFO полного пакета данных или при достижении 64 байт (биты RTC в ETH_DMAOMR). Он передаёт данные из FIFO вплоть до конца пакета. После передачи EOF контроллер DMA получает слово статуса. В память могут попадать и некоторые сбойные фреймы.

В режиме Накопления из RxFIFO (бит RSF в ETH_DMAOMR) извлекаются только завершённые фреймы. Фреймы со сбоями выбрасываются (если это дозволено ядру).

Приём запускается при появлении SFD на MII. Преамбула и SFD отсекаются ядром. Поля заголовков проверяются фильтрами, поле FCS используется для проверки CRC фрейма. Фреймы, не прошедшие фильтры адреса, выбрасываются.

Протокол приёма

Отбрасываются преамбула и SFD принятого фрейма. По получению SFD, MAC пишет данные фрейма в FIFO, начиная с адреса получателя. Метки времени IEEE 1588, если разрешены, отмечают системное время появления SFD на MII. Программе они подаются независимо от результата фильтрации пакетов.

Если значение поля длины/типа меньше 0x600 и разрешена авто-стрижка CRC/расширителя, то MAC отсылает в RxFIFO ровно указанное число байтов, безвозвратно теряя остальные (включая поле FCS).

Если значение поля длины/типа больше или равно 0x600, то MAC отсылает в RxFIFO все полученные байты, независимо от разрешения авто-стрижки. Сторожевой таймер MAC включён по умолчанию, то есть, фреймы длиннее 2048 байт (DA + SA + LT + Data + pad + FCS) подвергаются обрезанию. Это выключается битом WD в регистре конфигурации MAC. Фреймы длиннее 16 KB обрезаются в любом случае и выдаётся статус таймаута сторожевого таймера.

Приём CRC: автоматическое удаление CRC и расширения

32-бит CRC полей от адреса назначения до FCS вычисляется по полиному:

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x + 1$$

Для вычисления CRC, MAC принимает весь фрейм, независимо от разрешения авто-стрижки.

Контрольная сумма приёма

Приём контрольной суммы фреймов IPv4 и IPv6 разрешается установкой бита IPCO в регистре ETH_MACCR. Фреймы IPv4 и IPv6 различают по значениям 0x0800 и 0x86DD поля типа фрейма. Это относится и к фреймам с тегами VLAN. Вычисленная контрольная сумма заголовка IPv4 сравнивается с принятой. Бит ошибки заголовка ставится при любом несоответствии типа данных, версии заголовка IP и числе принятых данных, меньше указанного в заголовке IPv4 (или меньше 20 байтов для заголовков IPv4 и IPv6). Принятая контрольная сумма различает данные TCP, UDP и

ICMP в принятой дейтаграмме IP (IPv4 и IPv6). Проверка включает байты псевдо-заголовка TCP/UDP/ICMPv6. По результату ставится бит ошибки контрольной суммы в слове статуса. Также он ставится при несовпадении длины данных TCP, UDP и ICMP с заявленной в заголовке IP. Фрагментированные фреймы IP (IPv4 или IPv6), фреймы IP с обеспечением безопасности (вроде заголовков аутентификации или вложенными данными безопасности) и фреймы IPv6 с заголовками маршрута на контрольную сумму не проверяются. Факт проверки контрольной суммы отмечается в статусе приёма. В этой конфигурации байты контрольной суммы к принятым фреймам Ethernet не добавляются.

Таблица 190. Статусы фрейма

Бит 18: Фрейм Ethernet	Бит 27: Ошибка контрольной суммы заголовка	Бит 28: Ошибка контрольной суммы данных	Статус фрейма
0	0	0	Фрейм IEEE 802.3 (Поле длины меньше 0x0600).
1	0	0	Фрейм IPv4/IPv6 с ошибкой контрольной суммы.
1	0	1	Фрейм IPv4/IPv6 с ошибкой контрольной суммы данных
1	1	0	Фрейм IPv4/IPv6 с ошибкой контрольной суммы заголовка.
1	1	1	Фрейм IPv4/IPv6 с ошибкой контрольной суммы заголовка и данных.
0	0	1	Фрейм IPv4/IPv6 с ошибкой контрольной суммы заголовка без проверки данных.
0	1	1	Фрейм не типа IPv4 или IPv6 (без проверки)
0	1	0	Резерв

Контроллер фрейма приёма

Бит RA в регистре фильтра фреймов включает фильтрацию по адресам источника/получателя. При динамическом изменении параметров фильтра и сбое фильтра (DA-SA остаток фрейма отбрасывается и в слове статуса приёма отмечается причина сбоя. В режиме выключения питания Ethernet выбрасываются все принятые фреймы.

Управление приёмом

В полном дуплексе MAC может обнаруживать фреймы Паузы и приостанавливать передачу фреймов на указанное время. Это разрешается битом RFCE регистра ETH_MACFCR. При разрешённом управлении потоком приёма из него выявляются управляющие фреймы с множественным адресом назначения (0x0180 C200 0001). Бит PCF в регистре ETH_MACFFR разрешает передавать их программе.

В управляющем фрейме MAC различает поля типа, операции и таймера паузы. Если счётчик принятых байтов управляющего фрейма без ошибки CRC равен 64, то передатчик MAC приостанавливает выдачу фреймов на принятое число времени слота (64 байта режимов 10/100 Mbit/s). По фрейму паузы с нулевым временем текущая пауза обрывается.

Если тип управляющего фрейма не равен 0x8808), код не равен 0x00001 и длина не равна 64 байта, то пауза не генерируется. Фрейм паузы с множественным адресом получателя проходит фильтр.

При стоящем бите UPDF в регистре ETH_MACFCR фрейм паузы с уникальным адресом получателя проверяется по регистру 0 адреса MAC. Биты PCF регистра ETH_MACFFR определяют дополнительную фильтрацию адресов.

Приём нескольких фреймов

Поскольку статус следует сразу за данными, то FIFO может хранить любое число фреймов, вплоть до заполнения.

Обработка ошибок

Если RxFIFO заполняется до приёма EOF от MAC, то в статусе объявляется переполнение, выбрасывается весь фрейм и инкрементируется счётчик переполнений (регистр ETH_DMAMFBOCR). Биты FEF и FUGF в регистре ETH_DMAOMR позволяют RxFIFO отфильтровывать сбойные и маломерные фреймы.

В режиме Накопления (Store-and-forward) RxFIFO может отсевать все сбойные фреймы.

В режиме по умолчанию (Cut-through) RxFIFO может отсевать завершённые сбойные фреймы, незавершённые должен сливать DMA.

Слово статуса приёма

Закончив приём фрейма Ethernet, MAC передаёт DMA слово статуса приёма в битах[31:0] RDES0.

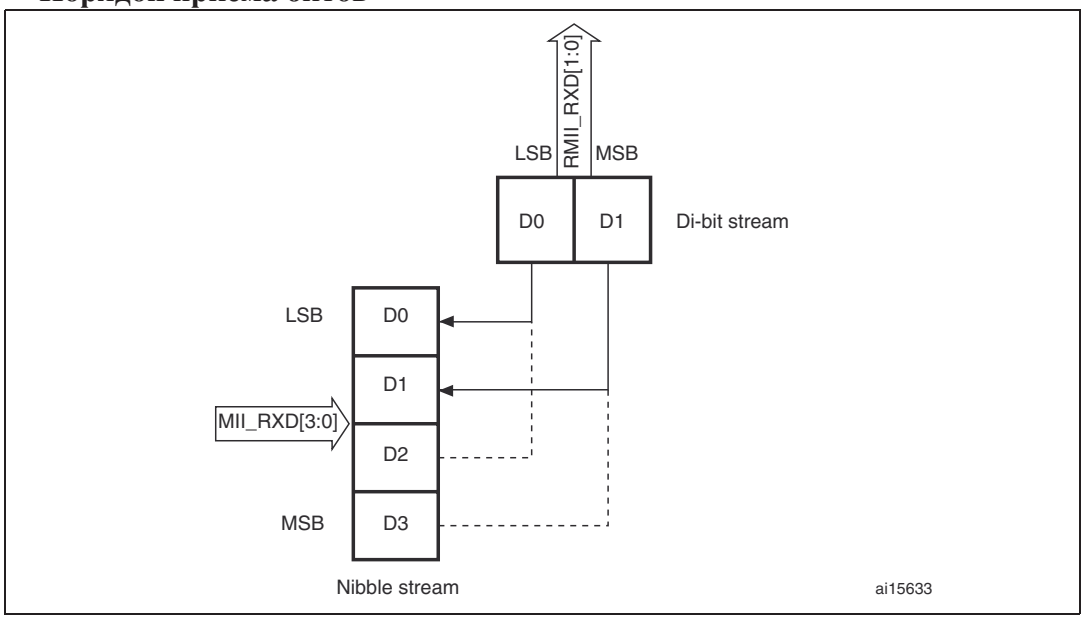
Длина приёма

В коммутаторах сети MAC обменивается с системой полными фреймами. Так что следить за длиной входящих фреймов для передачи их на выход нужно на уровне приложения. Фреймы нулевой длины попадают в RxFIFO при переполнении.

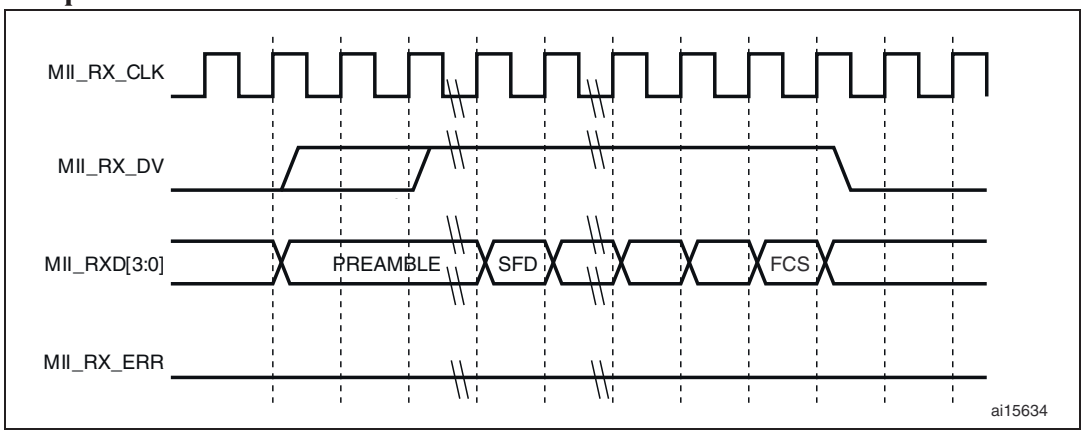
Порядок приёма битов MII/RMII

Полубайты из RMII в MII поступают парами, начиная с D0 и D1, затем D2 и D3)

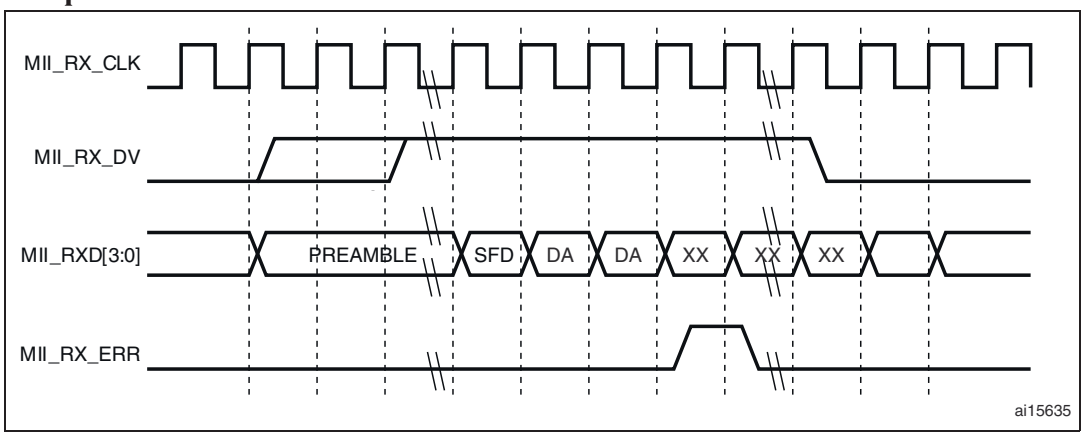
Порядок приёма битов



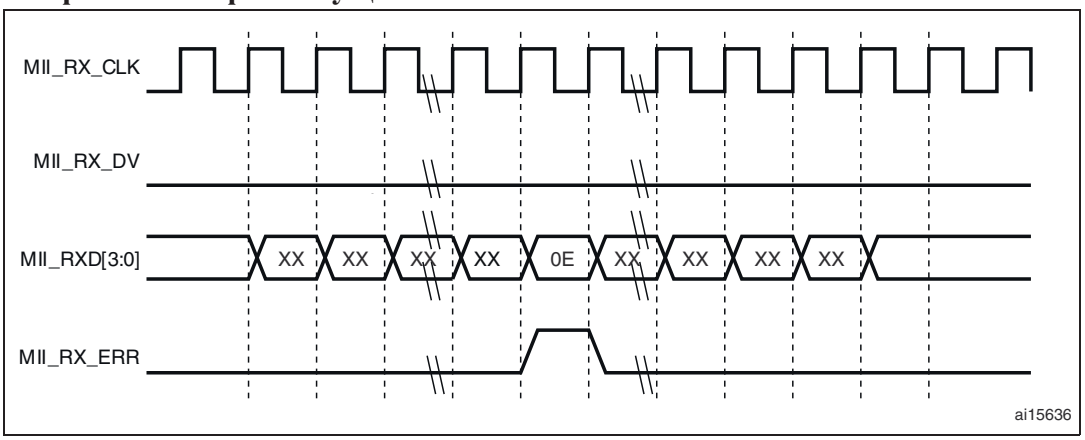
Приём без ошибок.



Приём с ошибками



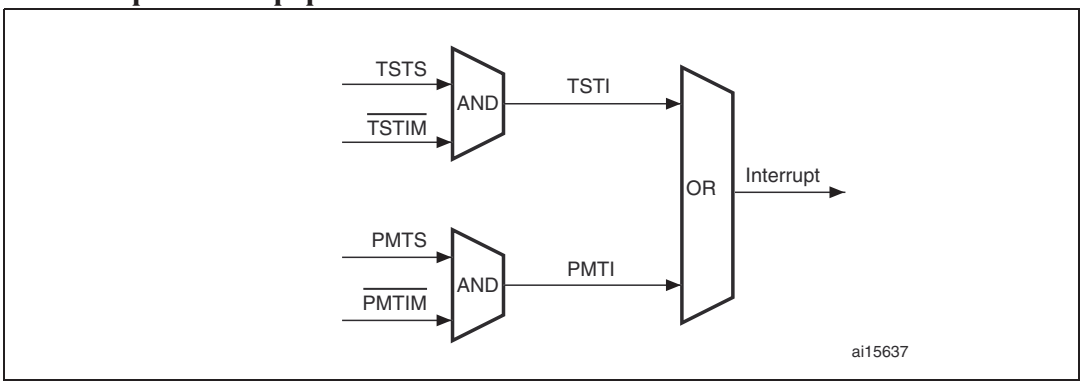
Приём с потерей несущей



33.5.4. Прерывания MAC

Источники прерываний отмечаются в регистре ETH_MACSR. Прерывания по ним разрешаются в регистре маски прерываний. Очищаются биты прерываний чтением соответствующих регистров состояния и других регистров источников.

Маскирование прерываний MAC



33.5.5. Фильтры MAC

Фильтры адреса

Все полученные фреймы фильтруются по адресу источника и назначения и им ставится статус. Параметры фильтрации устанавливаются программой. При фильтрации используется физический адрес (MAC) и Хэш-таблица.

Фильтр уникального адреса назначения

Тут используется до 4 MAC адресов. При снятом бите HU в регистре фильтра фрейма сравниваются все 48 бит полученного адреса. MacAddr0 разрешён всегда, адреса MacAddr1–MacAddr3 имеют свои биты разрешения. Так фильтруются групповые адреса. При стоящем бите HU для фильтрации используется 64-бит Хэш-таблица. Для этого в качестве её индекса используются 6 старших битов CRC. Значение 000000 соответствует биту 0, а 111111 выбирает бит 63 регистра Хэш-таблицы. Если выбранный бит стоит, то фрейм проходит сквозь фильтр.

Фильтр множественного адреса назначения

При стоящем бите PAM в регистре фильтра фреймов MAC пропускает все фреймы с множественными адресами. Если он сброшен, то фильтрация определяется битом HM этого регистра. В режиме полной фильтрации сравниваются регистры 1-3 адреса получателя MAC. Фильтрация групповых адресов поддерживается. При хэш-фильтрации используется 64-бит Хэш-таблица. Для этого в качестве её индекса используются 6 старших битов CRC. Значение 000000 соответствует биту 0, а 111111 выбирает бит 63 регистра Хэш-таблицы. Если выбранный бит стоит, то фрейм проходит сквозь фильтр.

Фильтр Хэш или полного адреса

Стоящий бит HPF в регистре фильтра позволяет пропускать фреймы, прошедшие один из фильтров DA (полного адреса или хэш) в зависимости от битов HU и HM bits. Если бит HPF сброшен, то работает только один фильтр.

Фильтр широковещательного адреса

При стоящем бите BFD в регистре фильтра все фреймы с широковещательными адресами теряются.

Фильтр уникального адреса источника

По умолчанию поле SA сравнивается с регистрами SA. Установка бита 30 в регистре DA ([1:3]) превращает его в SA. Групповая фильтрация SA поддерживается. При стоящем бите SAF в регистре фильтра фреймы, не прошедшие фильтр, отбрасываются, иначе им просто отмечается этот статус.

Инверсная фильтрация

Биты DAIF и SAIF позволяют инвертировать условие фильтрации фреймов с уникальными и множественными адресами.

Таблица 191. Фильтрация адреса назначения

Тип фрейма	PM	HPF	HU	DAIF	HM	PAM	DB	Что делает фильтр DA
Широковещательный	1	X	X	X	X	X	X	Пропускает
	0	X	X	X	X	X	0	Пропускает
	0	X	X	X	X	X	1	Выбрасывает
Уникальный	1	X	X	X	X	X	X	Пропускает все фреймы
	0	X	0	0	X	X	X	Пропускает по полному/групповому сравнению
	0	X	0	1	X	X	X	Выбрасывает по полному/групповому сравнению
	0	0	1	0	X	X	X	Пропускает по хэш сравнению
	0	0	1	1	X	X	X	Выбрасывает по хэш сравнению
	0	1	1	0	X	X	X	Пропускает по хэш или полному/групповом сравнению
	0	1	1	1	X	X	X	Выбрасывает по хэш или полному/групповом сравнению
Множественный	1	X	X	X	X	X	X	Пропускает все фреймы
	X	X	X	X	X	1	X	Пропускает все фреймы
	0	X	X	0	0	0	X	Пропускает по полному/групповому сравнению, теряет фреймы PAUSE при PCF = 0x
	0	0	X	0	1	0	X	Пропускает по хэш сравнению, теряет фреймы PAUSE при PCF = 0x
	0	1	X	0	1	0	X	Пропускает по хэш или полному/групповом сравнению, теряет фреймы PAUSE при PCF = 0x
	0	X	X	1	0	0	X	Выбрасывает по полному/групповом сравнению, теряет фреймы PAUSE при PCF = 0x
	0	0	X	1	1	0	X	Выбрасывает по хэш сравнению, теряет фреймы PAUSE при PCF = 0x
	0	1	X	1	1	0	X	Выбрасывает по хэш или полному/групповом сравнению, теряет фреймы PAUSE при PCF = 0x

Таблица 192. Фильтрация адреса источника

Тип фрейма	PM	SAIF	SAF	Что делает фильтр SA
Unicast	1	X	X	Пропускает все фреймы
	0	0	0	Пропускает статус по полному/групповом сравнению, фреймы не теряет
	0	1	0	Выбрасывает статус по полному/групповом сравнению, фреймы не теряет
	0	0	1	Пропускает по полному/групповом сравнению
	0	1	1	Выбрасывает по полному/групповом сравнению

33.5.6. Перемычка MAC

Подача передаваемых фреймов на приёмник разрешается битом Loopback в регистре ETH_MACCR.

33.5.7. Счётчики работы MAC: MMC

Это сбор статистики по передаваемым и принимаемым фреймам. Включает также 32-бит регистр управления, два 32-бит регистра общих прерываний (приём и передача) и два 32-регистра маски этих прерываний. На приёме учитываются только фреймы, прошедшие фильтр. В числе отбросов фреймы-коротышы учитываются.

Хорошо переданные и принятые фреймы

У хорошей передачи не появились ошибки:

+ Таймаут Трёпа
+ Нет/Потеря несущей
+ Поздняя коллизия
+ Исчерпание фрейма
+ Чрезмерная отсрочка
+ Излишние коллизии

У хорошего приёма не было ошибок:

+ Ошибка CRC
+ Фрейм-коротыш (короче 64 байт)
+ Ошибка выравнивания (только 10/100 Mbit/s)
+ Ошибка длины (только фреймы без типа)
+ Вне размера (только фреймы без типа, длиннее максимума)
+ Ошибка ввода MII_RXER

Максимальная длина фрейма:

+ Бестеговый фрейм = 1518
+ Фрейм VLAN = 1522

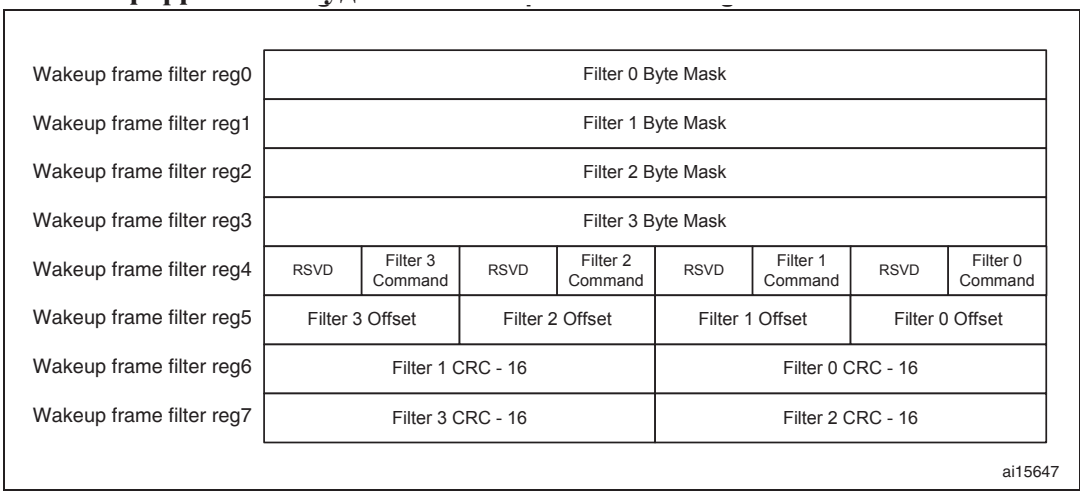
33.5.8. Управление питанием: PMT

Это приём Magic Packet и фреймов удалённой побудки с выдачей прерывания. PMT включается битами `WFE` и `MPE` в регистре `ETH_MACPMTCSR`. При включённом PMT все принятые фреймы отбрасываются. MAC выходит из выключенного питания по приёму Magic Packet или фрейма удалённой побудки при включённом распознавании.

Регистры фильтра фрейма побудки

Их 8 штук. Читают и пишут их последовательно за 8 обращений. При каждом смещается точка доступа к регистрам.

Фильтр фрейма побудки



- **Маска байта фильтра i**
  Определяет байт, проверяемый фильтром **i** (0, 1, 2 и 3) при определении фрейма побудки. MSB должен быть нулевым. Биты **j** [30:0] это маска байта. Если бит **j[номер байта]** стоит, то байт **Смещение фильтра i + j** обрабатывается блоком CRC, иначе игнорируется.

- **Команда фильтра i**
  Здесь 4 бита. Бит 3 это тип адреса получателя, 1 - множественный, 0 - уникальный. Биты 2 и 1 - резерв. Бит 0 это включение фильтра **i**, 1- Вкл.
- **Смещение фильтра i**
  Смещение проверяемого байта от начала фрейма для фильтра i. Минимальное значение равно 12, (13-й байт).
- **CRC-16 фильтра i**
  Нужное значение CRC_16 для проверяемого фрейма.

Обнаружение фрейма удалённой побудки

Выход MAC из режима сна по фрейму побудки разрешается установкой бита 2 в регистре ETH_MACPMTCSR. Программа пишет все 8 регистров фильтра application разрешает удалённую побудку. PMT имеет 4 фильтра для разных фреймов. Если фрейм проходит фильтры команды и CRC-16, то это побудка. Фрейм побудки проверяется на ошибки: длины, FCS, dribble bit, MII, коллизии и на фрейм-коротыш. Длинный фрейм (больше 512 байт) с нужным значением CRC считается побудкой. Обнаружение обновляется в регистре ETH_MACPMTCSR по каждому фрейму побудки. PMT может выдавать прерывание побудки.

Обнаружение Magic packet от фирмы AMD

Адрес получателя у Magic Packet может быть уникальным и широковещательным. Пакет сначала проходит фильтр адреса и затем проверяется на формат данных Wake-on-LAN. Это неразрывная последовательность 6 байтов единиц и 16 повторений адреса MAC. Обнаружение Magic Packet разрешается установкой бита 1 в регистре ETH_MACPMTCSR. PMT постоянно проверяет все адресованные сюда фреймы на последовательность Magic Packet. Она может начинаться в любом месте фрейма за адресами получателя и источника. Пакеты с множественным адресом тоже принимаются. Если адрес MAC узла равен 0x0011 2233 4455, то ищется последовательность:

Адрес назначения адрес источника …… FFFF FFFF FFFF
0011 2233 4455 0011 2233 4455 0011 2233 4455 0011 2233 4455
0011 2233 4455 0011 2233 4455 0011 2233 4455 0011 2233 4455
0011 2233 4455 0011 2233 4455 0011 2233 4455 0011 2233 4455
0011 2233 4455 0011 2233 4455 0011 2233 4455 0011 2233 4455
...CRC

Обнаружение обновляется в регистре ETH_MACPMTCSR по каждому Magic Packet. PMT может выдавать прерывание побудки.

Выключение питания системы

При разрешённой EXTI line 19 блок PMT может обнаруживать фреймы в режиме Stop. При этом машина состояний приёмника должна быть включена стоящим битом RE, а машину состояний передатчика надо выключить очисткой бита TE в регистре ETH_MACCR. Ethernet DMA надо выключить очисткой битов ST и SR в регистре ETH_DMAOMR.

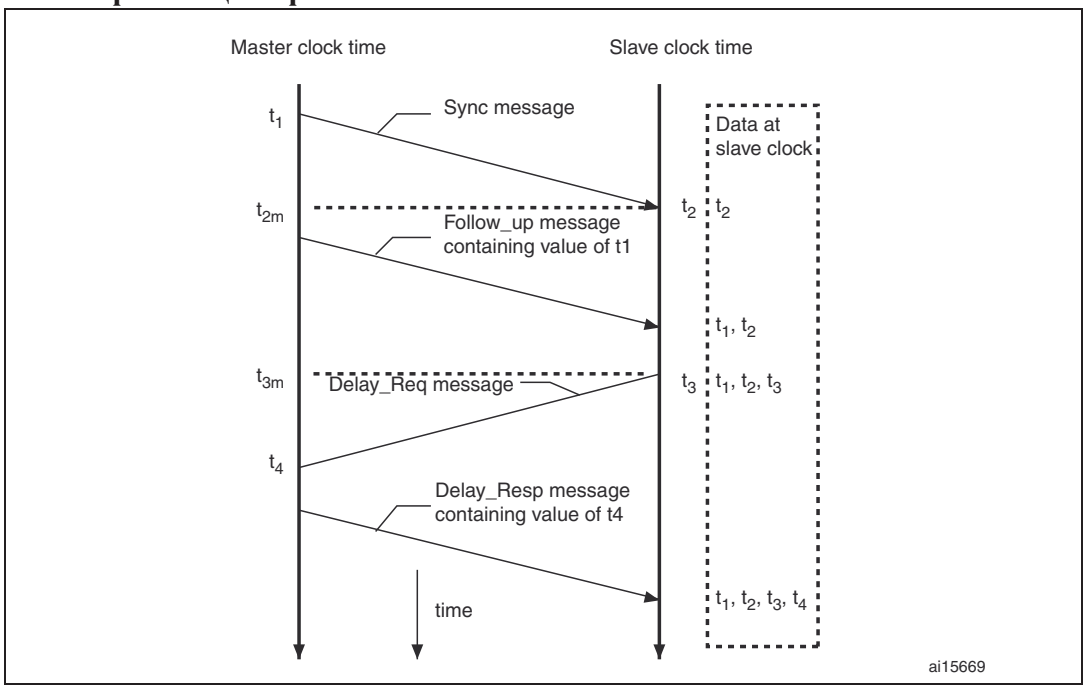
Рекомендуется следующая последовательность выключения и побудки:

1. Выключаем передающий DMA и ждём конца передачи по прерыванию в ETH_DMASR.
2. Выключаем передатчик и приёмник MAC очисткой битов RE и TE в регистре ETH_MACCR.
3. Ждём пока приёмный DMA очистит RxFIFO.
4. Выключаем приёмный DMA.
5. Ставим и разрешаем EXTI line 19 на выдачу события или прерывания.
6. При прерывании, обработчик ETH_WKUP_IRQ должен снимать бит удержания EXTI line 19.
7. Включаем обнаружение побудки установкой бита MFE/ WFE в регистре ETH_MACPMTCSR.
8. Включаем экономный режим MAC установкой бита PD в регистре ETH_MACPMTCSR.
9. Включаем приёмник MAC установкой бита RE в регистре ETH_MACCR.
10. Идём в режим Stop
11. По приёму фрейма побудки блок Ethernet выйдет из режима экономии.
12. Читаем ETH_MACPMTCSR для очистки флага события, включаем машину состояний передатчика и приём и передачу DMA.
13. Конфигурируем системные такты: включаем HSE и ставим частоту.

33.5.9. Протокол точного времени (IEEE1588 PTP)

В системе выделяются ведущий и ведомые узлы.

Синхронизация времени сети



1. Ведущий выдаёт широковещательные сообщения PTP **Sync** с информацией о своём опорном времени всем узлам. Момент выхода из ведущей системы это $\mathbf{t_1}$. Ethernet порты снимают его в момент появления на входе MII.
2. Ведомый принимает сообщение **Sync**, отмечая точный момент прихода по своему времени $\mathbf{t_2}$.
3. Далее ведущий посылает ведомому сообщение **Follow_up**, содержащее $\mathbf{t_1}$ чтобы знали.
4. Ведомый отсылает ведущему сообщение **Delay_Req**, отметив момент выхода из MII, $\mathbf{t_3}$.
5. Ведущий принимает его, отмечая точный момент его его прихода $\mathbf{t_4}$.
6. Ведущий отсылает ведомому время $\mathbf{t_4}$ в сообщении **Delay_Resp**.
7. По значениям $\mathbf{t_1}$, $\mathbf{t_2}$, $\mathbf{t_3}$ и $\mathbf{t_4}$ ведомый синхронизирует своё опорное время с опорным временем ведущего.

В большинстве случаев протокол реализуется поверх уровня UDP, но нужна аппаратная поддержка точных отметок времени входа и выхода пакетов из MII.

Источник опорного времени

Спецификация IEEE 1588 определяет 64-бит формат отметок времени (старшие 32 бита это секунды, младшие 32 бита - наносекунды).

Для учёта опорного времени (Системное Время) и меток времени PTP использует внешний источник с частотой большей или равной разрешению счётчика меток времени. Точность синхронизации времён ведущего и ведомого должна быть около 100 ns. Она зависит от входной частоты, смещения генератора и частоты процедуры синхронизации. Такты Tx и Rx синхронизированы с доменом опорной частоты PTP и точность меток времени равна 1 периоду опорных тактов. Нарушив разрешение, мы добавим в метки времени ещё пол-периода.

Передача фреймов с PTP

Метки времени хранят момент появления SFD на MII. Необходимость снятия метки времени отмечается в дескрипторе передачи фрейма. Метки времени возвращаются программе вместе со словом статуса передачи в дескрипторе. 64-бит метка времени пишется в поля TDES2 и TDES3, TDES2 это 32 младших бита метки.

Приём фреймов с PTP

Если метки времени разрешены, то они отмечают конец приёма любого входного фрейма на MII. Они возвращаются программе вместе со словом статуса приёма в дескрипторе. 64-бит метка времени пишется в поля RDES2 и RDES3, RDES2 это 32 младших бита метки.

Методы коррекции Системного Времени

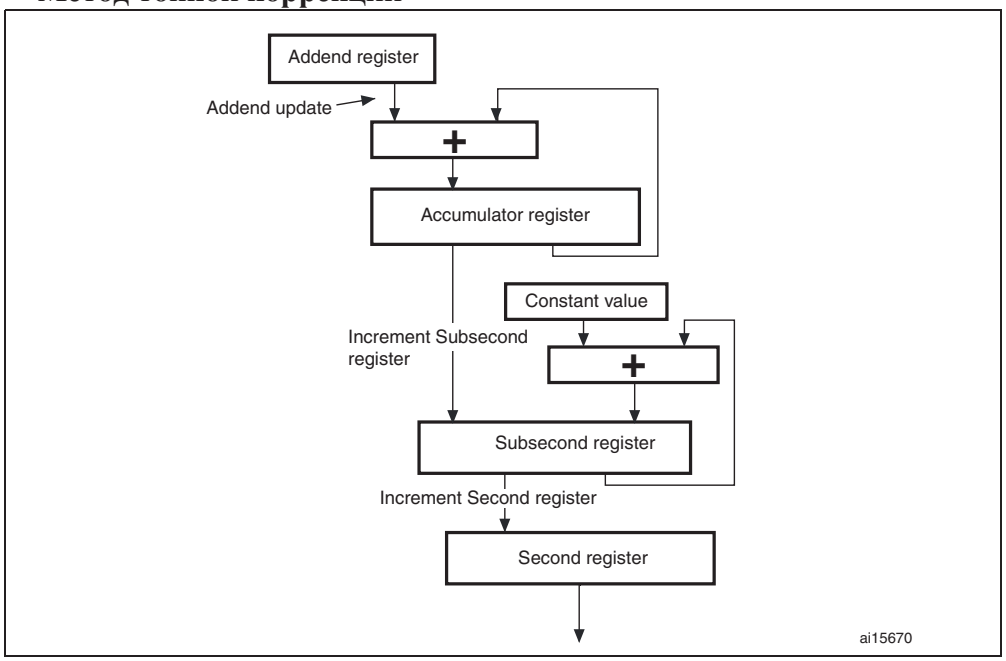
Их два: Грубый и Тонкий.

При грубом методе в регистр обновления метки времени пишется начальное значение или смещение. При инициализации этот регистр пишется в счётчик системного времени, при коррекции это значение добавляется или вычитается из счётчика.

При тонком методе отклонение частоты тактов ведомого от ведущего определяется за некоторый период времени, тогда как при грубом методе за один такт. Это такое сглаживание изменений.

При этом методе 32-бит аккумулятор накапливает содержимое 32-бит регистра слагаемого (Addend). Арифметический перенос аккумулятора используется для инкремента счётчика системного времени. Здесь аккумулятор служит высокоточным умножителем или делителем частоты.

Метод тонкой коррекции



Логика обновления системного времени требует частоты тактов 50 MHz для достижения 20 ns точности. Стало быть, если HCLK равна 66 MHz, то отношение частот FreqDivisionRatio равно 66 MHz/50 MHz = 1.32. Отсюда, слагаемое в регистре равно $2^{32}/1.32$ (0xC1F0 7C1F).

Например, если опорная частота уплывает до 65 MHz, то отношение 65/50 равно 1.3 и слагаемое становится равным $2^{32}/1.30$ (0xC4EC 4EC4). Если частота уходит в 67 MHz, То слагаемое становится равным 0xBF0 B7672. При нулевом отклонении в регистр слагаемого пишется 0xC1F0 7C1F ($2^{32}/1.32$).

На рисунке выше для инкремента суб-секундного регистра используется константа 0d43. Так достигается 20 ns точность системного времени.

Программа обновляет регистр слагаемого на основании сообщений **Sync**. Сначала в регистр слагаемого пишем значение компенсации частот:

FreqCompensationValue(0) = 2^{32}/FreqDivisionRatio

Если MasterToSlaveDelay (Задержка Ведущий-Ведомый) изначально полагается такой же как у последовательных сообщений **Sync**, то через несколько циклов **Sync** её можно определить точней и синхронизироваться с ведущим по алгоритму:

- В момент времени MasterSyncTime(n) ведущий посылает сообщение **Sync**. Ведомый получает его в момент SlaveClockTime(n) и вычисляет время ведущего:
  MasterClockTime(n) = MasterSyncTime(n) + MasterToSlaveDelay(n)
- Значение счётчика ведущего в текущем цикле **Sync** вычисляем как:
  MasterClockCount(n) = MasterClockTime(n) – MasterClockTime(n-1) (полагая, MasterToSlaveDelay одинаков для циклов **Sync** за номерами n и n – 1)
- Значение счётчика ведомого в текущем цикле **Sync** вычисляем как:
  SlaveClockCount(n) = SlaveClockTime(n) – SlaveClockTime(n – 1)
- Разность обоих счётчиков текущего цикла **Sync** равна:
  ClockDiffCount(n) = MasterClockCount(n) – SlaveClockCount(n)
- Множитель масштаба тактов ведомого равен:
  FreqScaleFactor(n) = (MasterClockCount(n) + ClockDiffCount(n)) / SlaveClockCount(n)

- Значение компенсации в регистр слагаемого равно:
  $FreqCompensationValue(n) = FreqScaleFactor(n) \times FreqCompensationValue(n - 1)$

В теории это может сработать в течении одного цикла **Sync**, но в реальности задержки в работающей сети непредсказуемы.

Это алгоритм само-настройки: если начальные установки тактов ведомого относительно ведущего неверны, то через какое-то число циклов **Sync** всё пойдёт путём.

Программные шаги запуска выдачи меток времени

Это включается установкой бита 0 регистра ETH__PTPTSCR. Но после этого надо инициализировать счётчик меток времени, вот тогда и запустится. Последовательность такая:

1. Маскируем прерывание запуска меток времени битом 9 в регистре MACIMR.
2. Битом 0 в регистре меток разрешаем их.
3. Пишем регистр инкремента суб-секунд на основе частоты тактов PTP.
4. Если используем тонкую коррекцию, то пишем регистр слагаемого и ставим бит 5 регистра управления метками 5 (обновление регистра слагаемого).
5. Ждём его снятия (опросом).
6. Режим тонкой коррекции можно включить битом 1 в регистре управления метками.
7. В старший и младший регистры обновления меток времени пишем нужное значение времени.
8. Ставим бит 2 регистра управления (инициируем метки времени).
9. Счётчик меток времени стартует после записи в регистр обновления меток времени.
10. Включаем передатчик и приёмник MAC.

NB: Если метки времени отключаются снятием бита 0 регистра ETH__PTPTSCR, то для нового запуска надо повторить все упомянутые шаги.

Программные шаги Грубой коррекции системного времени

Это:

1. В старший и младший регистры обновления меток пишем смещение (плюс или минус).
2. Ставим бит TSSTU в регистре управления метками.
3. Обновление регистров отмечается снятием этого бита.

Программные шаги Тонкой коррекции системного времени

Это:

1. По упомянутому выше алгоритму вычисляем скорость, с которой надо увеличивать или уменьшать инкремент системного времени.
2. Обновляем метки времени.
3. Ждём достижения системным временем нужного значения, можно по прерыванию запуска меток времени.
4. В старший и младший регистры целевого времени пишем нужное значение. Снимаем маску прерываний меток времени очистки бита 9 в регистре ETH_MACIMR.
5. Ставим бит TSARU в регистре управления метками времени.
6. По этому прерыванию читаем регистр ETH_MACSR.
7. Переписываем в регистр слагаемого старое значение и снова ставим бит 5 в ETH_TPTSCR.

Запуск TIM2 от сигнала PTP

Когда системное время достигает целевого значения MAC выдаёт прерывание запуска, что вносит известную задержку и неопределённость в время выполнения команд.

Дабы избежать неопределённости, вместо прерывания используют выходной сигнал запуска от PTP. Он внутренне подключён к входу запуска TIM2 со всеми его возможностями. Так его синхронизируют с системным временем PTP. Неопределённость исчезает, поскольку такты таймера (PCLK1: TIM2 APB1) и PTP (HCLK) синхронизированы.

Сигнал подключается битом 29 регистра AFIO_MAPR.

Секундный сигнал от PTP

Бит 30 регистра AFIO_MAPR разрешает выдачу посекундного сигнала (PPS) длительностью 125ms на внешний вывод. Он поможет с помощью осциллоскопа проверить и уточнить синхронизацию узлов всей сети.

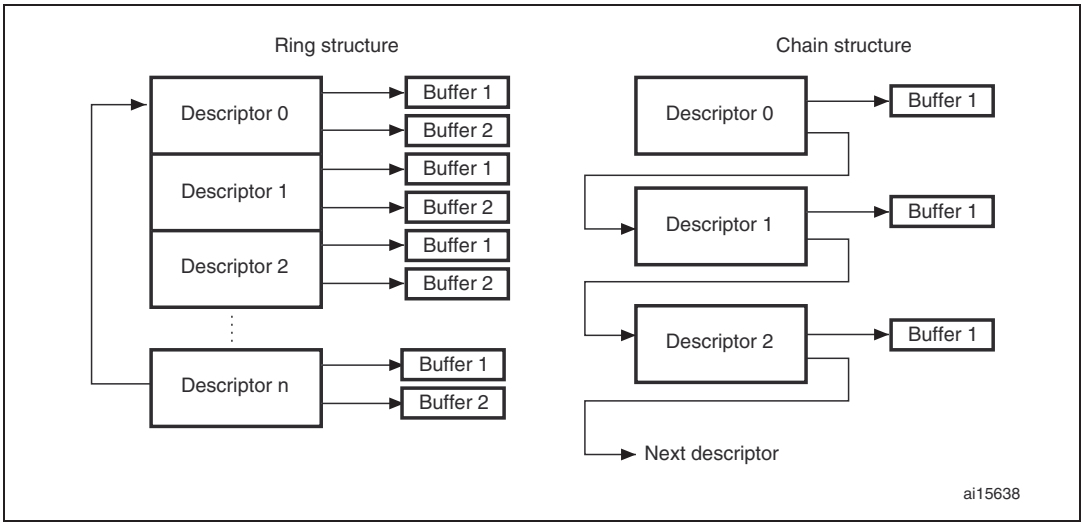
33.6. Функциональное описание Ethernet: Контроллер DMA

DMA имеет независимые машины приёма и передачи и пространство CSR. Для обмена с памятью используются TxFIFO и RxFIFO. Для передачи пакетов данных используются дескрипторы. Активно используются прерывания. Для работы DMA использует через две структуры данных:

- Регистры управления и состояния (CSR)
- Списки дескрипторов и буферы данных.

Дескрипторы лежат в памяти и работают как указатели буферов. Есть два списка дескрипторов: приёма и передачи. Базовые адреса списков пишутся в 3й и 4й регистры DMA. Список дескрипторов связанный (явно или неявно). Последний дескриптор может указывать на первый, образуя кольцо. Явная сцепка дескрипторов выполняется во втором адресе приёмных и передающих дескрипторов (RDES1[14] и TDES0[20]). Список дескрипторов лежит в физической памяти хоста. Один дескриптор может указывать максимум на два буфера. Это позволяет использовать не соприкасающиеся буферы. Буфер данных лежит в физической памяти и содержит часть фрейма, один целый фрейм, но не более. Буферы содержат только данные. Статус буфера отображается в дескрипторе. Цепочки данных относится к фреймам, занимающим несколько буферов, но один дескриптор не может охватывать несколько фреймов. По концу фрейма DMA переходит к следующему буферу. Цепочки данных можно разрешать и запрещать.

Кольцо и цепь дескрипторов



33.6.1. Инициализация передачи с DMA

Вот так:

1. В регистре ETH_DMABMR ставим параметры доступа к шине STM32F107xx.
2. В регистре ETH_DMAIER маскируем ненужные прерывания.
3. Программа создаёт списки дескрипторов прима и передачи и пишет их адреса начала в регистры ETH_DMARDLAR and ETH_DMATDLAR.
4. В регистрах MAC номер 1, 2 и 3 задаём параметры фильтрации.
5. В регистре ETH_MACCR задаём и включаем режимы передачи и приёма. Биты PS и DM ставятся на основе результатов переговоров (чтение из PHY).
6. В регистре ETH_DMAOMR битами 13 и 1 стартуем передачу и приём.
7. Машины передачи и приёма достают дескрипторы из списков и начинают работать независимо друг от друга.

33.6.2. Пакетный доступ хоста к шине

Если бит FB в регистре ETH_DMABMR стоит, то DMA обменивается с памятью пакетами фиксированной длины. Длина пакета ограничена значением поля PBL в регистре ETH_DMABMR. К дескрипторам всегда обращаются максимально возможными пакетами по 16 байт или PBL.

Tx DMA начинает передачу только если в TxFIFO хватает места для всего пакета или остатка для конца фрейма. Если интерфейс AHB поддерживает пакеты фиксированной длины, то выбирается лучшая комбинация передач INCR4, INCR8, INCR16 и SINGLE, иначе используются INCR (неопределённой длины) SINGLE.

Rx DMA начинает передачу только если в RxFIFO есть данные нужной длины или появился конец фрейма. Если интерфейс AHB поддерживает пакеты фиксированной длины, то выбирается лучшая комбинация передач INCR4, INCR8, INCR16 и SINGLE. Передачи, короче фиксированного пакета, дополняются пустыми передачами до нужной длины.

Если бит FB регистра ETH_DMABMR снят, то используются передачи INCR и SINGLE.

Если интерфейс AHB поставлен на передачи с выравненным адресом, то обе машины DMA смотрят, чтобы первая передача была равна или меньше значения PBL. Так все последующие передачи выравниваются на границу PBL. При PBL > 16 DMA выравнивает адреса на границу 16, поскольку AHB не поддерживает больше чем INCR16.

33.6.3. Выравнивание буфера данных хоста

Адрес начала буферов может начинаться с любого байта, но DMA передаёт данные в соответствии с шириной шины, заполняя ненужные линии пустыми данными. Обычно так бывает в начале или конце фрейма Ethernet.

- Пример чтения буфера:
  Адрес передающего буфера равен 0x0000 0FF2, передаётся 15 байт. DMA прочитает 5 слов с адреса 0x0000 0FF0, но в TxFIFO первые 2 байта и последние 3 не попадут. Кроме конца фрейма DMA всегда пишет в TxFIFO словами.
- Пример записи в буфер:
  Адрес приёмного буфера равен 0x0000 0FF2, передаётся 16 байт. DMA запишет пять 32-бит слов данных по адресу 0x0000 0FF0. Первые 2 байта первой передачи и последние 2 байта третьей передачи будут пустыми.

33.6.4. Вычисление размера буфера

В дескрипторах передающий DMA изменяет только статус (xDES0), размерами занимается программа. DMA передаёт ядру MAC точно указанное в поле размера буфера TDES1 число байт. Если дескриптор отмечен как первый (стоит бит FS в TDES0), то DMA отмечает первую передачу из буфера как начало фрейма. Если дескриптор отмечен как последний (бит LS в TDES0), то DMA отмечает последнюю передачу из буфера как конец фрейма.

Приёмный DMA передаёт данные в буфер вплоть до его заполнения или конца фрейма. Если дескриптор не отмечен как последний (бит LS в RDES0), то его буфер(ы) полный и число данных равно размеру буфера, и минус смещение указателя буфера при стоящем бите FS. Если указатель буфера выравнен на ширину шины, то смещение равно нулю. Если дескриптор отмечен как последний, то буфер может быть не полным. Число данных в этом буфере определяется вычитанием из длины фрейма (биты FL в RDES0[29:16]) суммы длин предыдущих буферов. Приёмный DMA всегда начинает передачу нового фрейма с новым дескриптором.

NB: Адрес начала приёмного буфера надо всегда выравнивать на ширину системной шины чтобы избежать выхода за его границы из-за требований DMA.

33.6.5. Арбитр DMA

Есть два типа арбитража доступа приёмного и передающего каналов DMA к шине AHB: круговой и приоритетный. При круговом (снят бит DA в ETH_DMABMR) выделяет шину в зависимости от бита PM в ETH_DMABMR. Если бит DA стоит, то приёмный DMA всегда приоритетнее передающего.

33.6.6. Извещение DMA об ошибке

Если при передаче DMA ведомый сообщил об ошибке, то DMA останавливает все передачи и ставит биты ошибки и фатальной ошибки шины в регистре состояния (ETH_DMASR). Возобновить работу можно только после программного или аппаратного сброса и инициализации DMA.

33.6.7. Конфигурация TxDMA

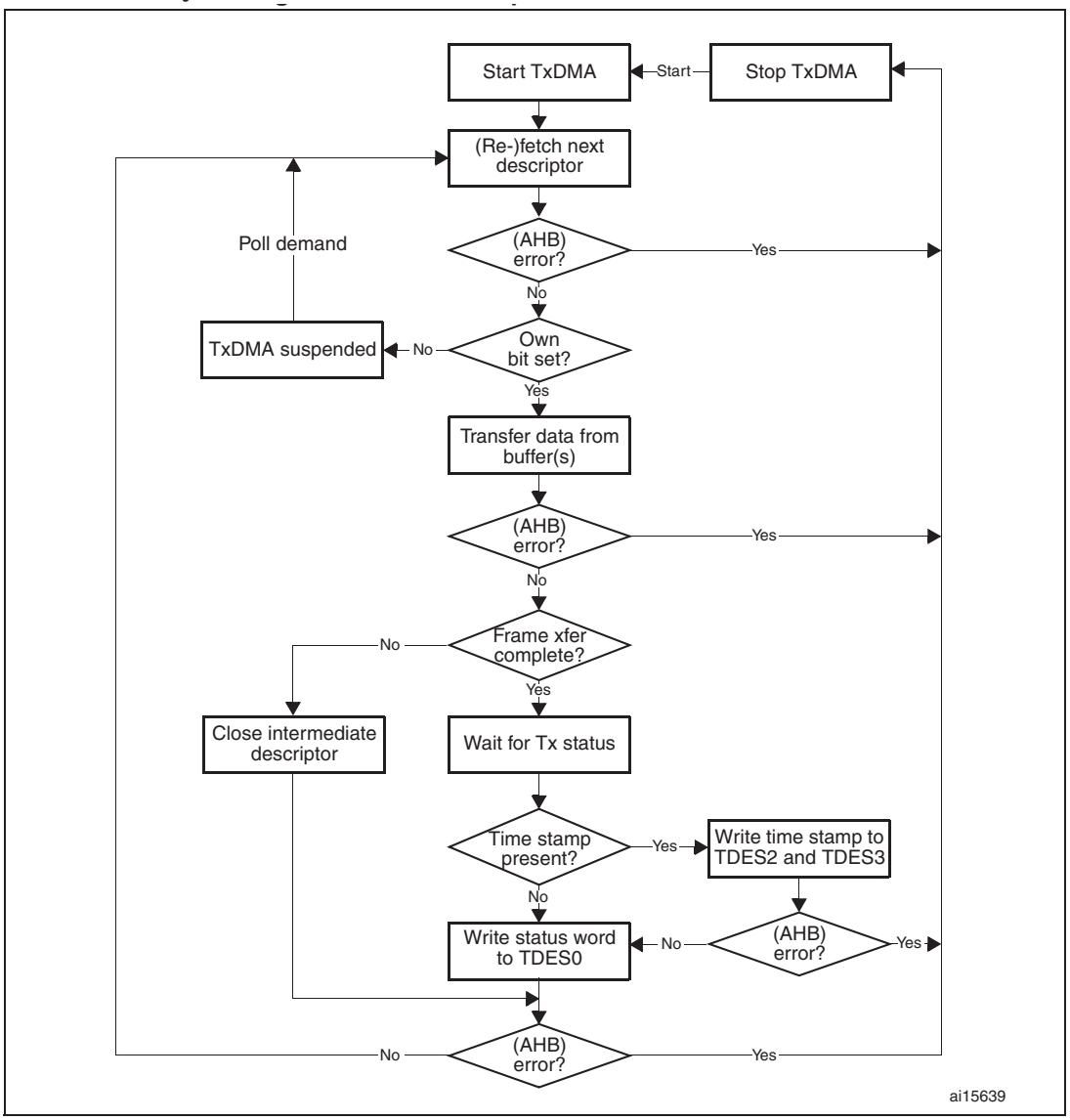
Работа TxDMA: режим по умолчанию (не-OSF)

1. Пользователь оформляет дескриптор (TDES0-TDES3) и его буфер(ы) с передаваемым фреймом и ставит бит OWN (TDES0[31]).
2. По установке бита ST (ETH_DMAOMR[13]) DMA идёт в режим Run.
3. В том режиме DMA ищет в списке дескрипторов передаваемые фреймы. При появлении ошибки или дескриптора, принадлежащего CPU, передача останавливается и ставятся биты

недоступности буфера (ETH_DMASR[2]) и Нормального прерывания (ETH_DMASR[16]). Машина переходит к шагу 9.

4. Из принадлежащего DMA дескриптора (стоит TDES0[31]) извлекается адрес буфера данных.
5. DMA передаёт данные из памяти.
6. Если фрейм лежит в буферах нескольких дескрипторов, то DMA закрывает промежуточный дескриптор и выбирает следующий. Шаги 3, 4 и 5 повторяются до конца фрейма.
7. По концу передачи фрейма в TDES0 пишется статус передачи, а в TDES2 и TDES3 может быть записана метка времени. Бит OWN снимается и дескриптор передаётся CPU.
8. По концу передачи фрейма со стоящем бите Прерывания по концу передачи (TDES1[31]) в последнем дескрипторе запрашивается прерывание Передачи (ETH_DMASR [0]). DMA возвращается к шагу 3.
9. В режиме Останова DMA, после получения запроса на просмотр передачи очереди пытается получить дескриптор передачи и вернуться к шагу 3. Прерывание исчерпания очищается.

Режим по умолчанию TxDMA



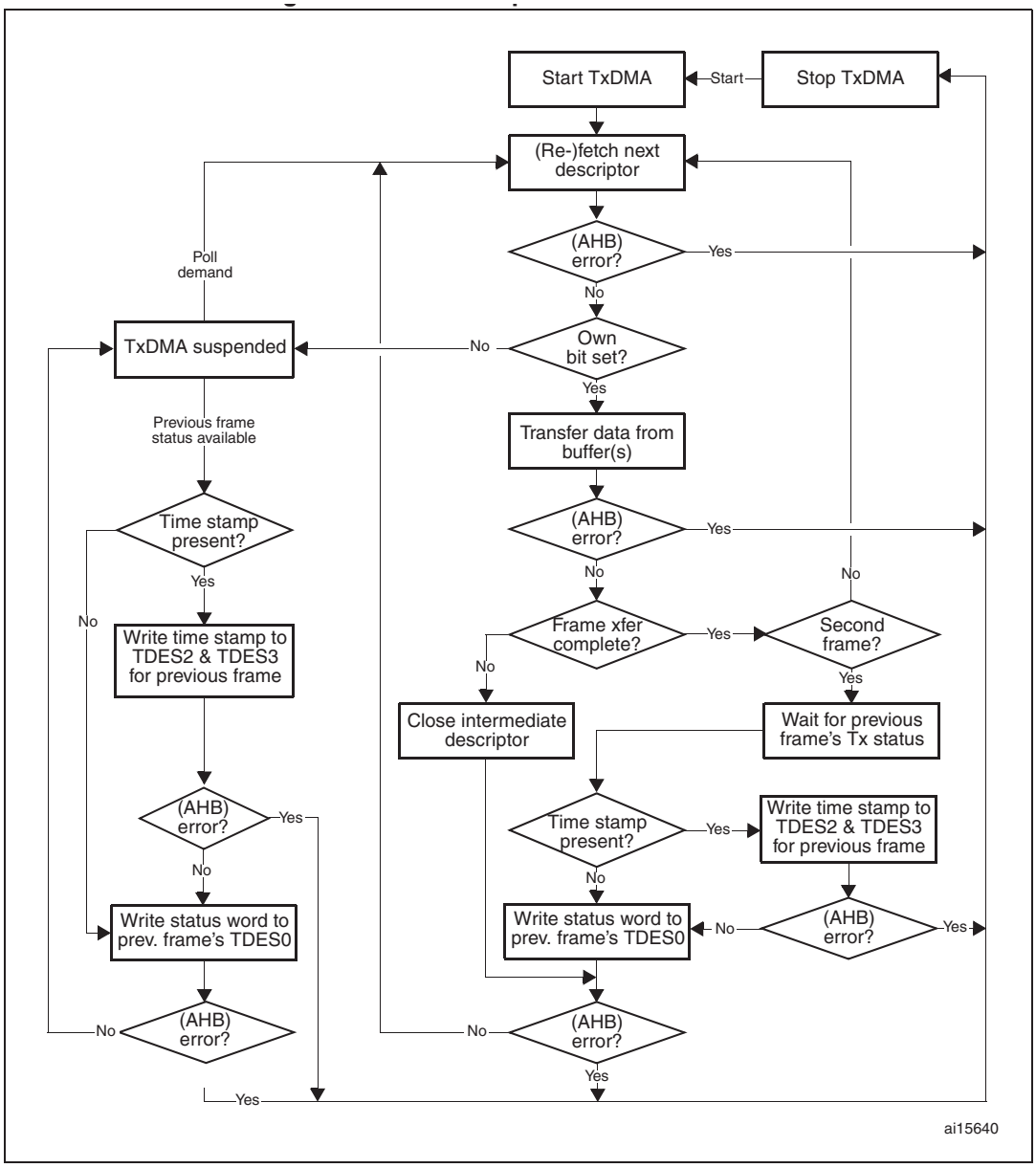
Работа TxDMA: режим OSF

При стоящем бите OSF в регистре ETH_DMAOMR[2] DMA может одновременно иметь два фрейма без закрытия дескриптора статуса первого. После завершения передачи первого фрейма DMA ищет в списке дескрипторов второй. Если он нормальный, то передаётся до записи статуса первого. Последовательность такая:

1. DMA выполняет шаги 1–6 режима по умолчанию TxDMA.
2. Затем DMA извлекает следующий дескриптор без закрытия первого.
3. Если DMA владеет этим дескриптором, то он извлекает адрес буфера, иначе идет на шаг 7.

4. DMA передаёт фрейм из памяти вплоть до его конца, закрывая промежуточные дескрипторы, если фрейм занимает несколько дескрипторов.
5. DMA ждёт статуса передачи и метку времени предыдущего фрейма, если надо, пишет метку времени в TDES2 и TDES3, статус передачи и снимает бит OWN в TDES0, закрывая дескриптор.
6. DMA ставит прерывание Передачи (если можно), извлекает следующий дескриптор и, если предыдущий статус нормальный, идёт к шагу 3. Иначе DMA идёт в Останов (шаг 7).
7. Если в режиме Останова DMA принимает запоздавшие статус и метку времени, то он пишет метку в TDES2 и TDES3, статус в соответствующий TDES0, ставит соответствующие прерывания и возвращается в Останов.
8. DMA выходит из Останова в Run (шаг 1 или 2 в зависимость от хранимого статуса) только по требованию Опроса передачи (регистр ETH_DMATPDR).

Режим OSF в TxDMA



Обработка фрейма передачи

Данные в буфере для DMA не должны содержать преамбулу, поля DA, SA, и Тип/Длина включаются всегда, а если выключено автоматическое добавление CRC и расширителя, то и эти поля и FCS. Фреймы могут охватывать цепочку буферов. Разделяются фреймы первым (TDES0[28]) и последним дескрипторами (TDES0[29]). Если установлены оба, то это единственный дескриптор. У промежуточных дескрипторов очищены и TDES0[28] и TDES0[29]. У последнего стоит TDES0[29]. После передачи последнего буфера, DMA возвращает в слово статуса дескриптора 0 (TDES0) статус из дескриптора последнего сегмента передачи. Если стоит Прерывание по завершению (TDES0[30]), то ставится Прерывание передачи (ETH_DMASR [0]), выбирается следующий дескриптор и процесс повторяется. Реальная передача начинается после заполнения TxFIFO до заданного порога

(ETH_DMAOMR[16:14]) или полном фрейме в FIFO. Ещё есть опция режима Накопления (Store-and-Forward, ETH_DMAOMR[21]). По завершению передачи DMA дескриптор освобождается (снимается бит OWN TDES0[31]).

Остановка опроса передачи

Это происходит когда:

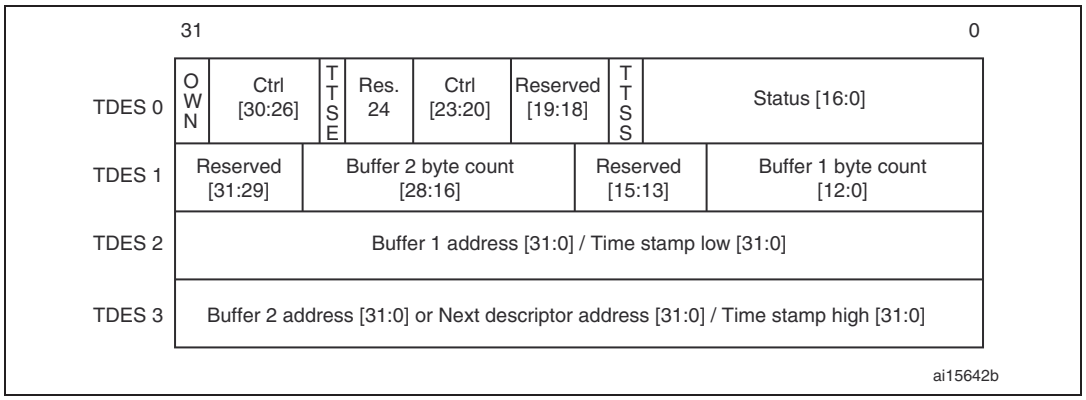
- DMA обнаруживает дескриптор, принадлежащий CPU (TDES0[31]=0), ставится флаг недоступности буфера (ETH_DMASR[2]). Программа должна вернуть дескриптор в собственность DMA и выдать требование Опроса.
- Передача прекращена по исчерпанию буфера, ставится нужный бит в TDES0. Ставятся биты Ненормального прерывания (ETH_DMASR [15]) и Исчерпания (ETH_DMASR[5]), в дескриптор 0 пишется причина прерывания.

Если DMA уходит в Останов по первому условию, то ставятся биты Нормального прерывания (ETH_DMASR [16]) и Недоступного буфера (ETH_DMASR[2]). В обоих случаях позиция в списке передачи сохраняется на следующем за последним дескриптором, закрытым DMA. После снятия причины останова надо программно выдать требование Опроса передачи.

Дескрипторы TxDMA

Это четыре 32-бит слова.

Дескриптор передачи



- **TDES0: Дескриптор передачи Слово 0**

При инициализации дескриптора надо писать управляющие биты [30:26]+[23:20] и бит OWN [31]. Возвращая дескриптор, DMA снимает все управляющие биты и бит OWN и пишет только статус.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OWN	IC	LS	FS	DC	DP	TTSE	Res	CIC		TER	TCH	Res.		TTSS	IHE
rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw			rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ES	JT	FF	IPE	LCA	NC	LCO	EC	VF	CC				ED	UF	DB
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Бит 31 **OWN:** Бит владения

1 - дескриптором владеет DMA, 0 - CPU. DMA снимает этот бит по завершению передачи или по полному прочтению буфера. Бит владения первого дескриптора фрейма надо ставить последним.

— Бит 30 **IC:** Прерывание по завершению

Если стоит, то по завершению передачи фрейма ставится бит прерывания передачи (Регистр 5[0]).

— Бит 29 **LS:** Последний сегмент фрейма

— Бит 28 **FS:** Первый сегмент фрейма

— Бит 27 **DC:** Запрет добавления CRC в конце фрейма

Действует только для первого сегмента фрейма (TDES0[28]=1).

— Бит 26 **DP:** Запрет добавления расширения в фрейм, короче 64 байт

Если бит снят, то DMA автоматически добавляет расширитель и CRC в конец фрейма-коротыша, независимо от состояния бита DC (TDES0[27])bit. Действует только для первого сегмента фрейма (TDES0[28]=1).

— Бит 25 **TTSE:** Разрешение меток времени передачи

Если стоит одновременно с битом TSE(ETH_PTPTSCR бит 0), то в дескриптор передачи пишется метка времени. Действует только для первого сегмента фрейма (TDES0[28]=1).

— Бит 24 Резерв, не трогать.

— Биты 23:22 **CIC:** Управление вставкой контрольной суммы

Аппаратное вычисление и вставка CRC:

00: Запрещено

01: Можно только в заголовках IP

10: Можно в заголовках IP и данных, кроме псевдо-заголовков

11: Можно в заголовках IP, данных, и псевдо-заголовках.

— Бит 21 **TER:** Конец кольца передачи

Указывает, что достигнут конец списка дескрипторов. DMA возвращает адрес начала списка, создавая кольцо.

— Бит 20 **TCH:** Второй адрес в цепочке

Указывает, что второй адрес это адрес следующего дескриптора, а не второго буфера. При этом значение TBS2 (TDES1[28:16]) безразлично. Бит TDES0[21] старше,чем TDES0[20].

— Биты 19:18 Резерв, не трогать.

— Бит 17 **TTSS:** Метка времени передачи фрейма захвачена

Если стоит, то в TDES2 и TDES3 содержат метку времени. Действует только для последнего сегмента фрейма (TDES0[29]=1).

— Бит 16 **IHE:** Ошибка заголовка IP

Длина в заголовке пакета IPv4 не совпала с числом принятых от программы байтов. В фреймах IPv6 длина главного заголовка не равна 40 байтов. Поле длины/типа фреймов IPv4 и IPv6 должно соответствовать версии заголовка IP пакета. У фреймов IPv4 поле длины заголовка меньше 0x5.

— Бит 15 **ES:** Индикатор ошибок

Это логическое OR битов:

TDES0[14]: Таймаут трёпа
TDES0[13]: Слив фрейма
TDES0[11]: Потеря несущей
TDES0[10]: Нет несущей
TDES0[9]: Поздняя коллизия
TDES0[8]: Излишек коллизий
TDES0[2]: Излишняя отсрочка
TDES0[1]: Исчерпание
TDES0[16]: Ошибка заголовка IP
TDES0[12]: Ошибка данных IP

— Бит 14 **JT:** Таймаут трёпа передатчика MAC

Работает при снятом бите JD регистра конфигурации MAC.

— Бит 13 **FF:** Фрейм слит DMA/MTL по команде от CPU

— Бит 12 **IPE:** Ошибка данных IP

Длина данных в заголовке IPv4 или IPv6 не совпала в с длиной пакета TCP, UDP или ICMP в дейтаграмме IP.

— Бит 11 **LCA:** Потеря несущей при передаче фрейма

Сигнал MII_CRS был неактивен в течении одного или более тактов передачи. Действителен только при передаче фрейма без коллизий в полу-дуплексе.

— Бит 10 **NC:** No carrier

При передаче сигнала Carrier Sense (Есть несущая) от PHY не было.

— Бит 9 **LCO:** Поздняя коллизия

Коллизия появилась позже окна коллизий (время 64 байтов, включая преамбулу, в режиме MII). Недействителен при стоящем бите Исчерпания.

— Бит 8 **EC:** Излишек коллизий

Передача прекращена после 16 коллизий при попытках передать текущий фрейм. Если бит RD (Запрет повторных попыток) в регистре конфигурации MAC стоит, то этот бит ставится после первой же коллизии передачи.

— Бит 7 **VF:** Передан фрейм VLAN

— Биты 6:3 **CC:** Счётчик коллизий передачи

При стоящем бите TDES0[8] это брехня.

— Бит 2 **ED:** Излишняя отсрочка

Передача прекращена из-за задержки больше 24 288 времён бита. Проверка включается битом DC в регистре управления MAC.

— Бит 1 **UF:** Исчерпание передающего буфера

Данные из памяти не успели поступить в буфер и он опустел во время передачи фрейма. Машина уходит в Останов, одновременно ставятся Исчерпание (Регистр 5[5]) и прерывание Передачи (Регистр 5[0]).

— Бит 0 **DB:** Отсрочка

MAC отложил передачу из-за несущей. Работает только в полу-дуплексе.

- **TDES1: Дескриптор передачи Слово 1**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			TBS2													Reserved			TBS1												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:29 Резерв, не трогать.

— Биты 28:16 **TBS2:** Размер передающего буфера 2 в байтах

Недействителен при стоящем TDES0[20].

— Биты 15:13 Резерв, не трогать.

— Биты 12:0 **TBS1:** Размер передающего буфера 1 в байтах

Если 0, то DMA использует буфер 2 или следующий дескриптор в зависимости от TCH (TDES0[20]).

- **TDES2: Дескриптор передачи Слово 2**

Указатель адреса первого буфера данных дескриптора или данные метки времени.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBAP1/TBAP/TTSL																															
rw																															

— Биты 31:0 **TBAP1:** Указатель адреса буфера 1/Младшее слово метки времени

TBAP: В момент передачи дескриптора DMA (установка бита OWN в TDES0) это физический адрес буфера 1. Ограничений по выравниванию нет.

TTSL: Если разрешено битом TTSE в TDES0 последнего сегмента, то перед снятием бита OWN в TDES0, DMA пишет сюда младшие 32 бита метки времени.

- **TDES3: Дескриптор передачи Слово 3**

Указатель адреса данных второго буфера дескриптора или данные метки времени.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBAP2/TBAP2/TTSH																															
rw																															

— Биты 31:0 **TBAP2:** Указатель адреса буфера 2 (Адрес следующего дескриптора)/Старшее слово метки времени

TBAP2: Если используется кольцо дескрипторов, то в момент передачи дескриптора DMA (установка бита OWN в TDES0) это физический адрес буфера 2. Если второй буфер в цепочке (TDES1[20]=1), то это адрес следующего дескриптора, при этом указатель должен выравнен на ширину шины (Младшие биты игнорируются внутри.)

TTSH: Если разрешено битом TTSE в TDES0 последнего сегмента, то перед снятием бита OWN в TDES0, DMA пишет сюда старшие 32 бита метки времени.

33.6.8. Конфигурация Rx DMA

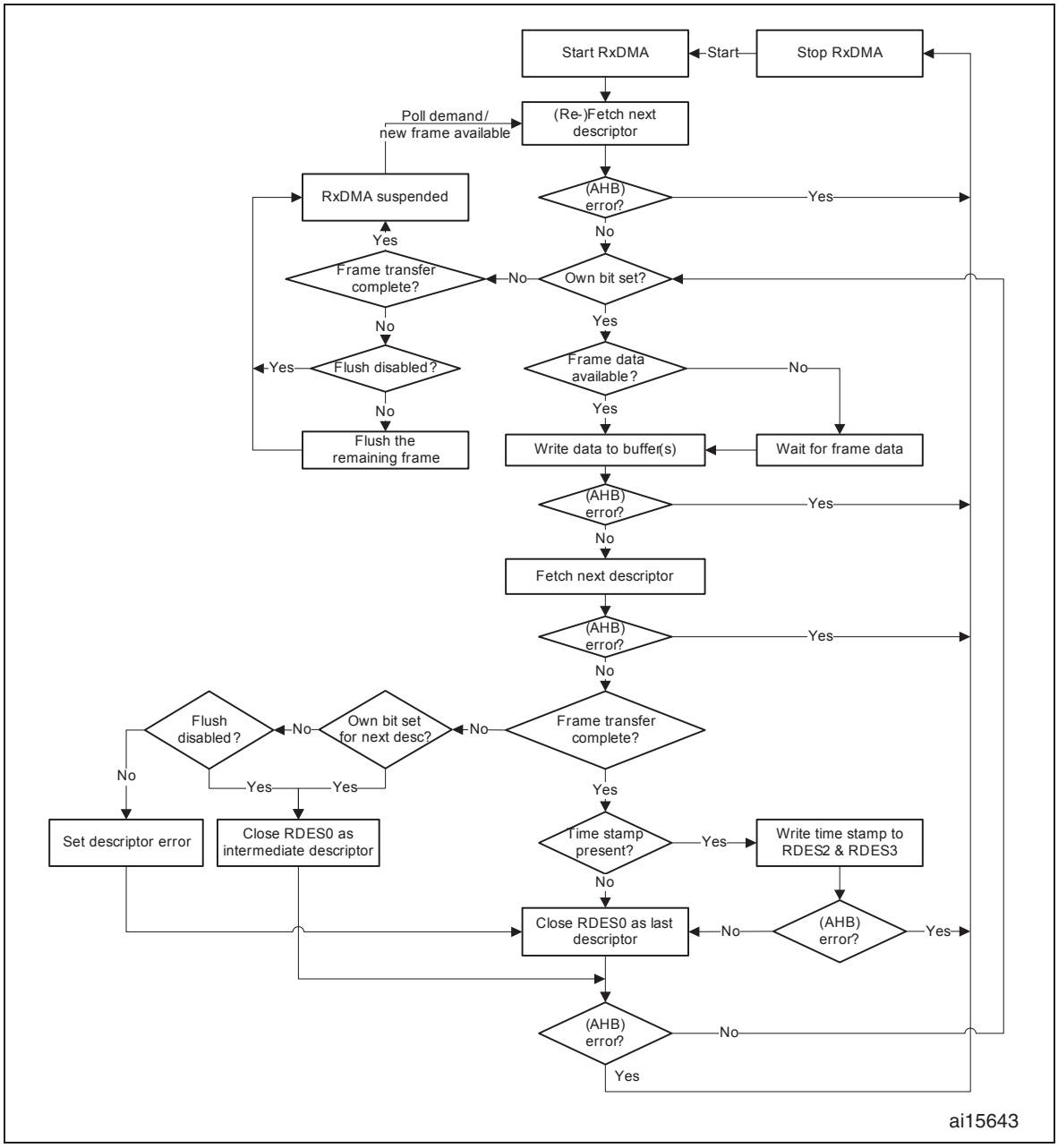
Последовательность такая:

1. CPU оформляет дескрипторы (RDES0-RDES3) и ставит бит OWN (RDES0[31]).
2. По установке бита SR (ETH_DMAOMR[1]) DMA идёт в режим Run. Теперь DMA опрашивает список дескрипторов в поисках свободного. Если извлечённый дескриптор принадлежит CPU, то DMA идёт в Останов и к шагу 9.
3. DMA достаёт из дескрипторов адреса буферов данных.
4. Входные фреймы размещаются в буферы данных дескриптора.
5. При заполнении буфера или при завершении фрейма машина достаёт следующий дескриптор.
6. Если передача текущего фрейма завершена, то DMA идёт к шагу 7. Если следующий дескриптор не принадлежит DMA, а фрейм ещё не завершён (не принят EOF), то DMA ставит бит ошибки дескриптора в RDES0 (если не выключен слив). DMA закрывает текущий дескриптор (снимает бит OWN) и делает его промежуточным, снимая бит LS в RDES1, и идёт на шаг 8. Если следующий дескриптор принадлежит DMA, но текущий фрейм ещё не закончился, то DMA закрывает текущий дескриптор как промежуточный и возвращается к шагу 4.

7. Если метки времени разрешены, то DMA пишет её в RDES2 и RDES3 текущего дескриптора, затем пишет статус в RDES0, снимает бит OWN и ставит бит LS.
8. Машина проверяет бит OWN последнего дескриптора. Если он принадлежит DMA, то идёт на шаг 4 и ждёт следующего фрейма, иначе ставит бит недоступности буфера (ETH_DMASR[7]) и уходит в Останов (шаг 9).
9. Перед Остановом частичные фреймы сливаются из FIFO (бит 24 регистра ETH_DMAOMR).
10. DMA выходит из Останова по требованию опроса или появлению в FIFO начала следующего фрейма. Он идёт к шагу 2 и повторно извлекает следующий дескриптор.

DMA подтверждает приём статуса после записи метки времени и самого статуса в дескриптор. Если в CSR метки времени разрешены, а действительной метки фрейма нет (например, FIFO заполнился до её записи), то DMA пишет в RDES2 и RDES3 другие.

Работа DMA приёма



Получение дескриптора приёма

DMA пытается заранее раздобыть дополнительный дескриптор приёма когда:

- При входе и режим Run бит Start/Stop (ETH_DMAOMR[1]) стоит.
- Буфер данных текущего дескриптора полон, а фрейм ещё не закончился.
- Контроллер завершил приём фрейма, а текущий дескриптор ещё не закрыт.
- Буфер принадлежит CPU (RDES0[31] = 0) и принят новый фрейм.
- Получено требование опроса.

Приём фрейма

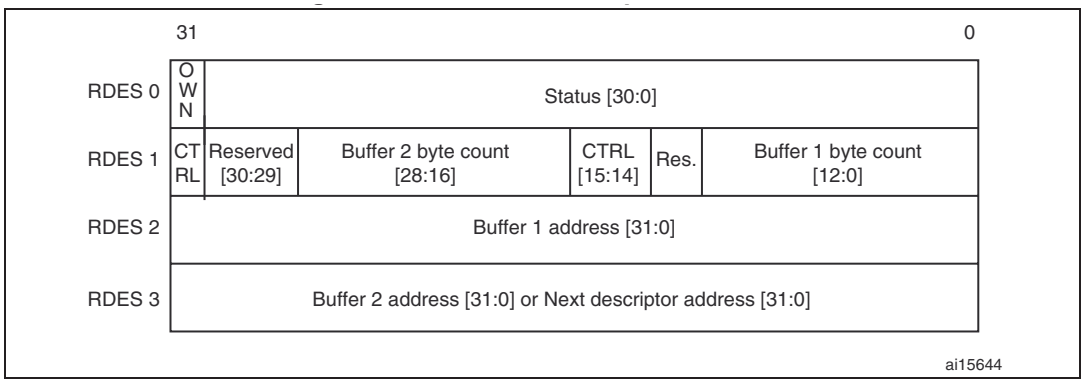
MAC передаёт фрейм в память только если он прошёл фильтр адреса и размером больше или равен установленному порогу RxFIFO, или в режиме Накопления в FIFO записан полный фрейм. Если бит Принять всё (ETH_MACFFR[31]) снят, то фреймы, не прошедшие фильтр адреса, выбрасываются. Фреймы, короче 64 байт из-за коллизии или преждевременного завершения, из RxFIFO можно вычищать. Передача данных фрейма в буфер текущего дескриптора начинается после достижения порога заполнения RxFIFO. Если DMA готов передавать данные в память, то для разделения фреймов ставится бит первого дескриптора (RDES0[9]). Дескрипторы освобождаются по снятию бита OWN (RDES0[31]), заполнению буфера данных или по записи последнего сегмента. Если фрейм лежит в одном дескрипторе, то стоят биты последнего (RDES0[8] и первого (RDES0[9]) дескриптора. DMA извлекает следующий дескриптор, в предыдущем дескрипторе фрейма ставит бит последнего дескриптора (RDES0[8]) и освобождает биты статуса RDES0. Наконец, DMA ставит бит прерывания приёма (ETH_DMASR [6]). Процесс повторяется до обнаружения дескриптора, принадлежащего CPU. При этом ставится бит недоступности буфера (ETH_DMASR[7]) и всё идёт в Останов. Позиция в списке приёма сохраняется.

Режим Останова приёма

Если в режиме Останова появляется новый фрейм, то DMA снова извлекает текущий дескриптор из памяти. Если он принадлежит DMA, то машина идёт в режим Run и начинает приём фрейма. Иначе DMA выбрасывает верхний фрейм из RxFIFO и наращивает счётчик выброшенных фреймов. Если в RxFIFO лежит более одного фрейма, то процесс повторяется. Выброс или слив верхнего фрейма из RxFIFO можно запретить установкой бита DFRF в регистре режима работы DMA. В таком случае ставится бит недоступности буфера и машина идёт в Останов.

Дескрипторы Rx DMA

Это структура из четырёх 32-бит слов.



- **RDES0: Дескриптор приёма Слово 0**

31	30	29:16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OWN	AFM	FL	ES	DE	SAF	LE	OE	VLAN	FS	LS	IPHCE	LCO	FT	RWT	RE	DE	CE	PCE
rw																		

— Бит 31 **OWN:** Бит принадлежности дескриптора
0 - CPU, 1- DMA. DMA снимает этот бит по завершению приёма фрейма или заполнению буфера.

— Бит 30 **AFM:** Фрейм не прошёл фильтр DA.

— Биты 29:16 **FL:** Длина фрейма в байтах в памяти (включая CRC).
Действителен в последнем дескрипторе (стоит RDES0[8]) при снятой ошибке **DE** (RDES0[14]).

— Бит 15 **ES:** Ошибка, логическое OR битов:
RDES0[1]: Ошибка CRC
RDES0[3]: Ошибка приёма
RDES0[4]: Таймаут сторожевого таймера
RDES0[6]: Поздняя коллизия
RDES0[7]: Фрейм-гигант (Не работает при ошибке контрольной суммы IPV4 (RDES0[7].)
RDES0[11]: Переполнение буфера
RDES0[14]: Ошибка последнего дескриптора (RDES0[8]=1).

— Бит 14 **DE:** Ошибка последнего дескриптора (RDES0[8]=1)
Не вместившийся в буфер фрейм усечён, следующий дескриптор не принадлежит DMA.

— Бит 13 **SAF:** Фрейм не прошёл фильтр SA.

— Бит 12 **LE:** Ошибка длины

Длина принятого фрейма не совпала с полем Типа/Длины, бит действителен при (RDES0[5]=0).

— Бит 11 **OE:** Переполнение входного буфера.

— Бит 10 **VLAN:** Тег фрейма VLAN.

— Бит 9 **FS:** Первый дескриптор

Если адрес первого буфера равен 0, то берётся адрес второго буфера. Если и он равен 0, то берётся следующий дескриптор.

— Бит 8 **LS:** Последний дескриптор фрейма.

— Бит 7 **IPHCE:** Ошибка контрольной суммы заголовка IPv4 или IPv6

Поле Типа не соответствует Версии заголовка IP, ошибка контрольной суммы заголовка IPv4, фрейм короче значения, указанного в заголовке IP.

— Бит 6 **LCO:** Позднее появление коллизии в полу-дуплексе.

— Бит 5 **FT:** Тип фрейма

0 - Фрейм IEEE802.3

1 - Фрейм Ethernet (поле LT больше или равно 0x0600).

Бит недействителен для фреймов короче 14 байт.

— Бит 4 **RWT:** Таймаут сторожевого таймера приёма, фрейм усечён.

— Бит 3 **RE:** Ошибка приёма, RX_ERR появился при стоящем RX_DV.

— Бит 2 **DE:** Ошибка дробности (Dribble)

Нечётное число полубайтов в режиме MII.

— Бит 1 **CE:** Ошибка CRC последнего дескриптора (RDES0[8]=1)

— Бит 0 **PCE**: Ошибка CRC данных

Вычисленная CRC пакетов TCP, UDP или ICMP не совпала с заявленной или длина принятых данных не совпала с заявленной в дейтаграммах IPv4 или IPv6 фрейма Ethernet.

Бит 5: FT	Бит 7: IPHCE	Бит 0: PCE	Статус фрейма
0	0	0	Фрейм IEEE 802.3 (Поле длины меньше 0x0600.)
1	0	0	Фрейм IPv4/IPv6, ошибки CRC нет
1	0	1	Фрейм IPv4/IPv6, ошибка CRC данных
1	1	0	Фрейм IPv4/IPv6, ошибка CRC заголовка IP
1	1	1	Фрейм IPv4/IPv6, ошибки CRC данных и заголовка IP
0	0	1	Фрейм IPv4/IPv6 ошибки CRC заголовка IP, данные не проверялись
0	1	1	Фрейм ни IPv4, ни IPv6 (CRC не проверялась.)
0	1	0	Резерв

- **RDES1: Дескриптор приёма Слово 1**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIC	RBS2		RBS2													RER	RCH	Reserved	RBS												
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	Reserved	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Бит 31 **DIC:** Запрет прерывания завершения по биту RS (CSR5[6]).

— Биты 30:29 Резерв, не трогать.

— Биты 28:16 **RBS2:** Размер приёмного буфера 2 в байтах при RDES1[14]=0

Должен быть кратен 4, 8 или 16, в зависимости от ширины шины (32, 64 или 128), даже при невыравненном адресе буфера. Иначе поведение непредсказуемо.

— Бит 15 **RER:** Конец кольца приёма

Последний дескриптор в списке, DMA возвращает адрес начала списка, создавая кольцо.

— Бит 14 **RCH:** Второй адрес в цепочке

Второй адрес в дескрипторе это адрес следующего дескриптора, а не второго буфера. Значение поля RBS2 (RDES1[28:16]) не волнует. RDES1[15] старше RDES1[14].

— Бит 13 Резерв, не трогать.

— Биты 12:0 **RBS1:** Размер приёмного буфера 1 в байтах

Должен быть кратен 4, 8 или 16, в зависимости от ширины шины (32, 64 или 128), даже при невыравненном адресе буфера. Иначе поведение непредсказуемо. Если равен 0, то DMA использует адрес буфера 2 или следующий дескриптор, в зависимости от бита RCH.

- **RDES2: Дескриптор приёма Слово 2**

Указатель адреса первого буфера данных дескриптора или данные метки времени.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RBP1 / RTSL																															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:0 **RBAP1 / RTSL:** Указатель адреса буфера 1/Младшее слово метки времени

RBAP1: В момент передачи дескриптора DMA (установка бита OWN в TDES0) это физический адрес буфера 1. Ограничений по выравниванию нет, за исключением передачи начала фрейма, когда DMA пишет в память с нулевыми битами RDES2[3/2/1:0], смещая реальные данные.

RTSL: Если это последний фрейм и метки времени разрешены, то перед снятием бита OWN в TDES0, DMA пишет сюда младшие 32 бита метки времени.

- **RDES3: Дескриптор приёма Слово 3**

Указатель адреса данных второго буфера дескриптора, адрес следующего дескриптора или данные метки времени.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RBP2 / RTSH																															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:0 **RBAP2/RTSH:** Указатель адреса буфера 2 (Адрес следующего дескриптора)/Старшее слово метки времени

TTSH: Если разрешено битом TTSE в TDES0 последнего сегмента, то перед снятием бита OWN в TDES0, DMA пишет сюда старшие 32 бита метки времени.

RBAP2: Если используется кольцо дескрипторов, то в момент передачи дескриптора DMA (установка бита OWN в RDES0) это физический адрес буфера 2.
Если бит цепочки (RDES1[24]) стоит, то это адрес следующего дескриптора. Он должен быть выравнен на границу ширины шины (RDES3[3,2,1:0] = 0, для ширины шины 128, 64 или 32. Младшие биты игнорируются внутри.)
Если бит цепочки (RDES1[24]) сброшен, то ограничений по выравниванию нет, за исключением передачи начала фрейма, когда DMA пишет в память с нулевыми битами RDES2[3/2/1:0], смещая реальные данные.

RTSH: Если это последний фрейм и метки времени разрешены, то перед снятием бита OWN в TDES0, DMA пишет сюда старшие 32 бита метки времени.

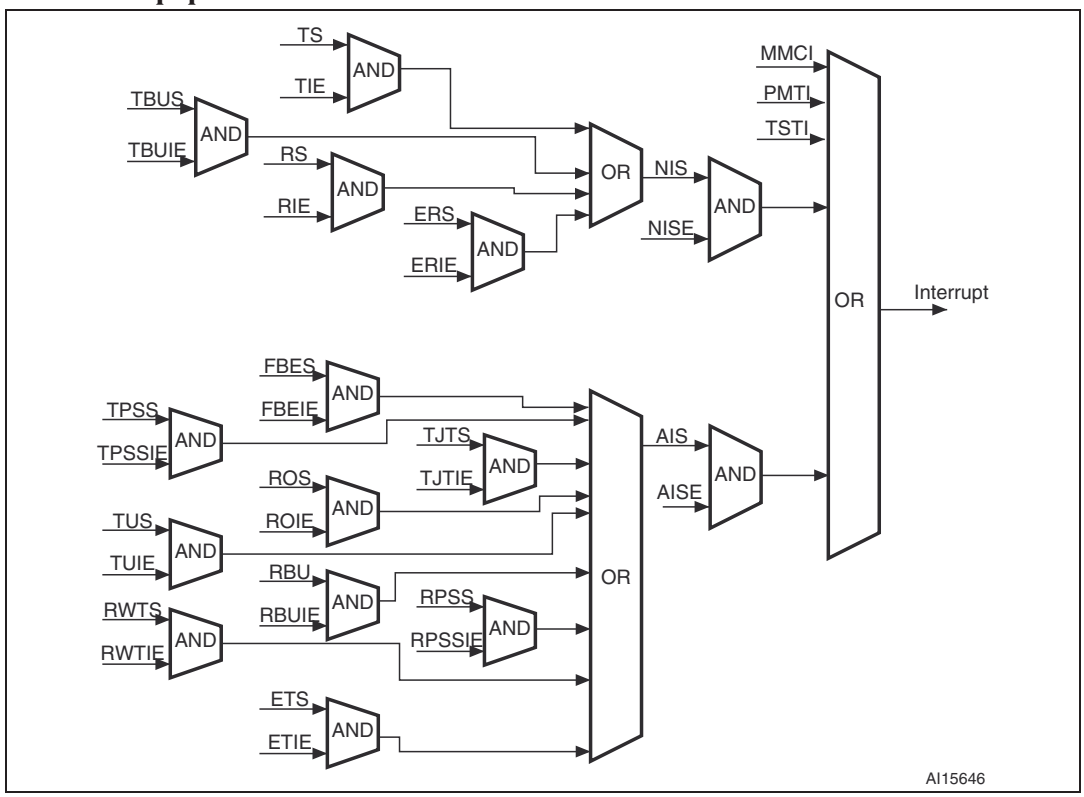
33.6.9. Прерывания DMA

Все источники прерываний отмечены своим битом регистра ETH_DMASR. Прерывания по ним разрешаются в регистре ETH_DMAIER.

Есть Нормальные и Ненормальные прерывания. Снимаются прерывания записью 1 в соответствующий бит. Общий бит прерывания снимается после очистки битов всех его источников прерываний. Ядро MAC вызывает прерывания только с битами TSTS или PMTS в ETH_DMASR.

Для всех одновременно стоящих запросов выдаётся только одно прерывание. Обработчик прерывания должен в регистре ETH_DMASR найти все причины прерывания и обслужить их. Если при выходе из разработчика сняты не все биты источников прерывания (или возникло новое), то вызывается новое прерывание.

Схема прерываний



33.7. Прерывания Ethernet

Контроллер Ethernet имеет два вектора прерываний: один для обычной работы, второй для события побудки по фрейму Побудки или Magic Packet, если оно подключено к EXTI line19.

Если разрешены и прерывание MAC PMT по событию побудки и прерывание по переднему фронту на EXTI Line19, то выдаются оба.

Если в сторожевой таймер (регистр `ETH_DMARSWTR`) что-то записано, то он активируется после завершения передачи фрейма в память без записи статуса (не разрешено в дескрипторе `RDES1`[31]). При достижении заданного значения ставится бит `RS` (`ETH_DMASR`) и, если разрешено битом `RIE` в регистре `ETH_DMAIER`, выдаётся прерывание. Если установка бита `RS` разрешена дескриптором, то по завершению приёма фрейма таймер отключается.

NB: Чтение регистра управления PMT автоматически снимает флаги приёма фрейма Побудки и Magic Packet. Но, поскольку регистры этих флагов лежат в домене CLK_RX, то может появиться задержка их появления (особо длинная в режиме 10 Mbit mode). Из-за этого при может возникнуть ложное прерывание даже после чтения PMT_CSR. Поэтому, обработчик должен выходить из прерывания только после снятия обоих битов приёма фрейма Побудки и Magic Packet.

33.8. Описание регистров Ethernet

Они доступны байтами (8-бит), полу-словами (16-бит) и словами (32-бита).

33.8.1. Описание регистров MAC

Регистр конфигурации Ethernet MAC (ETH_MACCR)

Смещение адреса: 0x0000

По сбросу: 0x0000 8000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								WD	JD	Reserved		IFG			CSD	Reserved	FES	ROD	LM	DM	IPCO	RD	Reserved	APCS	BL		DC	TE	RE	Reserved	
								rw	rw			rw			rw		rw	rw	rw	rw	rw	rw		rw	rw		rw	rw	rw		

— Биты 31:24 Резерв, не трогать.

— Бит 23 **WD:** Выключение сторожевого таймера приёма

0 - MAC принимает не более 2 048 байт, остальные усекаются.

1- Таймер выключен, MAC принимает фреймы до 16 384 байт.

— Бит 22 **JD:** Выключение сторожевого таймера трёпа передачи

0 - MAC передаёт не более 2 048 байт, остальные усекаются.

1- Таймер выключен, MAC передаёт фреймы до 16 384 байт.

— Биты 21:20 Резерв, не трогать.

— Биты 19:17 **IFG:** Промежуток между фреймами при передаче

000: Время 96 бит

001: Время 88 бит

010: Время 80 бит

....

111: Время 40 бит

NB: В полудуплексе IFG может быть равным только 0b100. Меньшие значения не рассматриваются.

— Бит 16 **CSD:** Выключение датчика несущей в полудуплексе

1 - Передатчик MAC игнорирует сигнал MII CRS. Ошибки потери и отсутствия несущей не выдаются.

0 - Эти ошибки выдаются и передача прекращается.

— Бит 15 Резерв, не трогать.

— Бит 14 **FES:** Скорость Fast Ethernet (MII)

0: 10 Mbit/s

1: 100 Mbit/s

— Бит 13 **ROD:** Выключение приёма своего в полудуплексе

0 - MAC принимает все фреймы, появляющиеся на PHY при передаче.

1 - MAC выключает приём фреймов.

Этот бит в дуплексе не работает.

— Бит 12 **LM:** Режим перемычки на MII

Нужны входные такты MII (RX_CLK), поскольку такты передачи внутри не перемыкаются.

— Бит 11 **DM:** Режим дуплекса

— Бит 10 **IPCO:** Проверка контрольной суммы IPv4

1 - проверяется CRC заголовков TCP/UDP/ICMP в данных пакетов IPv4.

0 - CRC заголовков TCP/UDP/ICMP не проверяется, биты статуса PCE и IPHCE всегда чистые.

— Бит 9 **RD:** Запрет повторной передачи в полу-дуплексе

1 - Только 1 попытка передачи. При появлении коллизии на MII, MAC прекращает передачу и пишет в статус ошибку Излишка коллизий.

0 - Число попыток передачи определено в BL.

— Бит 8 Резерв, не трогать.

— Бит 7 **APCS:** Автоматическое удаление Pad/CRC

1 - MAC удаляет поля Pad/FCS из принятых фреймов короче 1 501 байта. Фреймы длиннее 1 500 байт передаются в память как есть.

0 - Все принимаемые фреймы передаются в память как есть.

— Биты 6:5 **BL:** Предел задержки ответа в полу-дуплексе

Это целое число слотов (r) задержки (время 4 096 битов для 1000 Mbit/s и 512 битов для 10/100 Mbit/s), которые MAC ждёт до начала повторной передачи после коллизии.

00: k = min (*n*, 10)

01: k = min (*n*, 8)

10: k = min (*n*, 4)

11: k = min (*n*, 1),

где *n* = попытка передачи, *r* - случайное число в диапазоне $0 \le r < 2^k$

— Бит 4 **DC:** Проверка отсрочки в полу-дуплексе

При отсрочке больше 24 288 времён бита в режиме 10/100- Mbit/s, MAC выдаёт статус фрейма со стоящим битом Излишней задержки. Задержка начинается с момента готовности передачи при активном сигнале CRS (датчик несущей) на MII. Время отсрочки не накапливается. Если передатчик задерживается на 10 000 времён бита, то он передаёт, конфликтует, и после завершения ответа сбрасывает таймер отсрочки и начинает снова. При снятом бите MAC ждёт снятия сигнала CRS.

— Бит 3 **TE:** Включение передатчика

Снимается после завершения передачи текущего фрейма на MII.

— Бит 2 **RE:** Включение приёмника

Снимается после завершения приёма текущего фрейма из MII.

— Биты 1:0 Резерв, не трогать.

Регистр фильтра фрейма Ethernet MAC (ETH_MACFFR)

Смещение адреса: 0x0004

По сбросу: 0x0000 0000

Определяет первый уровень фильтрации адресов принятых фреймов.

31	30:11	10	9	8	7:6	5	4	3	2	1	0
RA	Reserved	HPF	SAF	SAIF	PCF	BFD	PAM	DAIF	HM	HU	PM
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Бит 31 **RA:** Принимать всё

1 - MAC принимает все фреймы, но результат фильтрации SA/DA отражается в слове статуса.

0 - MAC пропускает только фреймы, прошедшие фильтры SA/DA.

— Биты 30:11 Резерв, не трогать.

— Бит 10 **HPF:** Фильтрация хэш или полного адреса

1 - Если стоит бит HM или HU, то выполняется либо полная фильтрация адреса, либо хэш.

0 - Если стоит бит HM или HU, то фильтруется по хэш-таблице.

— Бит 9 **SAF:** Фильтрация адреса источника

Поле SA фрейма сравнивается с разрешёнными регистрами SA. При совпадении в слове статуса ставится бит SA Match.

1 - Отказ зависит от бита SAIF, фреймы, получившие отказ, выбрасываются.

0 - MAC передаёт принятый фрейм программе, в статусе передаётся бит SA Match.

— Бит 8 **SAIF:** Инверсия фильтра адреса источника

1 - Отказ получают фреймы, чей адрес SA есть в регистрах SA.

0 - Отказ получают фреймы, чьего адреса SA нет в регистрах SA.

— Биты 7:6 **PCF:** Пропускать управляющие фреймы, включая PAUSE

Обработка фреймов PAUSE зависит только от RFCE в регистре управления потоком[2].

00 или 01: MAC отбрасывает все управляющие фреймы

10: MAC пропускает все управляющие фреймы, независимо от фильтрации

11: MAC пропускает управляющие фреймы, прошедшие фильтр адреса.

— Бит 5 **BFD:** Запрет широковещательных фреймов

1 - Широковещательные фреймы получают отказ.

0 - Широковещательные фреймы пропускаются.

— Бит 4 **PAM:** Пропускать все многоадресные фреймы

1 - Пропускаются все многоадресные (первый бит в DA стоит) фреймы.

0 - Фильтрация зависит от бита HM.

— Бит 3 **DAIF:** Инверсная фильтрация DA одноадресных и многоадресных фреймов

0 - нормальная фильтрация DA адресов.

1 - Инверсная.

— Бит 2 **HM:** Многоадресная хэш-фильтрация

1 - Фильтрация многоадресных фреймов по хэш-таблице.

0 - Фильтрация многоадресных фреймов по полному адресу DA

— Бит 1 **HU:** Одноадресная хэш-фильтрация

1 - Фильтрация одноадресных фреймов по хэш-таблице.

0 - Фильтрация одноадресных фреймов по полному адресу DA

— Бит 0 **PM:** Беспорядочный режим

Если стоит, то пропускаются все фреймы, независимо от фильтрации, биты статуса SA/DA нулевые.

Старший регистр хэш-таблицы Ethernet MAC (ETH_MACHTHR)

Смещение адреса: 0x0008

По сбросу: 0x0000 0000

Для фильтрации групповых адресов используется 64-хэш-таблица в двух регистрах. Индексом к ней служат 6 старших битов CRC полученного фрейма. Старший бит это регистр, а остальные 5 это номер бита в регистре. Хэш 0b0 0000 определяет бит 0 регистра 0, а хэш 0b1 1111 бит 31 регистра 1.

Например, DA фрейма равен 0x1F52 419C B6AF, значение хэш равно 0x2C. Тогда проверяется бит [12] регистра `HTH`, при хэш, равном 0x07 проверяется бит [7] регистра `HTL`.

Фрейм принимается если проверяемый бит равен 1, иначе отвергается.

При стоящем бите PAM в регистре ETH_MACFFR пропускаются все фреймы с групповыми адресами, независимо от результата фильтрации.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HTH																															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:0 **HTH:** Старшие 32 бита хэш-таблицы.

Младший регистр хэш-таблицы Ethernet MAC (ETH_MACHTLR)

Смещение адреса: 0x000C

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HTL																															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:0 **HTL:** 32 бита хэш-таблицы.

Регистр адреса MII Ethernet MAC (ETH_MACMIIAR)

Смещение адреса: 0x0010

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																PA					MR					Reserved	CR			MW	MB
																rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rc_w1

— Биты 31:16 Резерв, не трогать.

— Биты 15:11 **PA:** Адрес устройства PHY из 32 возможных.

— Биты 10:6 **MR:** Регистр MII устройства PHY.

— Бит 5 Резерв, не трогать.

— Биты 4:2 **CR:** Диапазон частот HCLK для получения тактов MDC

000 60-72 MHz HCLK/42

001 Резерв -

010 20-35 MHz HCLK/16

011 35-60 MHz HCLK/26

100, 101, 110, 111 Резерв -

— Бит 1 **MW:** Запись MII

1 - Операция записи PHY через регистр данных MII.

0 - Операция чтения PHY в регистр данных MII.

— Бит 0 **MB:** MII занят (или Запуск операции)

Писать в регистры ETH_MACMIIAR и ETH_MACMIIDR можно только при снятом бите. Установка бита запускает работу PHY. Снимает аппаратно.

Регистр данных MII Ethernet MAC (ETH_MACMIIDR)

Смещение адреса: 0x0014

По сбросу: 0x0000 0000

Промежуточный регистр между программой и регистром PHY, заданным в ETH_MACMIIAR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																MD															
																rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:16 Резерв, не трогать.

— Биты 15:0 **MD:** 16-бит данные MII из PHY после операции чтения и для операции записи PHY.

Регистр управления потоком Ethernet MAC (ETH_MACFCR)

Смещение адреса: 0x0018

По сбросу: 0x0000 0000

Управляет приёмом и выдачей фреймов Паузы. Запись со стоящим битом FCB выдаёт фрейм паузы. Он остаётся стоять всё время передачи по кабелю, писать в регистр можно только при снятом бите.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PT																Reserved								ZQPD	Reserved	PLT		UPFD	RFCE	TFCE	FCB/ BPA
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw									rw		rw	rw	rw	rw	rw	rc_w1 /rw

— Биты 31:16 **PT:** Значение времени паузы в передаваемом фрейме

Если время сконфигурировано с двойной синхронизацией в домене тактов MII, то писать в этот регистр можно только после не менее 4 тактов домена назначения.

— Биты 15:8 Резерв, не трогать.

— Бит 7 **ZQPD:** Запрет выдачи фреймов Zero-quanta паузы

— Бит 6 Резерв, не трогать.

— Биты 5:4 **PLT:** Нижний порог паузы

Это время таймера паузы, после которого автоматически передаётся новый фрейм паузы. Должен быть меньше времени паузы в битах[31:16]. Например, если PT = 100H (256 времён слота), и PLT = 01, то второй фрейм PAUSE будет выдан, если можно, через 228 (256 – 28) времён слота после первого.

00 - Время паузы минус 4 времени слота

01 - Время паузы минус 28 времён слота

10 - Время паузы минус 144 времени слота

11 - Время паузы минус 256 времён слота

Время слота это время передачи 512 битов (64 байта) по MII.

— Бит 3 **UPFD:** Обнаружение одноадресного фрейма паузы

1 - Кроме фреймов паузы с уникальным групповым адресом обнаруживаются фреймы паузы с адресом, лежащим в регистрах ETH_MACA0HR и ETH_MACA0LR.

0 - Только фреймы с уникальным групповым адресом.

— Бит 2 **RFCE:** Разрешение приёма фреймов Паузы с прекращением передач

— Бит 1 **TFCE:** Разрешение передачи фреймов Паузы

В полном дуплексе разрешает выдачу фреймов паузы.

В полу-дуплексе разрешает операции обратного давления.

— Бит 0 **FCB/BPA**: Управление потоком занято (Запуск)/Включение обратного давления

В полном дуплексеThis bit initiates a Pause Control frame in Full-duplex mode and activates the back pressure function in Half-duplex mode if TFCE bit is set.

В полном дуплексе запускает выдачу фрейма паузы. Стоит всё время передачи фрейма паузы, снимается аппаратно. Писать в регистр можно только при снятом бите.

В полу-дуплексе, если одновременно стоит бит TFCE, то ядро MAC выставляет обратное давление. При этом по принятию ядром MAC нового фрейма передатчик начинает передавать JAM-паттерн, вызывая коллизию. В полном дуплексе BPA автоматически отключается.

Регистр тега VLAN Ethernet MAC (ETH_MACVLANTR)

Смещение адреса: 0x001C

По сбросу: 0x0000 0000

Фреймы VLAN имеют в поле Типа/Длины число 0x8100, и тег VLAN в следующих двух байтах. Появление такого фрейма ставит бит VLAN в слове статуса. Допустимая длина фрейма увеличивается с 1518 байт до 1522 байт.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															VLANTC	VLANTI															
															rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:17 Резерв, не трогать.

— Бит 16 **VLANTC:**12-бит сравнение тега VLAN

1 - Для сравнения и фильтрации используются биты [11:0] идентификатора VLAN.

0 - Сравниваются 16 бит идентификатора VLAN принятого фрейма.

— Биты 15:0 **VLANTI:** Идентификатор тега VLAN фреймов приёма

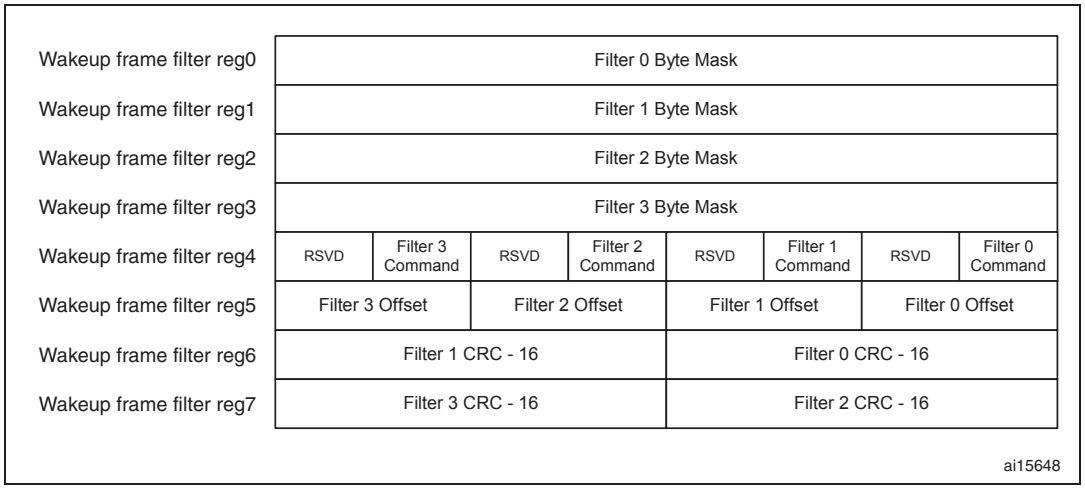
Биты[15:13] это приоритет пользователя, бит[12] это индикатор канонического формата (CFI), биты[11:0] это поле идентификатора VLAN (VID). Если идентификатор нулевой, то он не проверяется и фреймами VLAN объявляются все фреймы с полем типа, равным 0x8100.

Регистр фильтра фреймов удалённой побудки Ethernet MAC (ETH_MACRWUFFR)

Смещение адреса: 0x0028

По сбросу: 0x0000 0000

Это адрес доступа к восьми регистрам фильтра фреймов удалённой побудки по восьми последовательным операциям чтения или по восьми последовательным операциям записи. Этот регистр содержит старшие 16 бит 7-го адреса MAC.



Регистр управления и состояния PMT Ethernet MAC (ETH_MACPMTCSR)

Смещение адреса: 0x002C

По сбросу: 0x0000 0000

Определяет и управляет событиями побудки.

31	30–10	9	8–7	6	5	4–3	2	1	0
WFFRPR	Reserved	GU	Reserved	WFR	MPR	Reserved	WFE	MPE	PD
rs	Res.	rw		rc_r	rc_r		rw	rw	rs

— Бит 31 **WFFRPR:** Сброс указателя регистров фильтра фреймов побудки
Обнуляет указатель, снимается сам через 1 такт.

— Биты 30:10 Резерв, не трогать.

— Бит 9 **GU:** Разрешение опознания фрейма побудки в любом одноадресном пакете.

— Биты 8:7 Резерв, не трогать.

— Бит 6 **WFR:** Принят фрейм побудки
Ставится аппаратно, снимается чтением этого регистра.

— Бит 5 **MPR:** Принят Magic packet
Ставится аппаратно, снимается чтением этого регистра.

— Биты 4:3 Резерв, не трогать.

— Бит 2 **WFE:** Разрешение выдачи события PMT по фрейму побудки

— Бит 1 **MPE:** Разрешение выдачи события PMT по Magic Packet

— Бит 0 **PD:** Выключение питания
Все принимаемые фреймы теряются. Снимается автоматически по приёму Magic Packet или фрейма побудки. Ставить этот бит нужно только при стоящих битах WFE или MPE.

Регистр состояния прерываний Ethernet MAC (ETH_MACSR)

Смещение адреса: 0x0038

По сбросу: 0x0000 0000

15–10	9	8–7	6	5	4	3	2–0
Reserved	TSTS	Reserved	MMCTS	MMCRS	MMCS	PMTS	Reserved
	rc_r		r	r	r	r	

— Биты 15:10 Резерв, не трогать.

— Бит 9 **TSTS:** Статус запуска по времени
Ставится когда системное время достигает значения, заложенного в регистрах Целевого времени. Снимается чтением регистра ETH_PTPTSSR.

— Биты 8:7 Резерв, не трогать.

— Бит 6 **MMCTS:** Статус передачи MMC

Ставится по появлению прерываний в регистре ETH_MMCTIR. Снимается после очистки всех прерываний в регистре ETH_MMCTIR.

— Бит 5 **MMCRS:** Статус приёма MMC

Ставится по появлению прерываний в регистре ETH_MMCRIR. Снимается после очистки всех прерываний в регистре ETH_MMCRIR.

— Бит 4 **MMCS:** Статус MMC

Ставится при любом стоящем бите 6:5. Снимается при очистке обоих.

— Бит 3 **PMTS:** Статус PMT

Ставится по приёму Magic packet или фрейма Побудки в режиме выключения питания (см. биты 5 и 6 в регистре ETH_MACPMTCSR). Снимается при очистке обоих битов чтением регистра ETH_MACPMTCSR.

— Биты 2:0 Резерв, не трогать.

Регистр маски прерываний Ethernet MAC (ETH_MACIMR)

Смещение адреса: 0x003C

По сбросу: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						TSTIM	Reserved					PMTIM	Reserved		
						rw						rw			

— Биты 15:10 Резерв, не трогать.

— Бит 9 **TSTIM:** Запрет прерывания запуска по времени.

— Биты 8:4 Резерв, не трогать.

— Бит 3 **PMTIM:** Запрет прерывания PMT по биту PMTS в регистре ETH_MACSR.

— Биты 2:0 Резерв, не трогать.

Старший регистр адреса 0 Ethernet MAC (ETH_MACA0HR)

Смещение адреса: 0x0040

По сбросу: 0x8000 FFFF

NB: Байты DA передаются по MII начиная с младшего.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MO	Reserved															MACA0H															
1																rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Бит 31 **MO:** Всегда 1.

— Биты 30:16 Резерв, не трогать.

— Биты 15:0 **MACA0H:** Биты[47:32] MAC адреса 0 станции

Младший регистр адреса 0 Ethernet MAC (ETH_MACA0LR)

Смещение адреса: 0x0044

По сбросу: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MACA0L																															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:0 **MACA0L:** Биты[31:0] MAC адреса 0 станции.

Старший регистр адреса 1 Ethernet MAC (ETH_MACA1HR)

Смещение адреса: 0x0044

По сбросу: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AE	SA	MBC						Reserved								MACA1H															
rw	rw	rw	rw	rw	rw	rw	rw									rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Бит 31 **AE:** Разрешение участия MAC адреса 1 в полной фильтрации адреса.

— Бит 30 **SA:** Это адрес источника

1 - MAC адрес 1 [47:0] это адрес источника.

0 - MAC адрес 1 [47:0] это адрес получателя.

— Биты 29:24 **MBC:** Маска байтов сравнения

Если бит стоит, то соответствующий байт принятых DA/SA в сравнении с MAC адресом 1 из регистров не участвует. Соответствие битов байтам:

– Бит 29: ETH_MACA1HR [15:8]
– Бит 28: ETH_MACA1HR [7:0]
– Бит 27: ETH_MACA1LR [31:24]
...
– Бит 24: ETH_MACA1LR [7:0]

— Биты 23:16 Резерв, не трогать.
— Биты 15:0 **MACA1H:** Биты[47:32] MAC адреса 1 станции.

Младший регистр адреса 1 Ethernet MAC (ETH_MACA1LR)

Смещение адреса: 0x004C

По сбросу: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MACA1L																															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:0 **MACA1L:** Биты[31:0] MAC адреса 1 станции (надо писать при инициализации).

Старший регистр адреса 2 Ethernet MAC (ETH_MACA2HR)

Смещение адреса: 0x0050

По сбросу: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AE	SA	MBC						Reserved								MACA2H															
rw	rw	rw	rw	rw	rw	rw	rw									rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Бит 31 **AE:** Разрешение участия MAC адреса 2 в полной фильтрации адреса.
— Бит 30 **SA:** Это адрес источника
1 - MAC адрес 2 [47:0] это адрес источника.
0 - MAC адрес 2 [47:0] это адрес получателя.
— Биты 29:24 **MBC:** Маска байтов сравнения
Если бит стоит, то соответствующий байт принятых DA/SA в сравнении с MAC адресом 2 из регистров не участвует. Соответствие битов байтам:

– Бит 29: ETH_MACA2HR [15:8]
– Бит 28: ETH_MACA2HR [7:0]
– Бит 27: ETH_MACA2LR [31:24]
...
– Бит 24: ETH_MACA2LR [7:0]

— Биты 23:16 Резерв, не трогать.
— Биты 15:0 **MACA2H:** Биты[47:32] MAC адреса 2 станции.

Младший регистр адреса 2 Ethernet MAC (ETH_MACA2LR)

Смещение адреса: 0x0054

По сбросу: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MACA2L																															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:0 **MACA2L:** Биты[31:0] MAC адреса 2 станции (надо писать при инициализации).

Старший регистр адреса 3 Ethernet MAC (ETH_MACA3HR)

Смещение адреса: 0x0058

По сбросу: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AE	SA	MBC						Reserved								MACA3H															
rw	rw	rw	rw	rw	rw	rw	rw									rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

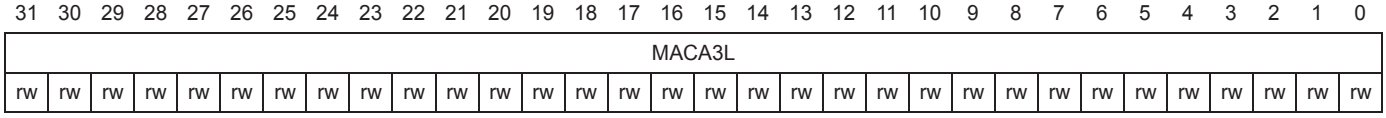
— Бит 31 **AE:** Разрешение участия MAC адреса 3 в полной фильтрации адреса.
— Бит 30 **SA:** Это адрес источника
1 - MAC адрес 3 [47:0] это адрес источника.
0 - MAC адрес 3 [47:0] это адрес получателя.

— Биты 29:24 **MBC:** Маска байтов сравнения
Если бит стоит, то соответствующий байт принятых DA/SA в сравнении с MAC адресом 2 из регистров не участвует. Соответствие битов байтам:
– Бит 29: ETH_MACA3HR [15:8]
– Бит 28: ETH_MACA3HR [7:0]
– Бит 27: ETH_MACA3LR [31:24]
...
– Бит 24: ETH_MACA3LR [7:0]
— Биты 23:16 Резерв, не трогать.
— Биты 15:0 **MACA3H:** Биты[47:32] MAC адреса 3 станции.

Младший регистр адреса 3 Ethernet MAC (ETH_MACA3LR)

Смещение адреса: 0x005C

По сбросу: 0xFFFF FFFF



— Биты 31:0 **MACA3L:** Биты[31:0] MAC адреса 3 станции (надо писать при инициализации).

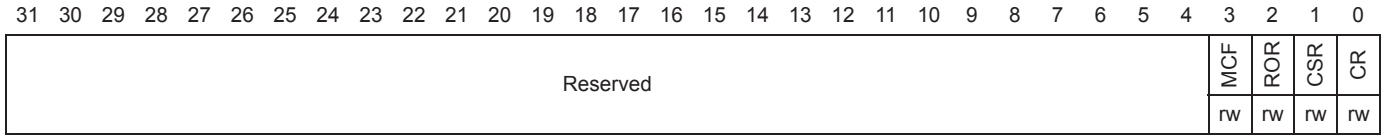
33.8.2. Описание регистров MMC

Регистр управления Ethernet MMC (ETH_MMCCR)

Смещение адреса: 0x0100

По сбросу: 0x0000 0000

Определяет режим работы счётчиков MMC.



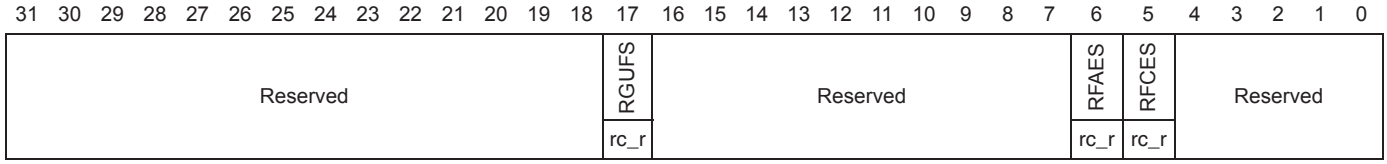
— Биты 31:4 Резерв, не трогать.
— Бит 3 **MCF:** Заморозка счётчиков MMC в текущем положении
Если какой-либо счётчик читается при стоящем бите ROR, то он сбрасывается и в этом режиме.
— Бит 2 **ROR:** Сброс счётчика после прочтения младшего байта
— Бит 1 **CSR:** Запрет возврата счётчиков к нулю при достижении максимального значения
— Бит 0 **CR:** Сброс всех счётчиков, снимается через 1 такт.

Регистр прерываний счётчиков приёма Ethernet MMC (ETH_MMCRIR)

Смещение адреса: 0x0104

По сбросу: 0x0000 0000

Запрос прерывания от счётчика ставится при достижении половины максимального значения (ставится старший бит). Снимается после чтения соответствующего счётчика.



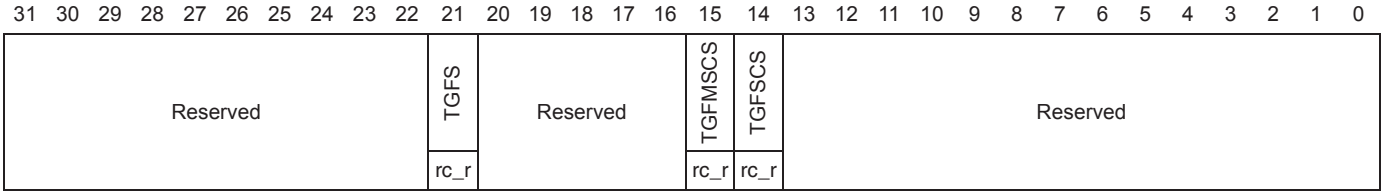
— Биты 31:18 Резерв, не трогать.
— Бит 17 **RGUFS:** Приём хороших одноадресных фреймов.
— Биты 16:7 Резерв, не трогать.
— Бит 6 **RFAES:** Приём фреймов с ошибкой выравнивания.
— Бит 5 **RFCES:** Приём фреймов с ошибкой CRC.
— Биты 4:0 Резерв, не трогать.

Регистр прерываний счётчиков передачи Ethernet MMC (ETH_MMCTIR)

Смещение адреса: 0x0108

По сбросу: 0x0000 0000

Запрос прерывания от счётчика ставится при достижении половины максимального значения (ставится старший бит). Снимается после чтения соответствующего счётчика.



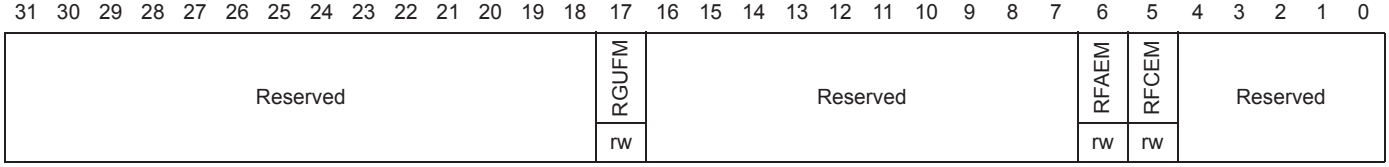
— Биты 31:22 Резерв, не трогать.
— Бит 21 **TGFS:** Передача хороших фреймов.
— Биты 20:16 Резерв, не трогать.
— Бит 15 **TGFMSCS:** Передача хороших фреймов с более чем одной коллизией.
— Бит 14 **TGFSCS:** Передача хороших фреймов с одной коллизией.
— Биты 13:0 Резерв, не трогать.

Регистр маски прерываний счётчиков приёма Ethernet MMC (ETH_MMCRIMR)

Смещение адреса: 0x010C

По сбросу: 0x0000 0000

Стоящий бит маски запрещает прерывание.



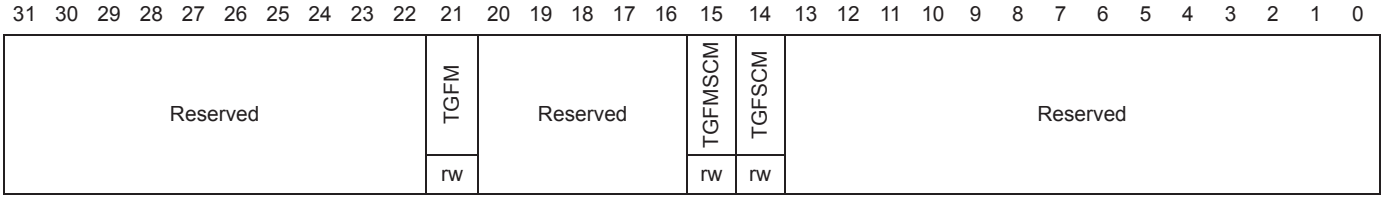
— Биты 31:18 Резерв, не трогать.
— Бит 17 **RGUFM:** Приём хороших одноадресных фреймов.
— Биты 16:7 Резерв, не трогать.
— Бит 6 **RFAEM:** Приём фреймов с ошибкой выравнивания.
— Бит 5 **RFCEM:** Приём фреймов с ошибкой CRC.
— Биты 4:0 Резерв, не трогать.

Регистр маски прерываний счётчиков передачи Ethernet MMC (ETH_MMCTIMR)

Смещение адреса: 0x0110

По сбросу: 0x0000 0000

Стоящий бит запрещает соответствующее прерывание.



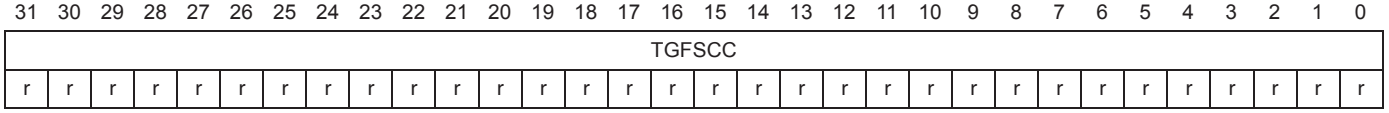
— Биты 31:22 Резерв, не трогать.
— Бит 21 **TGFM:** Передача хороших фреймов.
— Биты 20:16 Резерв, не трогать.
— Бит 15 **TGFMSCM:** Передача хороших фреймов с более чем одной коллизией.
— Бит 14 **TGFSCM:** Передача хороших фреймов с одной коллизией.
— Биты 13:0 Резерв, не трогать.

Регистр числа успешно переданных фреймов после одной коллизии Ethernet MMC (ETH_MMCTGFSCCR)

Смещение адреса: 0x014C

По сбросу: 0x0000 0000

Значение **TGFSCC** действительно в полу-дуплексе.



Регистр числа успешно переданных фреймов после нескольких коллизий Ethernet MMC (ETH_MMCTGFMSCCR)

Смещение адреса: 0x0150

По сбросу: 0x0000 0000

Значение **TGFMSCC** действительно в полу-дуплексе.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TGFMSCC																															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Регистр числа успешно переданных фреймов Ethernet MMC (ETH_MMCTGFCR)

Смещение адреса: 0x0168

По сбросу: 0x0000 0000

This register contains the number of good frames transmitted.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TGFC																															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

— Биты 31:0 **TGFC:** Transmitted good frames counter

Регистр числа принятых фреймов с ошибкой CRC Ethernet MMC (ETH_MMCRFCECR)

Смещение адреса: 0x0194

По сбросу: 0x0000 0000

RFCEC это число фреймов, принятых с ошибкой CRC.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RFCEC																															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Регистр числа принятых фреймов с ошибкой выравнивания Ethernet MMC (ETH_MMCRFAECR)

Смещение адреса: 0x0198

По сбросу: 0x0000 0000

RFAEC это число фреймов, принятых с ошибкой выравнивания (дробный полубайт).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RFAEC																															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Регистр числа принятых одноадресных фреймов Ethernet MMC (ETH_MMCRGUFCR)

Смещение адреса: 0x01C4

По сбросу: 0x0000 0000

RGUFC это число хорошо принятых одноадресных фреймов.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RGUFC																															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

33.8.3. Регистры меток времени IEEE 1588

Регистр управления меток времени Ethernet PTP (ETH_PTPTSCR)

Смещение адреса: 0x0700

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																										TTSARU	TSITE	TSSTU	TSSTI	TSFCU	TSE
																										rw	rw	rw	rw	rw	rw

— Биты 31:6 Резерв, не трогать.

— Бит 5 **TSARU:** Обновление регистра слагаемого при тонкой коррекции

Ставить можно только обнулённый бит.

— Бит 4 **TSITE:** Разрешение прерывания запуска по времени, после прерывания снимается.

— Бит 3 **TSSTU:** Обновление системного времени

Системное время изменяется на значение регистров обновления метки времени. Биты TSSTU и TSSTI должны быть чистыми. По завершению обновления бит снимается.

— Бит 2 **TSSTI:** Инициализация системного времени

В системное время пишется на значение регистров обновления метки времени. Ставить можно только обнулённый бит. После записи бит снимается.

— Бит 1 **TSFCU:** Грубая или тонкая коррекция

1 - Нужна Тонкая коррекция.

0 - Грубая коррекция.

— Бит 0 **TSE:** Разрешение передачи и приёма меток времени

После установки этого бита надо всегда инициализировать системное время.

Регистр суб-секундного инкремента Ethernet PTP (ETH_PTPSSIR)

Смещение адреса: 0x0704

По сбросу: 0x0000 0000

При Грубой коррекции (бит TSFCU в ETH_PTPTSCR) это значение добавляется в системное время по каждому такту HCLK. При Тонкой коррекции добавляется в системное время по переполнению Накопителя.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																								STSSI							
																								rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:8 Резерв, не трогать.

— Биты 7:0 **STSSI:** Суб-секундный инкремент системного времени

Старший регистр метки времени Ethernet PTP (ETH_PTPTSHR)

Смещение адреса: 0x0708

По сбросу: 0x0000 0000

STS это текущее значение секунд системного времени, обновляется само и постоянно.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STS																															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Младший регистр метки времени Ethernet PTP (ETH_PTPTSLR)

Смещение адреса: 0x070C

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STPNS	STSS																														
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

— Бит 31 **STPNS:** Знак системного времени, плюс или минус

Обычно всегда сброшен (плюс).

— Биты 30:0 **STSS:** Суб-секунды системного времени с точностью 0.46 ns.

Старший регистр обновления метки времени Ethernet PTP (ETH_PTPTSHUR)

Смещение адреса: 0x0710

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSUS																															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:0 **TSUS:** Секунды обновления системного времени

Используется для записи, сложения или вычитания из системного времени. Писать надо до установки битов TSSTI или TSSTU в регистре управления.

Младший регистр обновления метки времени Ethernet PTP (ETH_PTPTSLUR)

Смещение адреса: 0x0714

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSUPNS	TSUSS																														
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Бит 31 **TSUPNS:** Знак обновления, плюс (0) или минус (1)

При стоящем бите TSSTI (инициализация системного времени) должен быть нулевым. Если он стоит при стоящем бите TSSTU, то значение обновления вычитается из системного времени, иначе, добавляется.

— Биты 30:0 **TSUSS:** Суб-секунды обновления системного времени с точностью 0.46 ns.

Писать надо до установки битов `TSSTI` или `TSSTU` в регистре управления.

Регистр слагаемого метки времени Ethernet PTP (ETH_PTPTSAR)

Смещение адреса: 0x0718

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSA																															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:0 **TSA:** Слагаемое времени

Используется только при Тонкой коррекции (бит TSFCU в ETH_PTPTSCR). По каждому такту добавляется в регистр накопителя, системное время обновляется при переполнении Накопителя.

Старший регистр целевого времени Ethernet PTP (ETH_PTPTTHR)

Смещение адреса: 0x071C

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TTSH																															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:0 **TTSH:** Секунды целевого времени

При достижении системным временем значения целевого выдаётся прерывание (бит TSARU в ETH_PTPTSCR).

Младший регистр целевого времени Ethernet PTP (ETH_PTPTTLR)

Смещение адреса: 0x0720

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TTSL																															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:0 **TTSL:** Наносекунды целевого времени (знаковое)

При достижении системным временем значения целевого выдаётся прерывание (бит TSARU в ETH_PTPTSCR).

33.8.4. Описание регистров DMA

Регистр режима шины Ethernet DMA (ETH_DMABMR)

Смещение адреса: 0x1000

По сбросу: 0x0002 0101

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						AAB	FPM	USP	RDP						FB	PM		PBL						Reserved	DSL					DA	SR
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rs

— Биты 31:26 Резерв, не трогать.

— Бит 25 **AAB:** Выравнивание адреса частей

Если этот бит стоит вместе с битом FB, то AHB выдаёт все пакеты, выравненными по младшим битам адреса начала. Если бит FB равен 0, то первый пакет не выравнивается, а последующие таки да..

— Бит 24 **FPM:** Режим 4xPBL

Если бит стоит, то значение PBL (биты [22:17] и [13:8]) умножается на 4. И DMA передаёт данные за максимум 4, 8, 16, 32, 64 или 128 частей, в зависимости от значения PBL.

— Бит 23 **USP:** Использовать отдельный PBL

1 - RxDMA как PBL использует биты[22:17], а TxDMA биты[13:8].

0 - RxDMA и TxDMA для PBL используют биты[13:8].

— Биты 22:17 **RDP:** RxDMA PBL

Максимальное число частей одной передачи RxDMA. Допустимое значение RDP равно 1, 2, 4, 8, 16, или 32. Иное приводит к неадекватному поведению.

Работает только при стоящем бите USP.

— Бит 16 **FB:** Фиксированный пакет

1 - В начале нормальных передач AHB использует только SINGLE, INCR4, INCR8 или INCR16 операции.

0 - SINGLE и INCR.

— Биты 15:14 PM: Отношение приоритетов Rx:Tx при чистом бите DA

00: 1:1

01: 2:1

10: 3:1

11: 4:1

— Биты 13:8 **PBL:** Программируемая длина пакета

Максимальное число частей одной передачи DMA.

Допустимое значение PBL равно 1, 2, 4, 8, 16, или 32. Иное приводит к неадекватному поведению. При стоящем бите USP используется только TxDMA.

Ограничения PBL такие:

– Максимально допустимое значение частей (PBL) ограничено размером Tx FIFO и Rx FIFO.
– FIFO вынужденно поддерживает максимальное число частей, равное половине глубины FIFO.
– Если PBL общий для приёма и передачи, то надо учитывать меньшую глубину RxFIFO и TxFIFO.
– Недопустимое значение PBL сводит систему с ума.

— Бит 7 Резерв, не трогать.

— Биты 6:2 **DSL:** Длина пропуска дескриптора

Это число пропускаемых слов между двумя несцеплёнными дескрипторами от конца первого до начала второго. Если DSL равен нулю, то это неразрывная таблица дескрипторов в режиме кольца.

— Бит 1 **DA:** Арбитраж DMA

0: Круговой с отношением приоритетов Rx:Tx в битах [15:14]

1: Rx приоритетнее Tx

— Бит 0 **SR:** Программный сброс всех подсистем MAC DMA

Автоматически снимается после завершения сброса всех доменов тактирования. В регистры ядра можно писать только при нулевом бите.

Регистр требования опроса списка дескрипторов передачи Ethernet DMA (ETH_DMATPDR)

Смещение адреса: 0x1004

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TPD																															
rw_wt																															

— Биты 31:0 **TPD**: Требование опроса передачи

После записи любого числа DMA проверяет текущий дескриптор (адрес в регистре ETH_DMACHTDR) на доступность. Если он принадлежит CPU, то передача возвращается в Останов и ставится бит 2 регистра ETH_DMASR, иначе передача возобновляется.

Регистр требования опроса списка дескрипторов приёма Ethernet DMA (ETH_DMARPDR)

Смещение адреса: 0x1008

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RPD																															
rw_wt																															

— Биты 31:0 **RPD:** Требование опроса приёма

После записи любого числа DMA проверяет текущий дескриптор (адрес в регистре ETH_DMACHRDR) на доступность. Если он принадлежит CPU, то передача возвращается в Останов и ставится бит 7 регистра ETH_DMASR, иначе передача возобновляется.

Регистр базового адреса списка дескрипторов приёма Ethernet DMA (ETH_DMARDLAR)

Смещение адреса: 0x100C

По сбросу: 0x0000 0000

Список дескрипторов лежит в памяти и должен быть выравнен на границу слова. DMA сам выравнивает адрес на границу ширины шины, обнуляя младшие биты (биты[1/2/3:0] для 32/64/128-бит ширины). Писать в регистр можно только при остановленном приёме.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRL																															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:0 **SRL:** Начало списка приёма

Регистр базового адреса списка дескрипторов передачи Ethernet DMA (ETH_DMATDLAR)

Смещение адреса: 0x1010

По сбросу: 0x0000 0000

Список дескрипторов лежит в памяти и должен быть выравнен на границу слова. DMA сам выравнивает адрес на границу ширины шины, обнуляя младшие биты (биты[1/2/3:0] для 32/64/128-бит ширины). Писать в регистр можно только при остановленном приёме.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRL																															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:0 **STL:** Начало списка передачи

Регистр состояния Ethernet DMA (ETH_DMASR)

Смещение адреса: 0x1014

По сбросу: 0x0000 0000

Нерезервированные биты из ETH_DMASR[16:0] снимаются записью 1 в них. Прерывания по ним маскируются в регистре ETH_DMAIER.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		TSTS	PMTS	MMCS	Reserved	EBS			TPS			RPS			NIS	AIS	ERS	FBES	Reserved		ETS	RWTS	RPSS	RBUS	RS	TUS	ROS	TJTS	TBUS	TPSS	TS
		r	r	r		r	r	r	r	r	r	r	r	r	rc-w1	rc-w1	rc-w1	rc-w1			rc-w1	rc-w1	rc-w1	rc-w1	rc-w1	rc-w1	rc-w1	rc-w1	rc-w1	rc-w1	rc-w1

— Биты 31:30 Резерв, не трогать.

— Бит 29 **TSTS:** Статус прерывания запуска по времени

Снимается чтением регистра состояния ядра MAC, очищая источник (бит 9).

— Бит 28 **PMTS:** Статус прерывания PMT

Снимается чтением соответствующих регистров ядра MAC, очищая источники.

— Бит 27 **MMCS:** Статус прерывания MMC

Снимается чтением соответствующих регистров ядра MAC, очищая источники.

— Бит 26 Резерв, не трогать.

— Биты 25:23 **EBS:** Биты причины фатальной ошибки шины (ETH_DMASR[13])

Бит 23: 1 - Ошибка передачи TxDMA, 0 - Ошибка передачи RxDMA

Бит 24: 1 - Ошибка при чтении, 0 - Ошибка при записи

Бит 25: 1 - Ошибка при доступе к дескриптору, 0 - Ошибка при доступе к буферу данных

— Биты 22:20 **TPS:** Состояние процесса передачи (TxDMA FSM)

000: Стоит; Была команда Сброса или Останова передачи

001: Работа; Выборка дескриптора передачи

010: Работа; Ждёт статус
011: Работа; Пишет данные из буфера памяти в TxFIFO
100, 101: Резерв
110: Приостановка; Недоступен дескриптор передачи или Исчерпание буфера
111: Работа; Закрывает дескриптор передачи

— Биты 19:17 **RPS:** Состояние процесса приёма (RxDMA FSM)
000: Стоит: Была команда Сброса или Останова приёма
001: Работа: Выборка дескриптора приёма
010: Резерв
011: Работа: Ждёт приёмного пакет
100: Приостановка: Недоступен дескриптор приёма
101: Работа: Закрывает дескриптор приёма
110: Резерв
111: Работа: Передаёт данные из RxFIFO в память

— Бит 16 **NIS:** Общее нормальное прерывание
Это логическое OR запросов прерываний, разрешённых в регистре ETH_DMAIER:
– ETH_DMASR [0]: прерывание Передачи
– ETH_DMASR [2]: Буфер передачи недоступен
– ETH_DMASR [6]: прерывание Приёма
– ETH_DMASR [14]: прерывание Раннего приёма
Бит снимается записью 1 после снятия бита причины прерывания.

— Бит 15 **AIS:** Общее ненормальное прерывание
Это логическое OR запросов прерываний, разрешённых в регистре ETH_DMAIER:
– ETH_DMASR [1]: Передача остановлена
– ETH_DMASR [3]: Таймаут трёпа
– ETH_DMASR [4]: Переполнение RxFIFO
– ETH_DMASR [5]: Исчерпание передачи
– ETH_DMASR [7]: Буфер приёма недоступен
– ETH_DMASR [8]: Приём остановлен
– ETH_DMASR [9]: Таймаут сторожевого таймера приёма
– ETH_DMASR [10]: прерывание Ранней передачи
– ETH_DMASR [13]: Фатальная ошибка шины
Бит снимается записью 1 после снятия бита причины прерывания.

— Бит 14 **ERS:** Ранний приём
DMA заполнил первый буфер данных пакета. Прерывание приёма ETH_DMASR[6] автоматически снимает этот бит.

— Бит 13 **FBES:** Фатальная ошибка шины
Причина ошибки указана в битах [25:23]. Соответствующая машина DMA прекращает доступ к шине.

— Биты 12:11 Резерв, не трогать.

— Бит 10 **ETS:** Ранняя передача
Передаваемый фрейм полностью записан в TxFIFO.

— Бит 9 **RWTS:** Таймаут сторожевого таймера приёма
Длина принятого фрейма больше 2 048 байт.

— Бит 8 **RPSS:** Приём остановлен

— Бит 7 **RBUS:** Буфер приёма недоступен
Следующий дескриптор принадлежит CPU, а предыдущий принадлежал DMA. Приём приостановлен. Он восстановится после того как CPU изменит принадлежность дескриптора выдаст и требование опроса приёма или будет принят следующий распознанный фрейм.

— Бит 6 **RS:** Фрейм принят
Статус приёма уже записан в дескриптор. Машина остаётся в Работе.

— Бит 5 **TUS:** Исчерпание передачи
Передача приостановлена, ставится бит исчерпания TDES0[1].

— Бит 4 **ROS:** Переполнение буфера приёма
Если программе передана часть фрейма, то ставится бит переполнения RDES0[11].

— Бит 3 **TJTS:** Таймаут трёпа

Допустимое время активной передачи исчерпано, машина Остановлена, ставится флаг TDES0[14].

— Бит 2 **TBUS:** Буфер передачи недоступен

Следующий дескриптор принадлежит CPU. Передача приостановлена. Причина лежит в битах[22:20]. Передача восстановится после того как CPU изменит принадлежность дескриптора выдаст и требование опроса передачи.

— Бит 1 **TPSS:** Передача остеновлена.

— Бит 0 **TS:** Конец передачи фрейма, ставится TDES1[31].

Регистр режима работы Ethernet DMA (ETH_DMAOMR)

Смещение адреса: 0x1018

По сбросу: 0x0000 0000

При инициализации DMA этот регистр пишут последним

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					DTCEFD	RSF	DFRF	Reserved		TSF	FTF	Reserved			TTC			ST	Reserved					FEF	FUGF	Reserved	RTC		OSF	SR	Reserved
					rw	rw	rw			rw	rs				rw	rw	rw	rw						rw	rw		rw	rw	rw	rw	

— Биты 31:27 Резерв, не трогать.

— Бит 26 **DTCEFD:** Запрет выброса фреймов с ошибкой контрольной суммы только у вложенных пакетов TCP/IP.

— Бит 25 **RSF:** Режим Накопления (Store-and-forward) при приёме

1 - Фреймы читаются из RxFIFO после приёма всего фрейма, игнорируя биты RTC.

0 - RxFIFO работает в режиме Cut-through, порог начала выдачи задан в битах RTC.

— Бит 24 **DFRF:** Запрет слива принятых фреймов при недоступности дескриптора/буфера

— Биты 23:22 Резерв, не трогать.

— Бит 21 **TSF:** Режим Накопления (Store-and-forward) при передаче

1 - TxFIFO выдаются только полностью записанные фреймы, игнорируя биты TTC.

0 - TxFIFO учитывает порог начала выдачи, заданный в битах TTC.

Бит можно писать только при остановленной передаче.

— Бит 20 **FTF:** Слить TxFIFO

Контроллер TxFIFO и указатели сбрасываются в начальное состояние, все данные теряются. Бит снимается автоматически после завершения сброса. При стоящем бите писать в этот регистр нельзя.

— Биты 19:17 Резерв, не трогать.

— Биты 16:14 **TTC:** Порог передачи из TxFIFO (при TSF=0)

Передача начинается если размер части фрейма в TxFIFO превышает порог, или записан полный фрейм.

000: 64
001: 128
010: 192
011: 256
100: 40
101: 32
110: 24
111: 16

— Бит 13 **ST:** Старт/Стоп передачи

1 - Старт. Машина идёт в Работу, DMA ищет в списке передаваемый фрейм, начиная с текущей позиции (начала списка в ETH_DMATDLAR или сохранённого состояния). Если текущий дескриптор не принадлежит DMA, то машина идёт в Приостановку и ставится бит недоступности буфера (ETH_DMASR[2]). Старт работает только при остановленной передаче. Если команду выдать до установки регистра ETH_DMATDLAR, то DMA сходит с ума.

0 - Стоп. После завершения передачи текущего фрейма машина уходит в Останов и сохраняется позиция следующего фрейма для рестарта. Команда Стоп работает только после завершения передачи текущего фрейма или в состоянии Приостановки.

— Биты 12:8 Резерв, не трогать.

— Бит 7 **FEF:** Выдавать сбойные фреймы из RxFIFO

1 - DMA выдаются все фреймы, кроме коротышей (runt).

0 - RxFIFO отбрасывает все фреймы с ошибкой (ошибка CRC, коллизия, гигантский фрейм, таймаут сторожевого таймера, переполнение). Но, если в режиме порога начало фрейма уже поступило на сторону чтения из RxFIFO, то фрейм не выбрасывается.

— Бит 6 **FUGF:** Выдавать маломерные хорошие фреймы

1 - Rx FIFO выдаёт хорошие фреймы короче 64 байт, включая CRC и расширитель.

0 - Rx FIFO отбрасывает фреймы короче 64 байт, кроме прошедших порог RTC.

— Бит 5 Резерв, не трогать.

— Биты 4:3 **RTC:** Порог приёма

Если длина принятой части фрейма превышает порог, то начинается выдача его DMA. Фреймы короче порога выдаются автоматически.

NB: Значение порога 11 не применимо если глубина RxFIFO равна 128 байтам.

NB: Эти биты работают только при RSF=0.

00: 64

01: 32

10: 96

11: 128

— Бит 2 **OSF:** Обрабатывать второй фрейм

Если стоит, то DMA начинает обрабатывать второй фрейм ещё до получения статуса первого.

— Бит 1 **SR:** Старт/Стоп приёма

1 - Старт. Машина идёт в Работу. DMA выбирает из списка приёма принадлежащий DMA дескриптор (с текущего положения или начала списка в регистре ETH_DMARDLAR) и начинает передачу в память. Если такого дескриптора нет, то приём приостанавливается и ставится бит недоступности буфера (ETH_DMASR [7]). Команда Старт действует только при остановленном приёме. Выдавать команду надо после записи регистра DMA ETH_DMARDLAR, иначе DMA может спятить.

0 - После завершения передачи текущего фрейма машина уходит в Останов и сохраняется позиция следующего фрейма для рестарта. Команда Стоп работает только при ожидании приёма пакета или в состоянии Приостановки.

— Бит 0 Резерв, не трогать.

Регистр разрешения прерываний Ethernet DMA (ETH_DMAIER)

Смещение адреса: 0x101C

По сбросу: 0x0000 0000

Стоящая 1 в бите разрешает соответствующее прерывание из регистра ETH_DMASR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															NISE	AISE	ERIE	FBEIE	Reserved		ETIE	RWTIE	RPSIE	RBUIE	RIE	TUIE	ROIE	TJTIE	TBUIE	TPSIE	TIE
															rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:17 Резерв, не трогать.

— Бит 16 **NISE:** Общее нормальное прерывание

Разрешает общее прерывание по битам:

- ETH_DMASR [0]: прерывание Передачи
- ETH_DMASR [2]: Буфер передачи недоступен
- ETH_DMASR [6]: прерывание Приёма
- ETH_DMASR [14]: прерывание Раннего приёма

— Бит 15 **AISE:** Общее ненормальное прерывание

Разрешает общее прерывание по битам:

- ETH_DMASR [1]: Передача остановлена
- ETH_DMASR [3]: Таймаут трёпа
- ETH_DMASR [4]: Переполнение RxFIFO
- ETH_DMASR [5]: Исчерпание передачи
- ETH_DMASR [7]: Буфер приёма недоступен
- ETH_DMASR [8]: Приём остановлен
- ETH_DMASR [9]: Таймаут сторожевого таймера приёма
- ETH_DMASR [10]: прерывание Ранней передачи
- ETH_DMASR [13]: Фатальная ошибка шины

— Бит 14 **ERIE:** Ранний приём.
— Бит 13 **FBEIE:** Фатальная ошибка шины.
— Биты 12:11 Резерв, не трогать.
— Бит 10 **ETIE:** Ранняя передач.
— Бит 9 **RWTIE:** Таймаут сторожевого таймера приёма
— Бит 8 **RPSIE:** Приём остановлен.
— Бит 7 **RBUIE:** Буфер приёма недоступен.
— Бит 6 **RIE:** Прерывание Приёма.
— Бит 5 **TUIE:** Исчерпание буфера передачи.
— Бит 4 **ROIE:** Переполнение буфера приёма.
— Бит 3 **TJTIE:** Таймаут трёпа передачи.
— Бит 2 **TBUIE:** Буфер передачи недоступен.
— Бит 1 **TPSIE:** Передача остановлена.
— Бит 0 **TIE:** Прерывание передачи.

Регистр числа потерянных фреймов и переполнения буфера Ethernet DMA (ETH_DMAMFBOCR)

Смещение адреса: 0x1020

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			OFOC	MFA											OMFC	MFC															
			rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r

— Биты 31:29 Резерв, не трогать.
— Бит 28 **OFOC:** Бит переполнения счётчика переполнений FIFO
— Биты 27:17 **MFA:** Фреймы, потерянные программой
— Бит 16 **OMFC:** Бит переполнения счётчика потерянных фреймов
— Биты 15:0 **MFC:** Фреймы, потерянные контроллером по недоступности буфера.

Регистр текущего дескриптора передачи хоста Ethernet DMA (ETH_DMACHTDR)

Смещение адреса: 0x1048

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HTDAP																															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

— Биты 31:0 **HTDAP:** Адрес начала текущего дескриптора передачи, изменяется DMA.

Регистр текущего дескриптора приёма хоста Ethernet DMA (ETH_DMACHRDR)

Смещение адреса: 0x104C

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HRDAP																															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

— Биты 31:0 **HRDAP:** Адрес начала текущего дескриптора приёма, изменяется DMA.

Регистр адреса текущего буфера передачи хоста Ethernet DMA (ETH_DMACHTBAR)

Смещение адреса: 0x1050

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HTBAP																															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

— Биты 31:0 **HTBAP:** Адрес буфера передачи, изменяется DMA.

Регистр адреса текущего буфера приёма хоста Ethernet DMA (ETH_DMACHRBAR)

Смещение адреса: 0x1054

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HRBAP																															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

— Биты 31:0 **HRBAP:** Адрес буфера передачи, изменяется DMA.

33.8.5. Карта регистров Ethernet

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	ETH_MACCR	Reserved						CSTF	eserved	WD	JD	Reserved		IFG			CSD	Reserved	FES	ROD	LM	DM	IPCO	RD	Reserved	APCS	BL		DC	TE	RE	Reserved	
	Reset value							0		0	0			0	0	0	0		0	0	0	0	0	0		0	0	0	0	0	0		
0x04	ETH_MACFFR	RA	Reserved																				HPF	SAF	SAIF	PCF		BFD	PAM	DAIF	HM	HU	PM
	Reset value	0																					0	0	0	0	0	0	0	0	0	0	0
0x08	ETH_MACHTHR	HTH[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	ETH_MACHTLR	HTL[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	ETH_MACMIIAR	Reserved																PA					MR					CR				MW	MB
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	ETH_MACMIIDR	Reserved																MD															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	ETH_MACFCR	PT																Reserved								ZQPD	Reserved	PLT			UPFD	RFCE	TFCE	FCB/BPA
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									0		0	0	0	0	0	0
0x1C	ETH_ MACVLANTR	Reserved															VLANTC	VLANTI															
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	ETH_ MACRWUFFR	Frame filter reg0\Frame filter reg1\Frame filter reg2\Frame filter reg3\Frame filter reg4\...\Frame filter reg7																															
	Reset value	0																															

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2C	ETH_ MACPMTCSR	WFFRPR	Reserved																					GU	Reserved		WFR	MPR	Reserved		WFE	MPE	PD
	Reset value	0																						0			0	0			0	0	0
0x34	ETH_ MACDBGR	Reserved						TFF	TFNEGU	Reserved	TFWA	TFRS	MTP	MTFCS	MMTEA	Reserved								RFFL		Reserved	RFRCS		RFWRA	Reserved	MSFRWCS		MMRPEA
	Reset value							0	0		0	0	0	0	0	0	0							0	0		0	0	0		0	0	0
0x38	ETH_MACSR																	Reserved						TSTS	Reserved		MMCTS	MMCRS	MMCS	PMTS	Reserved		
	Reset value																							0			0	0	0	0			
0x3C	ETH_MACIMR																	Reserved						TSTIM	Reserved					PMTIM	Reserved		
	Reset value																							0						0			
0x40	ETH_MACA0HR	MO	Reserved															MACA0H															
	Reset value	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x44	ETH_MACA0LR	MACA0L																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x48	ETH_MACA1HR	AE	SA	MBC[6:0]						Reserved								MACA1H															
	Reset value	0	0	0	0	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x4C	ETH_MACA1LR	MACA1L																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x50	ETH_MACA2HR	AE	SA	MBC						Reserved								MACA2H															
	Reset value	0	0	0	0	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x54	ETH_MACA2LR	MACA2L																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x58	ETH_MACA3HR	AE	SA	MBC						Reserved								MACA3H															
	Reset value	0	0	0	0	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x5C	ETH_MACA3LR	MACA3L																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x100	ETH_MMCCR	Reserved																										MCFHP	MCP	MCF	ROR	CSR	CR
	Reset value																											0	0	0	0	0	0
0x104	ETH_MMCRIR	Reserved														RGUFS	Reserved										RFAES	RFCES	Reserved					
	Reset value															0											0	0						
0x108	ETH_MMCTIR	Reserved												TGFS	Reserved			TGFMSCS	TGFSCS	Reserved													
	Reset value													0				0	0														
0x10C	ETH_MMCRIMR	Reserved														RGUFM	Reserved										RFAEM	RFCEM	Reserved					
	Reset value															0											0	0						

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x110	ETH_MMCTIMR	Reserved										TGFM	Reserved					TGFMSCM	TGFSCM	Reserved													
	Reset value											0						0	0														
0x14C	ETH_MMCTGFSCCR	TGFSCC																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x150	ETH_MMCTGFMSCCR	TGFMSCC																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x168	ETH_MMCTGFCR	TGFC																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x194	ETH_MMCRFCECR	RFCEC																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x198	ETH_MMCRFAECR	RFAEC																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C4	ETH_MMCRGUFCR	RGUFC																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x700	ETH_PTPTSCR	Reserved													TSPFFMAE	TSCNT		TSSMRME	TSSEME	TSSIPV4FE	TSSIPV6FE	TSSPTPOEFE	TSPTPPSV2E	TSSSR	TSSARFE	Reserved		TTSARU	TSITE	TSSTU	TSSTI	TSFCU	TSE
	Reset value														0	0	0	0	0	1	0	0	0	0	0			0	0	0	0	0	0
0x704	ETH_PTPSSIR	Reserved																										STSSI					
	Reset value																									0	0	0	0	0	0	0	0
0x708	ETH_PTPTSHR	STS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x70C	ETH_PTPTSLR	STPNS	STSS																														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x710	ETH_PTPTSHUR	TSUS																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x714	ETH_PTPTSLUR	TSUPNS	TSUSS																														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x718	ETH_PTPTSAR	TSA																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x71C	ETH_PTPTTHR	TTSH																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x720	ETH_PTPTTLR	TTSL																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x728	ETH_PTPTSSR	Reserved																														TSTTR	TSSO
	Reset value																															0	0
0x72C	ETH_PTPPPSCR	Reserved																														PPS FREQ	
	Reset value																															0	0
0x1000	ETH_DMABMR	Reserved					MB	AAB	FPM	USP	RDP						FB	PM		PBL						EDFE	DSL					DA	SR
	Reset value						0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
0x1004	ETH_DMATPDR	TPD																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1008	ETH_DMARPDR	RPD																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x100C	ETH_DMARDLAR	SRL																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1010	ETH_DMATDLAR	STL																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1014	ETH_DMASR	Reserved	TSTS	PMTS	MMCS	Reserved	EBS			TPS			RPS			NIS	AIS	ERS	FBES	Reserved	ETS	RWTS	RPSS	RBUS	RS	TUS	ROS	TJTS	TBUS	TPSS	TS		
	Reset value		0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0		
0x1018	ETH_DMAOMR	Reserved					DTCEFD	RSF	DFRF	Reserved	TSF	FTF	Reserved	TTC			ST	Reserved								FEF	FUGF	Reserved	RTC		OSF	SR	Reserved	
	Reset value						0	0	0		0	0		0	0	0	0									0	0		0	0	0	0		
0x101C	ETH_DMAIER	Reserved															NISE	AISE	ERIE	FBEIE	Reserved	ETIE	RWTIE	RPSIE	RBUIE	RIE	TUIE	ROIE	TJTIE	TBUIE	TPSIE	TIE		
	Reset value																0	0	0	0		0	0	0	0	0	0	0	0	0	0	0		
0x1020	ETH_DMAMFBOCR	Reserved		OFOC	MFA												OMFC	MFC															
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1024	ETH_DMARSWTR	Reserved																								RSWTC							
	Reset value																									0	0	0	0	0	0	0	0
0x1048	ETH_DMACHTDR	HTDAP																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x104C	ETH_DMACHRDR	HRDAP																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1050	ETH_DMACHTBAR	HTBAP																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1054	ETH_DMACHRBAR	HRBAP																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

34. Полноскоростной автономный USB (OTG_FS)

34.1. Введение в OTG_FS

Использованы сокращения:

FS	Полная скорость
LS	Низкая скорость
MAC	Контроллер доступа среды
OTG	On-the-go (здесь "*автономный*")
PFC	Контроллер FIFO пакетов
PHY	Физический уровень
USB	Универсальная последовательная шина
UTMI	USB 2.0 transceiver macrocell interface (UTMI)

OTG_FS это двухфункциональный (DRD) контроллер, поддерживающий режимы периферийного устройства и хоста USB, в соответствии с *On-The-Go Supplement to the USB 2.0 Specification*. Может работать только хостом и только устройством в соответствии с *USB 2.0 Specification*. В режиме хоста OTG_FS поддерживает полноскоростные (FS, 12 Мбит/с) и низкоскоростные (LS, 1.5 Мбит/с) передачи, а в режиме устройства только полноскоростные (FS, 12 Мбит/с) передачи. OTG_FS поддерживает и HNP и SRP. В режиме хоста нужно только внешнее питание V_{BUS}.

34.2. Основные свойства OTG_FS

Их можно разделить на три категории: общие, хоста и устройства.

34.2.1. Общие свойства OTG_FS

Они таковы:

- Интерфейс USB-IF, сертифицированный для *Universal Serial Bus Specification Rev 2.0*
- Полная поддержка (PHY) необязательного протокола OTG см. *On-The-Go Supplement Rev 1.3 specification*
- *On-The-Go Supplement Rev 1.3 specification*

—Встроенная поддержка Идентификации Устройств A-B (линия ID)

—Встроенная поддержка хостом Протокола Сообщений (HNP) и Протокола Запроса Сессии SRP

—Выключение хостом V_{BUS} для экономии батарей систем OTG

—Поддержка контроля OTG уровня V_{BUS} внутренними компараторами

—Поддержка динамического переключения режимов хост/устройство

- Программная конфигурация в режиме:

—Устройство USB FS с SRP (устройство B)

—Хост USB FS с SRP (устройство A)

—Двухфункциональный OTG FS USB

- Поддержка FS SOF и поддержки соединения LS с:

—Импульс SOF от PAD

—Импульс SOF от внутреннего таймера 2 (TIM2)

—Изменяемый период фрейма

—Прерывание конца фрейма

- Включает экономию энергии вроде останова системы во время остановки USB, выключения тактового домена ядра, управления питанием PHY и DFIFO
- Выделенная 1.25 КБ RAM с продвинутым управлением FIFO:

—Программируемое разделение пространства RAM для разных FIFO

—Все FIFO могут содержать много пакетов

—Динамическое выделение памяти

—Размеры FIFO не кратны степени двойки ради неразрывности памяти

- Гарантия максимальной полосы пропускания USB до одного фрейма (1 ms) без вмешательства системы

34.2.2. Свойства хоста OTG_FS

Это:

- Внешнее питание V_{BUS}.
- До 8 каналов хоста с динамической конфигурацией для любого типа передач USB.
- Встроенный аппаратный планировщик, имеющий:

— Аппаратную очередь для 8 прерываний плюс запросов изохронных передач

— Аппаратную очередь для 8 управляющих событий плюс запросов не-периодических групповых передач

- Управление общим RX FIFO, периодическим и не-периодическим TX FIFO.

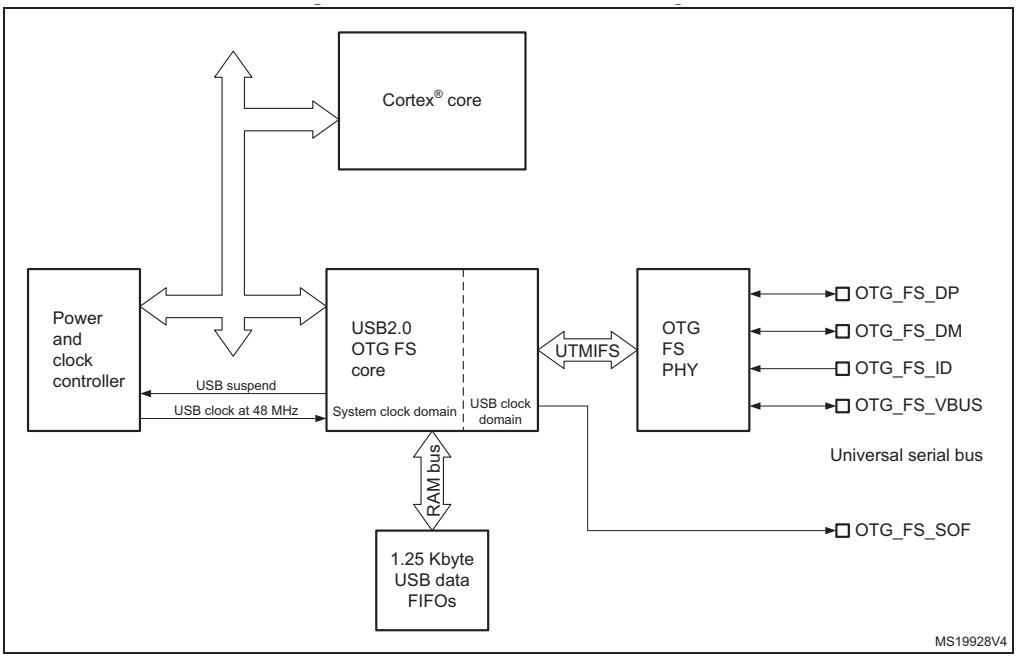
34.2.3. Свойства устройства OTG_FS

Это:

- 1 двунаправленная конечная точка 0
- 3 конечные точки (EP) IN для Групповых, Прерывающих и Изохронных передач
- 3 конечные точки OUT для Групповых, Прерывающих и Изохронных передач
- Управление общим Rx FIFO и Tx-OUT FIFO
- Управление до 4 выделенными Tx-IN FIFO (по одному для каждой активной IN EP)
- Поддержка программного отключения.

34.3. Функциональное описание OTG_FS

Блок-схема OTG_FS



34.3.1. Ножки OTG

Таблица 196. Для чего нужны ноги OTG_FS

Signal name	Signal type	Description
OTG_FS_DP	Digital input/output	USB OTG D+ line
OTG_FS_DM	Digital input/output	USB OTG D- line
OTG_FS_ID	Digital input	USB OTG ID
OTG_FS_VBUS	Analog input	USB OTG VBUS
OTG_FS_SOF	Digital output	USB OTG Start Of Frame (visibility)

34.3.2. Полноскоростное ядро OTG

USB OTG FS получает 48 MHz ±0.25% такты от контроллера сброса и тактов (RCC) через внешний кварц. Такты USB питают 48 MHz домен на полной скорости (12 Мбит/с) и должны включаться до конфигурации ядра OTG FS.

CPU и OTG FS соединены шиной AHB. Для USB OTG выделена одна линия прерываний.

CPU посылает данные через USB записью 32-бит слов по адресам регистров OTG_FS. Они автоматически идут в Tx-FIFO данных в памяти данных USB. Каждой IN EP (устройство) и каждому выходному каналу (хост) выделяется один Tx-FIFO.

CPU получает данные от USB чтением 32-бит слов из адресов регистров OTG_FS. Они автоматически идут из общего Rx-FIFO в 1.25 КБ памяти данных USB. Каждой OUT EP (устройство) и каждому входному каналу (хост) выделяется один регистр Rx-FIFO.

Уровень протокола USB использует последовательный преобразователь (SIE) и с шиной USB общается через встроенный приёмо-передатчик физического уровня (PHY).

34.3.3. Полноскоростной OTG PHY

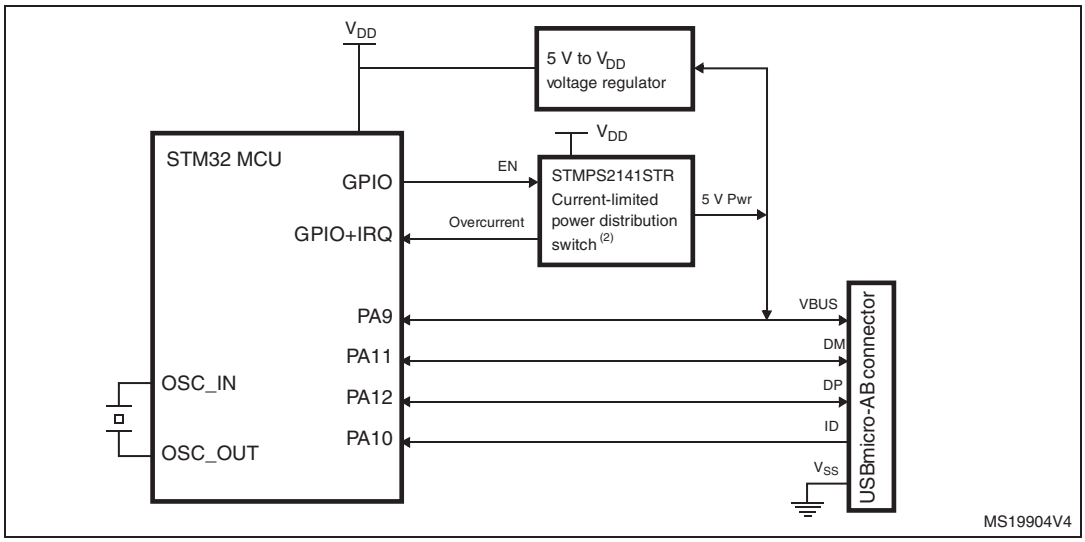
OTG PHY использует подмножество шины UTMI+ Bus (UTMIFS) и включает компоненты:

- Модуль приёмопередатчика FS/LS хоста и устройства.
- Встроенный резистор подпорки линии ID для идентификации устройств A/B.
- Встроенные резисторы подпорки и подтяжки линий DP/DM, управляемые ядром OTG_FS. В режиме устройства к V_{BUS} подключается резистор подпорки DP (полноскоростная периферия). В режиме хоста к обеим линиям DP/DM подключаются резисторы подтяжки. Все резисторы подключаются динамически в соответствии с протоколом HNP.
- Резистор подпорки DP состоит из 2 раздельно подключаемых резисторов. Динамическая подстройка подпорки DP помогает бороться с шумами сигнала Tx/Rx.
- Компараторы уровня V_{BUS} с гистерезисом для определения V_{BUS} в Норме, A-B Сессия в Норме и порогов напряжения конца сессии. Они используются в протоколе запроса сессии (SRP), определяя начало и конец сессии и следят за уровнем V_{BUS}.
- Импульсная схема заряда/разряда V_{BUS} через во время SRP (слабый привод).

NB: Частота шины AHB должна быть выше 14.2 MHz.

34.4. Двухфункциональный OTG_FS (DRD)

Подключение устройства OTG A-B



1. Внешний регулятор напряжения только при питании устройства от V_{BUS}
2. STMPS2141STR на плате системы требуется только если надо подать V_{BUS} на внешнее устройство.

34.4.1. Определение линии ID

Режим хоста или периферии (по умолчанию) подразумевается зависящим от входной ножки ID. Состояние линии ID зависит от стороны кабеля USB, воткнутого в "разъёмаму" micro-AB.

- Если вставлена сторона B с оборванным проводом ID, то встроенный резистор подпорки создаёт высокий уровень ID и это Периферия.
- Если вставляется сторона A с заземлённым ID, то OTG_FS выдаёт прерывание изменения состояния линии ID (бит `CIDSCHG` в `OTG_FS_GINTSTS`) для инициализации программ хоста и автоматически переключается в режим хоста.

34.4.2. Двухфункциональное устройство HNP

Бит разрешения протокола HNP (бит HNPCAP в OTG_FS_GUSBCFG) позволяет ядру OTG_FS динамически переключаться из A-хоста в A-периферию и наоборот, и из B-периферии в B-хост. Текущее состояние устройства можно определить по сочетанию битов состояния ID (бит CIDSTS в OTG_FS_GOTGCTL) текущего режима (бит CMOD в OTG_FS_GINTSTS).

Программная модель HNP описана в *Секции 34.17*.

34.4.3. Двухфункциональное устройство SRP

Бит разрешения протокола SRP (бит SRPCAP в OTG_FS_GUSBCFG) разрешает ядру OTG_FS выключать питание от V_{BUS} для A-устройств, они и так всегда с ним работают.

Программная модель SRP A/B-устройств описана в *Секции 34.17*.

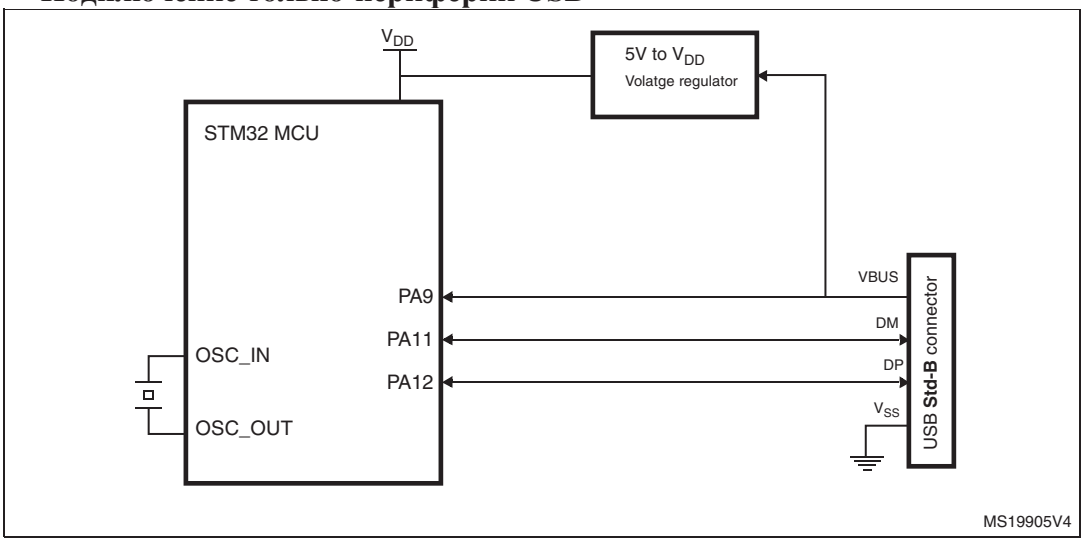
34.5. Периферийный USB

OTG_FS работает периферией USB как:

- OTG B-периферия
  - —OTG B-устройство по умолчанию, если кабель USB воткнут стороной B
- OTG A-периферия
  - —OTG A-устройство после того как HNP переключает OTG_FS в режим периферии
- B-устройство
  - —Если линия ID есть, работает и подключена к стороне B кабеля USB, и очищен бит HNPCAP в регистре OTG_FS_GUSBCFG.
- Только периферия:
  - —Стоит бит FDMOD в регистре OTG_FS_GUSBCFG. В этом случае линия ID игнорируется, даже если она есть в разъёме.

NB: При питании устройств в режимах B и только-периферии от V_{BUS}, V_{DD} надо подавать через внешний регулятор.

Подключение только-периферии USB



34.5.1. Устройство с SRP

Разрешение протокола SRP включается битом SRPCAP в регистре OTG_FS_GUSBCFG. Таким образом устройство A может выключать V_{BUS} при остановке сессии USB.

34.5.2. Состояния устройств

Запитанное

Обнаружение V_{BUS} при B-Сессии делает состояние Запитанным. OTG_FS автоматически подключает резистор подпорки сигнала DP полноскоростного хоста и выдаёт запрос прерывания сессии (бит SRQINT в регистре OTG_FS_GINTSTS).

Также проверяется рабочий уровень V_{BUS} от хоста. Если в B-сессии V_{BUS} падает, то OTG_FS автоматически отключается и выдаёт прерывание выхода из запитанного режима (бит SEDET в регистре OTG_FS_GOTGINT).

В запитанном режиме OTG_FS ждёт сигнала сброса от хоста, и всё. При появлении сброса (USBRST в OTG_FS_GINTSTS) выдаётся прерывание. По завершении сброса выдаётся прерывание конца переучёта (ENUMDNE в OTG_FS_GINTSTS) и OTG_FS идёт в состояние по умолчанию.

Программное отключение

Установка бита SDIS в регистре OTG_FS_DCTL отключает резистор подпорки DP и хост выдаёт прерывание отключения даже при реально воткнутом в порт хоста кабеле USB.

По умолчанию

В этом режиме OTG_FS ждёт от хоста команду SET_ADDRESS. По ней в поле DAD регистра OTG_FS_DCFG пишется адрес устройства, OTG_FS становится адресованным и может отвечать на запросы хоста.

Остановленное

Периферийный OTG_FS постоянно отслеживает активность USB. После 3 ms простоя USB выдаётся прерывание раннего останова (бит ESUSP в OTG_FS_GINTSTS) и через 3 ms подтверждается прерыванием останова (бит USBSUSP в OTG_FS_GINTSTS). Автоматически ставится бит состояния (SUSPSTS в OTG_FS_DSTS) и OTG_FS стопорится.

Выйти из останова можно самостоятельно установкой бита удалённой побудки (RWUSIG в OTG_FS_DCTL) и его снятием после задержки от 1 до 15 ms.

При появлении сигнала побудки от хоста выдаётся прерывание (бит WKUPINT в регистре OTG_FS_GINTSTS) и автоматически снимается бит останова устройства.

34.5.3. Конечные точки устройств

Ядро OTG_FS обслуживает следующие конечные точки USB:

- Управляющая точка 0:
  - Двунаправленная, только для управляющих сообщений.
  - Отдельные наборы регистров для входных и выходных передач.
  - Собственные регистры управления (OTG_FS_DIEPCTL0/OTG_FS_DOEPCTL0), конфигурации передач (OTG_FS_DIEPTSIZ0/OTG_FS_DIEPTSIZ0) и состояния/прерываний (OTG_FS_DIEPINT0/OTG_FS_DOEPINT0) с несколько отличными от других точек битами.
- 3 конечных точки IN
  - Поддерживают изохронные, групповые и прерывающие передачи.
  - Собственные регистры управления (OTG_FS_DIEPCTL*x*), конфигурации передач (OTG_FS_DIEPTSIZ*x*) и состояния/прерываний (OTG_FS_DIEPINT*x*).
  - Регистр маски общих прерываний (включая EP0) Устройства IN (OTG_FS_DIEPMSK).
  - Поддержка прерывания незавершённых изохронных передач IN (бит IISOIXFR в регистре OTG_FS_GINTSTS). Оно выставляется во время прерывания конца периодического фрейма (OTG_FS_GINTSTS/EOPF).
- 3 конечных точки OUT
  - Поддерживают изохронные, групповые и прерывающие передачи.
  - Собственные регистры управления (OTG_FS_DOEPCTL*x*), конфигурации передач (OTG_FS_DOEPTSIZ*x*) и состояния/прерываний (OTG_FS_DOEPINT*x*).
  - Регистр маски общих прерываний (включая EP0) Устройства Out (OTG_FS_DOEPMSK).
  - Поддержка прерывания незавершённых изохронных передач OUT (бит INCOMPISOOUT в OTG_FS_GINTSTS). Оно выставляется во время прерывания конца периодического фрейма (OTG_FS_GINTSTS/EOPF).

Управление конечной точкой

- Через регистры управления (DIEPCTLx/DOEPCTLx) можно:
  - Включить/выключить точку
  - Активировать точку в текущей конфигурации
  - Установить тип передачи USB (изохронная, групповая, прерывающая)

— Задать размер пакета
— Задать номер Tx-FIFO для точки IN
— Задать ожидаемый или передаваемый DATA0/DATA1 PID (только групповые/прерывающие)
— Задать чётный/нечётный фрейм изохронных передач
— Установкой бита NAK всегда выдавать хосту не-ответ независимо от состояния FIFO
— Установкой бита STALL всегда стопорить токены хоста этой точке
— Установить режим SNOOP точки OUT, чтобы не проверяла CRC принятых данных.

Передачи конечной точки

Размер и состояние передачи хранятся в регистрах DIEPTSIZx/DOEPTSIZx. Писать в него нужно до запуска конечной точки. После запуска точки регистр доступен только по чтению, пишет в него OTG FS.

Можно установить параметры:

- Размер передачи в байтах
- Число пакетов всей передачи

Состояние/прерывания конечной точки

События со стороны USB и AHB хранятся в регистрах прерываний (DIEPINTx/DOPEPINTx). Читать их надо при стоящих битах прерываний (бит OEPINT в OTG_FS_GINTSTS для точки OUT и бит IEPINT в OTG_FS_GINTSTS для точки IN). Перед чтением этих регистров надо получить номер прервавшей конечной точки из общего регистра прерываний (OTG_FS_DAINT). Снятие бита в этих регистрах чистит соответствующие биты регистров DAINT и GINTSTS.

Ядро периферии проверяет и выдаёт прерывания для:

- Завершение передачи для сторон AHB и USB
- Фаза SETUP завершена (только управляющие OUT)
- FIFO передачи наполовину или полностью пуст (точки IN)
- Хосту передан ответ NAK (только изохронные IN)
- Принят токен IN при пустом Tx-FIFO (только групповые IN и прерывающие IN)
- Принят токен OUT для ещё не включённой точки
- Ошибка перекрёстного шума
- Программа выключила точку
- Программа включила NAK точки (только изохронные IN)
- Принята череда более 3 пакетов SETUP (только управляющая OUT)
- Таймаут (только управляющие IN)
- Отброшен изохронный пакет OUT без выдачи прерывания.

34.6. Хост USB

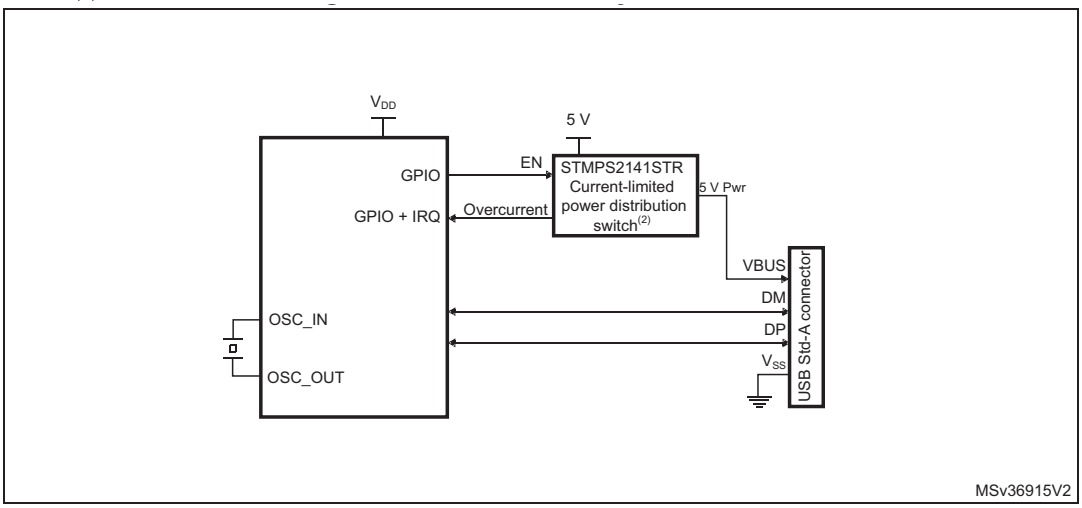
OTG_FS работает USB хостом как:

• OTG A-хост
– состояние A-устройства по умолчанию когда вставлена сторона A кабеля USB
• OTG B-host
– B-устройство после HNP переключения в режим хоста
• A-устройство
– Если линия ID есть, работает и подключена к A-стороне кабеля USB, и снят бит HNPCAP в регистре OTG_FS_GUSBCFG. К линиям DP/DM автоматически подключаются встроенные резисторы подтяжки.
• Только хост
– Включается битом FHMOD в регистре OTG_FS_GUSBCFG. В этом случае линия ID игнорируется даже при её наличии в разъёме USB. К линиям DP/DM автоматически подключаются встроенные резисторы подтяжки.

NB: Встроенный источник 5V V_{BUS} не поддерживается, на линию надо подавать внешнее напряжение. Внешнюю подзарядку можно подавать через выход GPIO. Это требуется для конфигураций OTG A-хост, A-устройства и только-хост.

Уровень V_{BUS} при работе USB проверяется на входе V_{BUS}. Сигнал перегрузки источника подзарядки V_{BUS} можно подавать на вход GPIO для выдачи прерывания. Прерывание перегрузки должно немедленно выключать V_{BUS}.

Подключение только-хост USB



1. STMPS2141STR нужна только при питании устройств от V_{BUS}. При наличии 5V питания на плате можно использовать перемычку.
2. V_{DD} между 2 V и 3.6 V

34.6.1. Хост с SRP

Поддержка SRP включается битом SRPCAP в регистре OTG_FS_GUSBCFG. Теперь можно выключать питание V_{BUS} при остановке сессии USB.

34.6.2. Состояния хоста USB

Питание порта хоста

Встроенное питание 5V V_{BUS} не поддерживается. Стало быть нужно использовать внешнее. Внешнюю подзарядку можно подавать через выходы GPIO. При питании V_{BUS} от GPIO надо ставить бит PPWR в регистре OTG_FS_HPRT.

V_{BUS} в норме

При включённом HNP или SRP ножку контроля V_{BUS} (PA9) надо подключать к V_{BUS}. Непредвиденное падение напряжения V_{BUS} ниже порога (4.25 V) запускается прерывание конца сессии (бит SEDET в OTG_FS_GOTGINT). Затем надо отключить V_{BUS} и снять бит питания порта.

При выключенных HNP SRP ножку контроля V_{BUS} (PA9) к V_{BUS} подключать не надо.

Сигнал перегрузки источника подзарядки V_{BUS} можно подавать на вход GPIO для выдачи прерывания. Прерывание перегрузки должно немедленно выключать V_{BUS} и снять бит питания порта.

Обнаружение хостом подключения периферии

При включённых SRP или HNP ядро OTG_FS обнаруживает подключение периферии USB или устройства B только при рабочем уровне питания (V_{BUS} выше 4.75 V). При подключении внешнего устройства B ядро OTG_FS выдаёт прерывание установкой бита PCDET в регистре OTG_FS_HPRT.

При выключенных HNP и SRP периферия USB и устройства B обнаруживаются сразу после подключения. Ядро OTG_FS выдаёт прерывание установкой бита PCDET в регистре OTG_FS_HPRT.

Обнаружение хостом отключения периферии

Прерывание выдаётся установкой бита DISCINT в регистре OTG_FS_GINTSTS.

Переучёт хоста

После обнаружения подключённого устройства хост должен начать переучёт, посылая новому устройству сброс USB и команды конфигурации.

Перед посылкой сброса USB надо дождаться прерывания конца антидребезга (бит DBCDNE в OTG_FS_GOTGINT), появляющегося при подключении резисторов к DP (FS) или DM (LS).

Сигнал сброса USB подаётся снятием бита PRST в регистре OTG_FS_HPRT минимум на 10 ms и максимум на 20 ms. Считать это время надо осторожно.

После сброса USB выдаётся прерывание изменения состояния включения (бит PENCHNG в OTG_FS_HPRT). Теперь можно узнать скорость подключённого устройства (бит PSPD в OTG_FS_HPRT) и начать выдавать SOF (FS) или Живи (Keep alive) (LS) и посылать команды конфигурации.

Остановка хоста

Активность USB останавливают установкой бита PSUSP в регистре OTG_FS_HPRT. OTG_FS прекращает посылку SOFs и идёт в режим останова. Выйти из него можно по инициативе устройства. При этом выдаётся прерывание побудки (бит WKUPINT в регистре OTG_FS_GINTSTS), автоматически ставится бит восстановления (бит PRES в OTG_FS_HPRT) и на USB выдаётся сигнал восстановления. Через некоторое время бит восстановления снимается и и возобновляется выдача SOF.

При побудке по инициативе хоста автоматически ставится бит восстановления через некоторое время он снимается и возобновляется выдача SOF.

34.6.3. Каналы хоста

Ядро OTG_FS обслуживает 8 каналов хоста. Хост не может одновременно поддерживать больше 8 запросов передач. Если их больше, то контроллер хоста (HCD) должен перераспределить каналы после их освобождения завершением передачи.

Канал хоста поддерживает IN/OUT и любой тип периодических/не-периодических передач. Канал имеет собственные регистры управления (HCCHAR*x*), конфигурации (HCTSIZ*x*) состояния/прерываний (HCINT*x*) со связанным регистром маски (HCINTMSK*x*).

Управление каналом хоста

- Регистр HCCHAR*x* позволяет:

— Включать/выключать канал.

— Ставить скорость FS/LS целевой периферии USB

— Задавать адрес целевой периферии USB

— Задавать номер конечной точки целевой периферии USB

— Задавать направление передач IN/OUT

— Задавать тип передач USB (управляющая, групповая, прерывающая, изохронная)

— Задавать максимальный размер пакета (MPS)

— Задавать периодические передачи для выполнения во время чётных/нечётных фреймов

Передачи канала хоста

Длина и состояние передач содержатся в регистре HCTSIZ*x*. Писать в него надо до запуска канала. После запуска регистр доступен только по чтению.

- Можно задавать:

— размер передачи в байтах

— число пакетов всей передачи

— начальный PID данных

Состояние/прерывания канала хоста

События со стороны USB и AHB отмечаются в регистре HCINT*x*. Читать его нужно при стоящем бите прерывания (бит HCINT в регистре OTG_FS_GINTSTS). Перед этим нужно прочесть номер прервавшего канала из регистра HCAINT. Снимаются биты прерывания очисткой соответствующих битов в регистрах HAINT и GINTSTS. Маска прерываний есть в регистре OTG_FS_HCINTMSK*x*.

- Ядро хоста проверяет следующие события прерываний:

— Завершение передачи на сторонах AHB и USB

— Остановка канала по концу передачи, ошибке передачи USB или команде выключения

— Передающий FIFO наполовину или полностью пуст (точки IN)

— Принят ответ ACK

— Принят ответ NAK

— Принят ответ STALL

— Ошибка передачи USB по ошибке CRC, таймауту, битовому заполнению, сбойному EOP
— Ошибка перекрёстных шумов
— Переполнение фрейма
— Ошибка переключения данных

34.6.4. Планировщик хоста

По началу фрейма хост сначала выполняет одновременно затребованные периодические (изохронные и прерывающие) передачи, а затем не-периодические (управляющие и групповые). Это делается с помощью очередей запросов (по одной для каждого вида передач). Каждая очередь имеет до 8 элементов. Элемент очереди содержит номер IN или OUT канала и информацию к его обработке через USB. Порядок запросов в очереди определяет последовательность пересылок по USB.

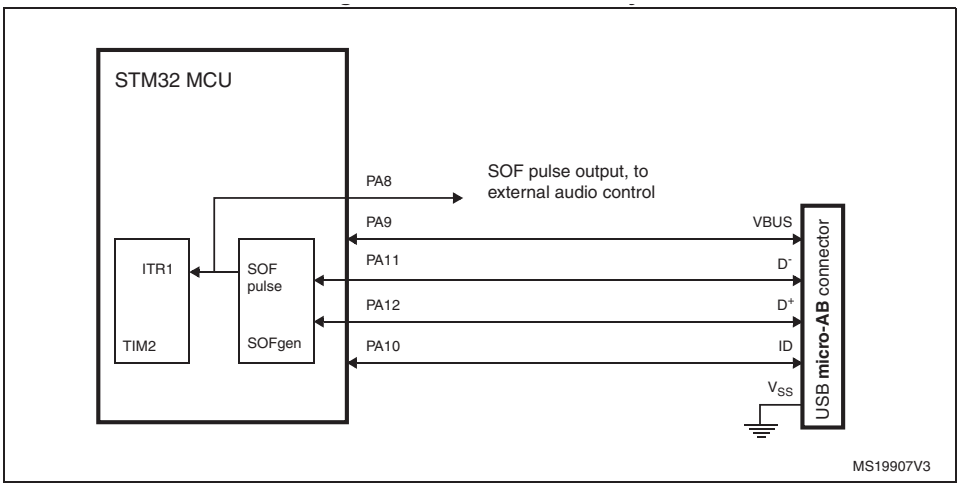
Если для текущего фрейма ещё запланирована изохронная или прерывающая передача, то в его конце выдаётся прерывание незавершённой периодической передачи (IPXFR в OTG_FS_GINTSTS). Управляет очередями запросов только ядро OTG HS. Регистры FIFO и состояния периодической (HPTXSTS) и не-периодической (HNPTXSTS) очередями доступны только по чтению. Они содержат:

- Число свободных строк (до 8)
- Доступное место в Tx-FIFO (передачи OUT)
- Токен IN/OUT, номер канала и состояние.

Перед отправкой запроса в очередь нужно убедиться в наличии там свободного места проверив биты PTXQSAV в регистре OTG_FS_HNPTXSTS или NPTQXSAV в регистре OTG_FS_HNPTXSTS.

34.7. Выдача SOF

Подключение SOF



Ядро OTG FS помогает отслеживать и конфигурировать размещение SOF в хосте и периферии, равно как и выдачу импульсов SOF. Это полезно при подключении звуковой периферии для синхронизации изохронного потока между устройством и PC.

34.7.1. SOF хоста

В режиме хоста число тактов PHY между двумя последовательными токенами SOF (FS) или Keep-alive (LS) пишутся в регистр интервала фреймов (HFIR). Прерывание выдаётся по каждому началу фрейма (бит SOF в OTH_FS_GINTSTS). Номер текущего фрейма и остаток времени до следующего отслеживается в регистре HFNUM.

Выдаваемый по токену SOF импульс, длительностью 12 системных тактов можно направлять на выходную ножку SOF с помощью бита SOFOUTEN в регистре OTG_FS_GCCFG. Также импульс SOF внутренне подключён к входу запуска TIM2.

34.7.2. SOF устройства

В режиме устройства при получении токена SOF (бит SOF в OTH_FS_GINTSTS) выдаётся прерывание. Номер фрейма можно почитать в регистре состояния устройства (FNSOF в регистре OTG_FS_DSTS). Выдаваемый по токену SOF импульс, длительностью 12 системных тактов можно направлять на выходную ножку SOF с помощью бита SOFOUTEN в регистре OTG_FS_GCCFG. Также импульс SOF внутренне подключён к входу запуска TIM2.

Время выдачи прерывания конца периодического фрейма (GINTSTS/EOPF) определено как 80%, 85%, 90% или 95% ожидаемого времени фрейма в регистре конфигурации устройства (PFIVL в OTG_FS_DCFG). Так можно определить конец всей изохронной передачи.

34.8. Экономные режимы OTG

Таблица 197. Совместимость экономных режимов STM32 с OTG

Режим	Описание	Совместимость с USB
Run	MCU совсем работает	Нужна при не остановленном хосте USB not in.
Sleep	Выход USB из останова выводит устройство из Sleep. Регистры периферии сохраняются.	Доступно при остановленном хосте USB.
Stop	Выход USB из останова выводит устройство из Stop. Регистры периферии сохраняются(1).	Доступно при остановленном хосте USB.
Standby	Выкл. питания. Периферию надо инициализировать.	Не совместимо с работой USB

Потребление энергии OTG PHY определяется тремя битами регистра конфигурации ядра:

- Выключение PHY (GCCFG/PWRDWN)
- Включение компараторов V_{BUS} устройства A (GCCFG/VBUSASEN).
- Включение компараторов V_{BUS} устройства B (GCCFG/VBUSASEN)

У остановленного USB экономить можно так:

- Остановка тактов PHY (бит STPPCLK в OTG_FS_PCGCCTL).
- Запрет HCLK (GATEHCLK в OTG_FS_PCGCCTL).
- Стоп системы USB.

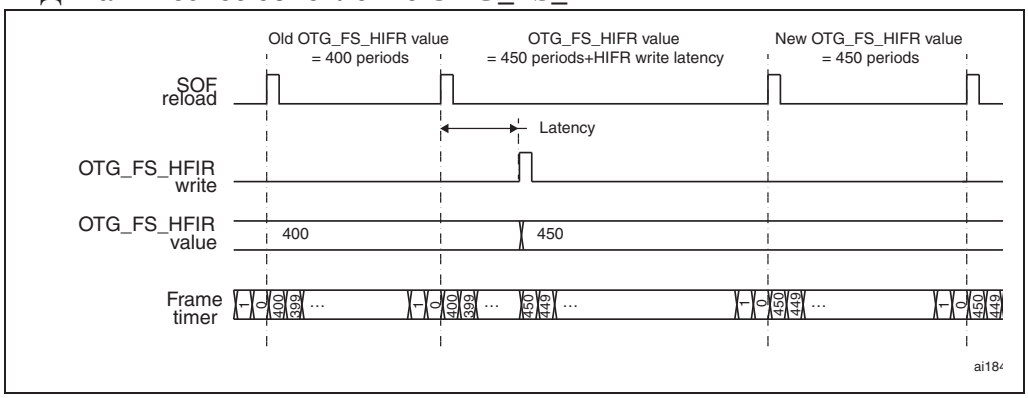
Во благо экономии, FIFO данных USB тактируются только при доступе из ядра OTG_FS.

34.9. Динамическое обновление регистра OTG_FS_HFIR

Ядро USB в режиме хоста может подстраивать время выдачи фреймов микро-SOF для синхронизации внешнего устройства.

Если регистр OTG_HS_HFIR изменился внутри текущего фрейма микро-SOF, то изменение периода SOF вступит в силу в следующем фрейме.

Динамическое обновление OTG_FS_HFIR

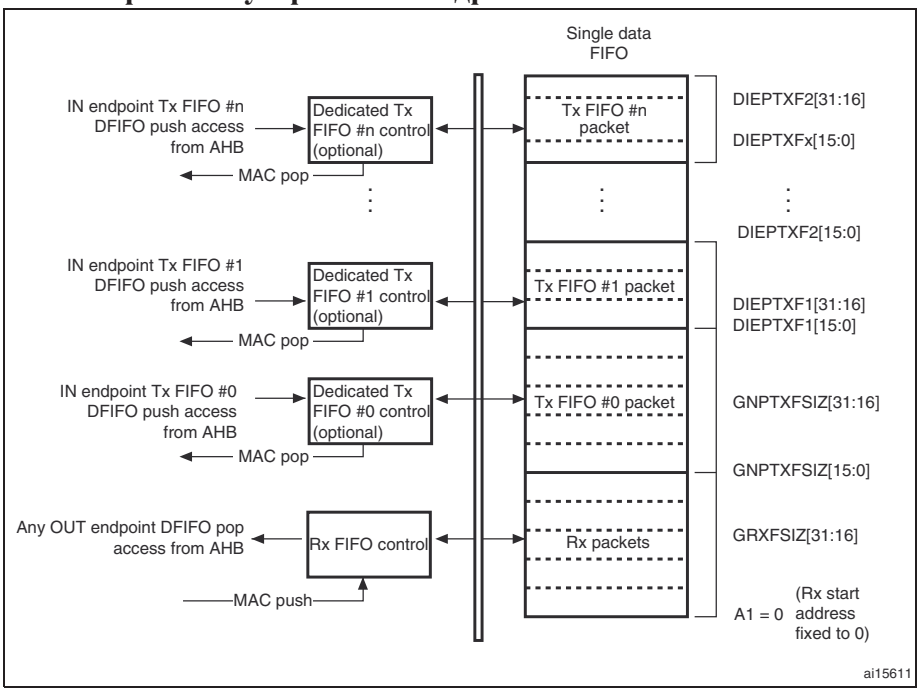


34.10. FIFO данных USB

Система USB имеет 1.25 КБ выделенного RAM для одного Rx-FIFO и нескольких Tx-FIFO. Число и использование FIFO зависит от режима устройства. В режиме периферии каждой активной точке IN выделяется дополнительный Tx-FIFO. Размер FIFO определяется программно.

34.11. Архитектура FIFO устройства

FIFO режима устройства и адреса AHB



34.11.1. Rx FIFO устройства

Он принимает данные в доступное место Rx-FIFO от всех точек OUT устройства. Состояние принятого (номер точки OUT, счётчик байт, PID данных и их достоверность) хранится вместе с данными. При исчерпании места в FIFO передача от хоста получает NACK и выдаётся прерывание адресованной конечной точки. Размер приёмного FIFO задаётся в регистре GRXFSIZ.

Выгоды одного FIFO приёма:

- Все точки OUT используют один буфер RAM (общий FIFO)
- Токены OUT могут поступать в любой последовательности

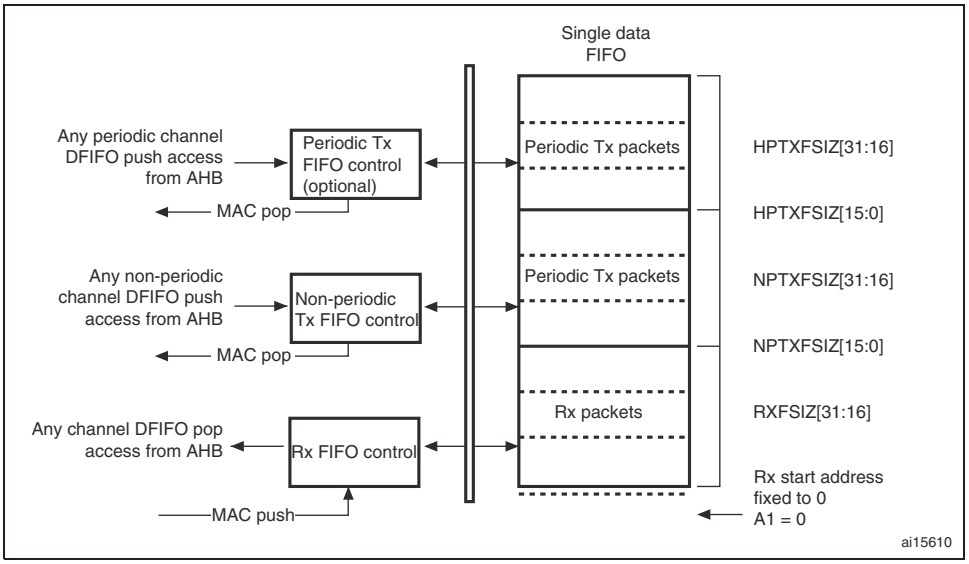
Если Rx-FIFO не пуст (бит RXFLVL в OTG_FS_GINTSTS) то программа получает прерывания. Информация пакета есть в регистре состояния пакета (GRXSTSP), данные читают по соответствующему адресу FIFO.

34.11.2. Tx FIFO устройства

Для каждой точки IN выделяется свой FIFO. Размеры FIFO пишутся в регистры размера FIFO, (OTG_FS_TX0FSIZ) для точки IN-0 и (DIEPTXFx) для точек IN-x.

34.12. Архитектура FIFO хоста

FIFO режима хоста и адреса AHB



34.12.1. Rx FIFO хоста

Он принимает данные в доступное место Rx-FIFO все периодические и не-периодические передачи от всех точек IN устройства. Состояние принятого (номер точки OUT, счётчик байт, PID данных и их достоверность) хранится вместе с данными. Размер приёмного FIFO задаётся в регистре GRXFSIZ.

Выгоды одного FIFO приёма:

- Все точки OUT используют один буфер RAM (общий FIFO)
- Запись любой последовательности токенов IN определяется программой хоста

Если Rx-FIFO не пуст, то программа получает прерывания. Информация пакета есть в регистре состояния пакета, данные читают по соответствующему адресу FIFO.

34.12.2. Tx FIFO хоста

Для периодических (изохронных и прерывающих) и не-периодических (управляющих и групповых) передач OUT используются отдельные FIFO. В них хранятся передаваемые данные. Размер периодического/не-периодического Tx FIFO определяется каждый в своём регистре (HPTXFSIZ/HNPTXFSIZ).

В начале фрейма сначала обрабатывается очередь периодических Tx FIFO, затем не-периодических.

Передающие FIFO работают так:

- Для периодических (не-периодических) каналов OUT используется один буфер RAM
- Последовательность заполнения передающего FIFO определяется программой

Выдача прерываний полупустого или пустого периодического Tx FIFO (PTXFE в OTG_FS_GINTSTS) задаётся в регистре конфигурации AHB (PTXFELVL в OTG_FS_GAHBCFG). Запись передаваемых данных возможна только при наличии свободного места в Tx FIFO и очереди запросов (регистр HPTXSTS).

Выдача прерываний полупустого или пустого не-периодического Tx FIFO (NPTXFE в регистре OTG_FS_GINTSTS) задаётся в регистре конфигурации AHB (TXFELVL в OTG_FS_GAHBCFG). Запись передаваемых данных возможна только при наличии свободного места в Tx FIFO и очереди запросов (регистр HNPTXSTS).

34.13. Размещение FIFO в RAM

34.13.1. Режим устройства

Память для приёмного FIFO: надо выделить память для 10 пакетов SETUP, ядро к этой памяти не суётся. Один участок выделяется для Глобального OUT NAK. Состояние принятого пакета пишется вместе с данными.

Поэтому для приёма пакетов надо выделять минимум (Максимальный_ Размер_Пакета/ 4) + 1. При нескольких изохронных точках надо выделить не меньше двух таких участков. Обычно выделяют два таких участка.

Вместе с последним пакетом передачи конечной точки в FIFO пихается состояние завершения передачи. Для неё нужен один участок на точку OUT.

Память для передающего FIFO: минимальный размер FIFO равен максимальному размеру пакета точки IN.

NB: Больший размер FIFO точки IN повышает производительность USB.

34.13.2. Режим хоста

Память для приёмного FIFO

Состояние принятого пакета пишется вместе с данными.

Поэтому для приёма пакетов надо выделять минимум (Максимальный_ Размер_Пакета/ 4) + 1. При нескольких изохронных точках надо выделить не меньше двух таких участков. Обычно выделяют два таких участка.

Вместе с последним пакетом передачи конечной точки в FIFO пихается состояние завершения передачи. Для неё нужен один участок на канал.

Память для передающего FIFO

Минимальный размер не-периодического передающего FIFO равен самому большому размеру пакетов не-периодических каналов OUT. Обычно выделяют два максимальных размера пакета.

Минимальный размер периодического передающего FIFO равен самому большому размеру пакетов периодических каналов OUT. Если есть изохронные точки OUT, то максимальный размер этого канала надо удвоить.

NB: Больший размер не-периодического передающего FIFO повышает производительность USB.

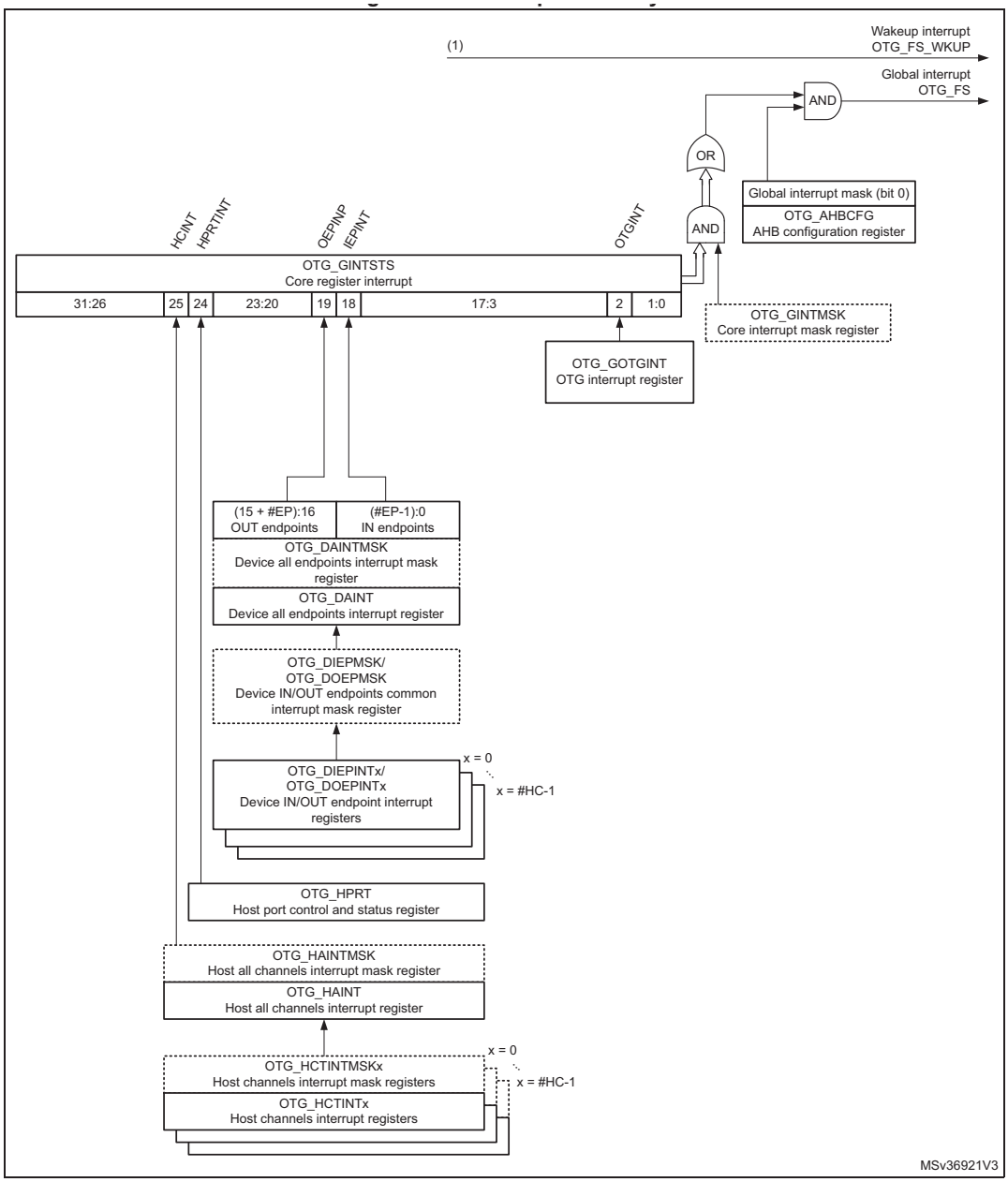
34.14. Производительность системы с USB

Наилучшая производительность USB и системы достигается благодаря большим буферам RAM, гибким размерам FIFO, быстрому 32-бит доступу к FIFO через регистры AHB и развитому механизму управления FIFO. Можно достичь максимальной скорости USB. Качества таковы:

- Программе очень удобно:
  - —Накапливать много данных для передачи по USB
  - —Собирать много данных в приёмный FIFO
- Ядру USB удобно:
  - —Аппаратно планировать передачу большого числа данных по USB
  - —Иметь много места для размещения данных из USB

34.15.Прерывания OTG_FS

Иерархия прерываний



34.16. Регистры управления и состояния OTG_FS

Контроллер OTG_FS управляется через регистры состояния и управления (CSR) по шине AHB. Это 32-бит регистры, выравненные на границу 32-бит, доступные 32-бит словами.

CSR разделяются на:

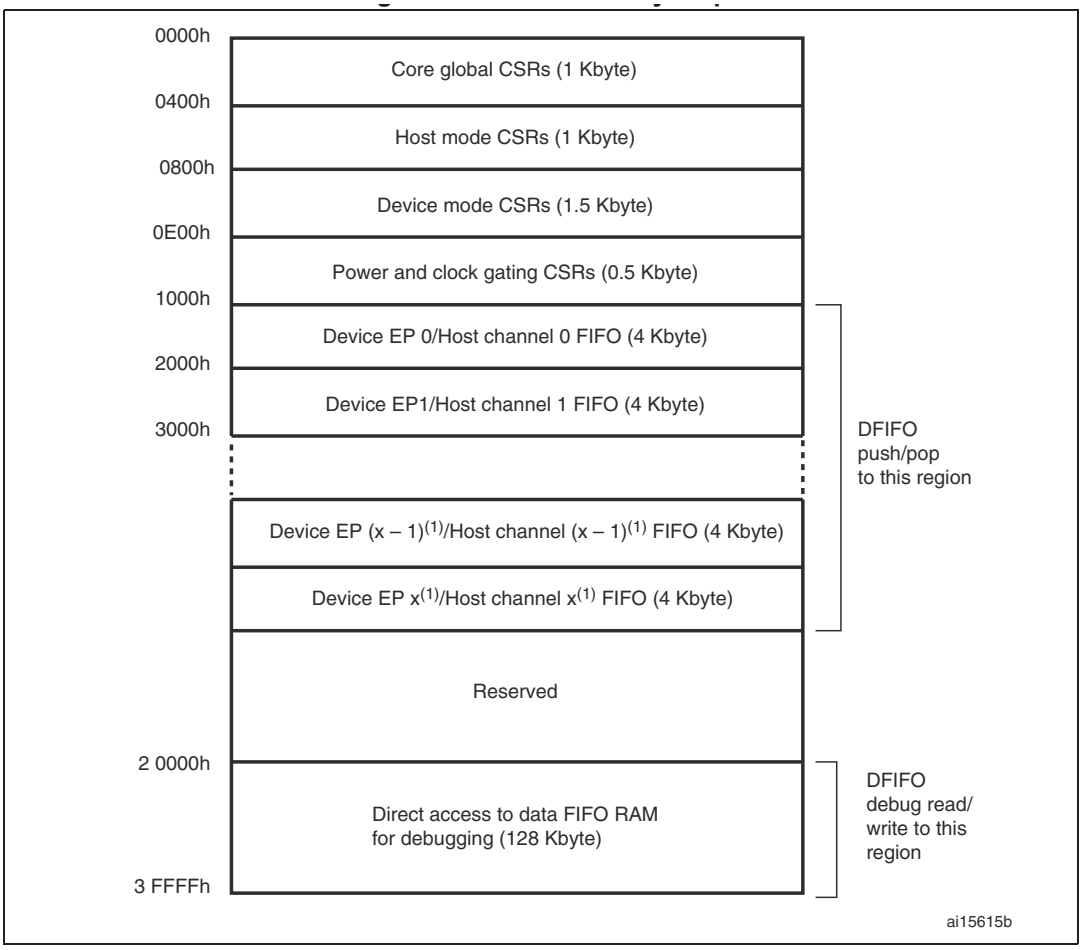
- Общие регистры ядра
- Регистры режима хоста
- Общие регистры хоста
- CSR портов хоста
- Регистры каналов хоста
- Регистры режима устройства
- Общие регистры устройства
- Регистры конечных точек устройства
- Регистры управления питанием и тактами
- Регистры доступа к FIFO данных (DFIFO)

Одновременно в режимах хоста и ядра доступны только Общие регистры ядра, Регистры управления питанием и тактами, Регистры доступа к FIFO данных и CSR портов хоста. Работая в одном режиме, контроллер OTG_FS не должен обращаться к регистрам другого режима. Это вызовет прерывание несовпадения режима (бит `MMIS` в регистре `OTG_FS_GINTSTS`). При переключении режимов регистры нужно переписать как при выходе из сброса.

34.16.1. Карта памяти CSR

Регистры режимов хоста и устройства занимают разные адреса. Все регистры сделаны в домене тактов AHB.

Карта памяти CSR



1. x=3 у устройства и x=7 у хоста.

Карта общих CSR

Эти регистры доступны в режимах хоста и устройства.

Таблица 198. Общие CSR

Имя	Смещение адреса	Имя регистра
OTG_FS_GOTGCTL	0x000	Регистр управления и состояния OTG_FS (OTG_FS_GOTGCTL)
OTG_FS_GOTGINT	0x004	Регистр прерываний OTG_FS (OTG_FS_GOTGINT)
OTG_FS_GAHBCFG	0x008	Регистр конфигурации AHB OTG_FS (OTG_FS_GAHBCFG)
OTG_FS_GUSBCFG	0x00C	Регистр конфигурации USB OTG_FS (OTG_FS_GUSBCFG)
OTG_FS_GRSTCTL	0x010	Регистр сброса OTG_FS (OTG_FS_GRSTCTL)
OTG_FS_GINTSTS	0x014	Регистр прерываний ядра OTG_FS (OTG_FS_GINTSTS)
OTG_FS_GINTMSK	0x018	Регистр маски прерываний OTG_FS (OTG_FS_GINTMSK)
OTG_FS_GRXSTSR	0x01C	Регистр состояния приёма отладки OTG_FS/ Регистр состояния и чтения OTG (OTG_FS_GRXSTSR/OTG_FS_GRXSTSP)
OTG_FS_GRXSTSP	0x020	
OTG_FS_GRXFSIZ	0x024	Регистр размера приёмного FIFO OTG_FS (OTG_FS_GRXFSIZ)
OTG_FS_HNPTXFSIZ/ OTG_FS_DIEPTXF0(1)	0x028	Регистр размера FIFO не-периодической передачи хоста OTG_FS (OTG_FS_HNPTXFSIZ)/размера FIFO передачи конечной точки 0 (OTG_FS_DIEPTXF0)
OTG_FS_HNPTXSTS	0x02C	Регистр FIFO не-периодической передачи /состояния очереди OTG_FS (OTG_FS_HNPTXSTS)
OTG_FS_GCCFG	0x038	Общий регистр конфигурации ядра OTG_FS (OTG_FS_GCCFG)
OTG_FS_CID	0x03C	Регистр ID ядра OTG_FS (OTG_FS_CID)
OTG_FS_HPTXFSIZ	0x100	Регистр размера FIFO периодических передач OTG_FS (OTG_FS_HPTXFSIZ)
OTG_FS_DIEPTXFx	0x104 0x108 0x10C	Регистр размера FIFO передач конечной точки IN устройства OTG_FS (OTG_FS_DIEPTXFx) (x = 1..3, где x это номер FIFO)

1. Общее правило: OTG_FS_HNPTXFSIZ в режиме хоста, OTG_FS_DIEPTXF0 в режиме устройства.

Карта CSR хоста

Эти регистры писать при каждом входе в режим хоста.

Таблица 199. CSR режима хоста

Имя	Смещение адреса	Имя регистра
OTG_FS_HCFG	0x400	Регистр конфигурации хоста OTG_FS (OTG_FS_HCFG)
OTG_FS_HFIR	0x404	Регистр интервала фрейма хоста OTG_FS (OTG_FS_HFIR)
OTG_FS_HFNUM	0x408	Регистр номера фрейма/остатка времени хоста OTG_FS (OTG_FS_HFNUM)
OTG_FS_HPTXSTS	0x410	Регистр FIFO периодических передач/состояния очереди хоста OTG_FS (OTG_FS_HPTXSTS)
OTG_FS_HAINT	0x414	Регистр прерываний всех каналов хоста OTG_FS (OTG_FS_HAINT)
OTG_FS_HAINTMSK	0x418	Регистр маски прерываний всех каналов хоста OTG_FS (OTG_FS_HAINTMSK)
OTG_FS_HPRT	0x440	Регистр управления и состояния хоста OTG_FS (OTG_FS_HPRT)
OTG_FS_HCCHARx	0x500 0x520 ... 0x6E0	Регистр характеристик канала x хоста OTG_FS (OTG_FS_HCCHARx) (x = 0..7, где x = Номер канала)
OTG_FS_HCINTx	0x508	Регистр прерываний канала x хоста OTG_FS(OTG_FS_HCINTx) (x = 0..7, где x = Номер канала)
OTG_FS_HCINTMSKx	0x50C	Регистр маски прерываний канала x хоста OTG_FS Host (OTG_FS_HCINTMSKx) (x = 0..7, где x = Номер канала)
OTG_FS_HCTSIZx	0x510	Регистр размера передачи канала x хоста OTG_FS (OTG_FS_HCTSIZx) (x = 0..7, где x = Номер канала)

Карта CSR устройства

Эти регистры писать при каждом входе в режим устройства.

Таблица 200. CSR режима устройства

Имя	Смещение адреса	Имя регистра
OTG_FS_DCFG	0x800	Регистр конфигурации устройства OTG_FS (OTG_FS_DCFG)
OTG_FS_DCTL	0x804	Регистр управления устройства OTG_FS (OTG_FS_DCTL)
OTG_FS_DSTS	0x808	Регистр состояния устройства OTG_FS (OTG_FS_DSTS)
OTG_FS_DIEPMSK	0x810	Общий регистр маски прерывания конечных точек IN устройства OTG_FS (OTG_FS_DIEPMSK)
OTG_FS_DOEPMSK	0x814	Общий регистр маски прерывания конечных точек OUT устройства OTG_FS (OTG_FS_DOEPMSK)
OTG_FS_DAINT	0x818	Регистр прерываний всех конечных точек устройства OTG_FS (OTG_FS_DAINT)
OTG_FS_DAINTMSK	0x81C	Регистр маски прерываний всех конечных точек устройства OTG_FS (OTG_FS_DAINTMSK)
OTG_FS_DVBUSDIS	0x828	Регистр времени разряда V_{BUS} устройства OTG_FS (OTG_FS_DVBUSDIS)
OTG_FS_DVBUSPULSE	0x82C	Регистр времени импульса V_{BUS} устройства OTG_FS (OTG_FS_DVBUSPULSE)
OTG_FS_DIEPEMPMSK	0x834	Регистр маски пустого FIFO конечных точек IN устройства OTG_FS(OTG_FS_DIEPEMPMSK)
OTG_FS_DIEPCTL0	0x900	Регистр управления IN конечной точки 0 устройства OTG_FS (OTG_FS_DIEPCTL0)
OTG_FS_DIEPCTLx	0x920 0x940 0x960	Регистр управления конечной точки x устройства OTG (OTG_FS_DIEPCTLx) (x = 1..3, где x = Номер конечной точки)
OTG_FS_DIEPINTx	0x908	Регистр прерываний конечной точки x устройства OTG_FS (OTG_FS_DIEPINTx) (x = 0..3, где x = Номер конечной точки)
OTG_FS_DIEPTSIZ0	0x910	Регистр размера передачи IN конечной точки 0 OTG_FS (OTG_FS_DIEPTSIZ0)
OTG_FS_DTXFSTSx	0x918	Регистр состояния FIFO передачи конечной точки IN OTG_FS (OTG_FS_DTXFSTSx) (x = 0..3, где x = Номер конечной точки)
OTG_FS_DIEPTSIZx	0x930 0x950 0x970	Регистр размера передачи OUT конечной точки x устройства OTG_FS (OTG_FS_DOEPTSIZx) (x = 1..3, где x = Номер конечной точки)
OTG_FS_DOEPCTL0	0xB00	Регистр управления OUT конечной точки 0 устройства OTG_FS (OTG_FS_DOEPCTL0)
OTG_FS_DOEPCTLx	0xB20 0xB40 0xB60	Регистр OTG управления конечной точки x устройства (OTG_FS_DIEPCTLx) (x = 1..3, где x = Номер конечной точки)
OTG_FS_DOEPINTx	0xB08	Регистр прерываний конечной точки x устройства OTG_FS (OTG_FS_DOEPINTx) (x = 0..3, где x = Номер конечной точки)
OTG_FS_DOEPTSIZ0	0xB10	Регистр размера передачи OUT конечной точки 0 устройства OTG_FS
OTG_FS_DOEPTSIZx	0xB30 0xB50 0xB70	Регистр размера передачи OUT конечной точки x устройства OTG_FS (OTG_FS_DOEPTSIZx) (x = 1..3, где x = Номер конечной точки)

Карта регистров доступа к FIFO данных (DFIFO)

Регистры доступны в обоих режимах хоста и устройства. У каналов хоста IN, FIFO доступен по чтению. У каналов хоста OUT, FIFO доступен по записи.

Таблица 201. Карта регистров доступа к FIFO данных (DFIFO)

Секция регистров доступа к FIFO	Диапазон адресов	Доступ
Device IN Endpoint 0/Host OUT Channel 0: DFIFO Write Access Device OUT Endpoint 0/Host IN Channel 0: DFIFO Read Access	0x1000–0x1FFC	w r
Device IN Endpoint 1/Host OUT Channel 1: DFIFO Write Access Device OUT Endpoint 1/Host IN Channel 1: DFIFO Read Access	0x2000–0x2FFC	w r
...	...	...
Device IN Endpoint x(1)/Host OUT Channel x(1): DFIFO Write Access Device OUT Endpoint x(1)/Host IN Channel x(1): DFIFO Read Access	0xX000–0xXFFC	w r

1. x=3 у устройства и x=7 у хоста.

Карта CSR питания и тактов

Это отдельный регистр, доступный в режимах хоста и устройства.

Таблица 202. Регистр управления питанием и тактами

Название	Имя	Смещение адреса: 0xE00–0xFFF
Power and clock gating control register	OTG_FS_PCGCCTL	0xE00-0xE04
Reserved		0xE05–0xFFF

34.16.2. Общие регистры OTG_FS

Доступны в режимах хоста и устройства, при переключении режимов переписывать не надо.

Регистр управления и состояния (OTG_FS_GOTGCTL)

Смещение адреса: 0x000

По сбросу: 0x0000 0800

31–20	19	18	17	16	15–12	11	10	9	8	7–2	1	0
Reserved	BSVLD	ASVLD	DBCT	CIDSTS	Reserved	DHNPEN	HSHNPEN	HNPRQ	HNGSCS	Reserved	SRQ	SRQSCS
	r	r	r	r		rw	rw	rw	r		rw	r

— Биты 31:20 Резерв, не трогать.

— Бит 19 **BSVLD:** B-сессия в норме

0: B-сессия сбойная.

1: B-сессия достойная.

В режиме OTG можно использовать для определения факта подключения устройства.

NB: Доступен только в режиме устройства.

— Бит 18 **ASVLD:** A-сессия в норме

0: A-сессия сбойная

1: A-сессия достойная

NB: Доступен только в режиме хоста

— Бит 17 **DBCT:** Длинное/короткое время антидребезга обнаруженного подключения

0: Длинное, физическое подключение (100 ms + 2.5 µs)

1: Короткое, программное подключение (2.5 µs)

NB: Доступен только в режиме хоста

— Бит 16 **CIDSTS**: Состояние ID подключения

0: Контроллер OTG_FS в режиме A-устройства

1: Контроллер OTG_FS в режиме B-устройства

NB: Доступен в обоих режимах.

— Биты 15:12 Резерв, не трогать.

— Бит 11 **DHNPEN**: Разрешение HNP устройства

Успешно принята команда **SetFeature.SetHNPEnable** от хоста USB.

0: HNP не разрешён

1: HNP разрешён

NB: Доступен только в режиме устройства.

— Бит 10 **HSHNPEN**: Хост установил HNP устройству

Программно ставится после разрешения HNP на устройстве (командой **SetFeature.SetHNPEnable**).

0: Неудача

1: Успех

NB: Доступен только в режиме хоста

— Бит 9 **HNPRQ**: Выдача запроса HNP

Ставится программно. Программно можно снять записью 0 после установки бита HNSSCHG в регистре OTG_FS_GOTGINT. Ядро снимает этот бит после очистки бита HNSSCHG.

0: Запроса HNP нет

1: Запрос HNP

NB: Доступен только в режиме устройства.

— Бит 8 **HNGSCS**: Успешная беседа с Хостом

Ставится ядром при успешной беседе с хостом. Снимается ядром при стоящем запросе HNP (HNPRQ).

0: Не договорились

1: Успешно побеседовали

NB: Доступен только в режиме устройства.

— Биты 7:2 Резерв, не трогать.

— Бит 1 **SRQ**: Программный запрос сессии на USB

Ставится программой. Программно может сниматься записью 0 при изменении состояния беседы с хостом (стоит бит HNSSCHG в регистре OTG_FS_GOTGINT). Ядро снимает этот бит при чистом бите HNSSCHG. При использовании полноскоростного USB 1.1 перед запросом сессии надо дождаться разрядки V_{BUS} до 0.2 V после снятия бита BSVLD в OTG_FS_GOTGCTL. это время приведено в документации производителя.

0: Ничего не просили

1: Просим сессию

NB: Доступен только в режиме устройства.

— Бит 0 **SRQSCS**: Успешный запрос сессии

Ставится ядром.

0: Сбойный

1: Успех

NB: Доступен только в режиме устройства.

Регистр прерываний (OTG_FS_GOTGINT)

Смещение адреса: 0x004

По сбросу: 0x0000 0000

Флаги ставятся при прерываниях OTG, снимаются записью 1 в нужный бит.

31 … 20	19	18	17	16 … 10	9	8	7 … 3	2	1 … 0
Reserved	DBCDNE	ADTOCHG	HNGDET	Reserved	HNSSCHG	SRSSCHG	Reserved	SEDET	Res.
	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1		rc_w1	

— Биты 31:20 Резерв, не трогать.

— Бит 19 **DBCDNE**: Антидребезг завершён

Ставится ядром после подключения устройства. Теперь можно выдавать сброс USB. Бит работает только при стоящем бите HNPCAP или SRPCAP в регистре OTG_FS_GUSBCFG.

NB: Доступен только в режиме хоста

— Бит 18 **ADTOCHG**: Таймаут A-устройства

Ставится ядром после таймаута ожидания подключения B-устройства.

NB: Доступен в обоих режимах.

— Бит 17 **HNGDET**: Начало беседы хоста

Ставится ядром после обнаружения запроса беседы с хостом на USB.

NB: Доступен в обоих режимах.

— Биты 16:10 Резерв, не трогать.

— Бит 9 **HNSSCHG**: Состояние беседы хоста изменилось

Ставится ядром после изменения состояния бита HNGSCS в регистре OTG_FS_GOTGCTL.

NB: Доступен в обоих режимах.

— Бит 8 **SRSSCHG**: Состояние запроса сессии изменилось

Ставится ядром после изменения состояния бита SRQSCS в регистре OTG_FS_GOTGCTL.

NB: Доступен в обоих режимах.

— Биты 7:3 Резерв, не трогать.

— Бит 2 **SEDET**: Конец сессии

Ставится ядром после падения $V_{BUS} < 0.8$ V. Конец сессии B-устройства.

— Биты 1:0 Резерв, не трогать.

Регистр конфигурации AHB (OTG_FS_GAHBCFG)

Смещение адреса: 0x008

По сбросу: 0x0000 0000

Пишется после включения питания или смены режима до запуска любой передачи по AHB или USB. Параметры относятся в основном к AHB. После начальной записи изменять его нельзя.

31:9	8	7	6:1	0
Reserved	PTXFELVL	TXFELVL	Reserved	GINTMSK
	rw	rw		rw

— Биты 31:9 Резерв, не трогать.

— Бит 8 **PTXFELVL**: Уровень пустоты Периодического TxFIFO

Выдача прерывания пустого периодического TxFIFO (бит PTXFE в OTG_FS_GINTSTS) при:

0: Полупустом периодическом TxFIFO

1: Совсем пустом периодическом TxFIFO

NB: Доступен только в режиме хоста

— Бит 7 **TXFELVL**: Уровень пустоты TxFIFO не-периодического/точек IN

В режиме устройства: выдача прерывания пустого TxFIFO точек IN (TXFE в OTG_FS_DIEPINTx) при:

0: Полупустом TxFIFO

1: Совсем пустом TxFIFO

В режиме хоста: Выдача прерывания пустого не-периодического TxFIFO (бит NPTXFE в регистре OTG_FS_GINTSTS) при:

0: Полупустом не-периодическом TxFIFO

1: Совсем пустом не-периодическом TxFIFO

— Биты 31:9 Резерв, не трогать.

— Бит 0 **GINTMSK**: Маска Общего прерывания

Маска собранных воедино источников прерываний. Регистры состояния прерываний всё равно изменяются независимо от этого бита.

0: Прерывать нельзя.

1: Прерывать можно.

NB: Доступен в обоих режимах.

Регистр конфигурации USB (OTG_FS_GUSBCFG)

Смещение адреса: 0x00C

По сбросу: 0x0000 0A00

Пишется после включения питания или смены режима до запуска любой передачи по AHB или USB. Параметры относятся к USB и USB-PHY. После начальной записи изменять его нельзя.

31	30	29	28:14	13:10	9	8	7	6	5:4	3:2	1:0
CTXPKT	FDMOD	FHMOD	Reserved	TRDT	HNPCAP	SRPCAP	Res.	PHYSEL	Reserved		TOCAL
rw	rw	rw		rw	r/rw	r/rw		wo			rw

— Бит 31 **CTXPKT**: Нарушенный пакет Tx

Только для отладки, не ставьте этот бит ни за что и никогда.

NB: Доступен в обоих режимах.

— Бит 30 **FDMOD**: Принудительный режим устройства

Запись 1 переключает в режим устройства независимо от входной ножки ID.

0: Нормальный режим

1: Принудительный режим устройства

Изменения наступают через 25 ms после установки этого бита.

NB: Доступен в обоих режимах.

— Бит 29 **FHMOD**: Принудительный режим хоста

Запись 1 переключает в режим хоста независимо от входной ножки ID.

0: Нормальный режим

1: Принудительный режим хоста

Изменения наступают через 25 ms после установки этого бита.

NB: Доступен в обоих режимах.

— Биты 28:14 Резерв, не трогать.

— Биты 13:10 **TRDT**: Время обратной связи USB в тактах PHY

Они ставятся в соответствии с Таблицей 204, в зависимости от частоты AHB.

NB: Доступен только в режиме устройства.

— Бит 9 **HNPCAP**: Разрешение HNP

0: HNP запрещён.

1: HNP Разрешён.

NB: Доступен в обоих режимах.

— Бит 8 **SRPCAP**: Разрешение SRP

0: SRP запрещён.

1: SRP Разрешён.

NB: Доступен в обоих режимах.

— Бит 7 Резерв, не трогать.

— Бит 6 **PHYSEL**: Выбор полноскоростного трансивера

Всегда 1, только запись.

— Биты 5:3 Резерв, не трогать.

— Биты 2:0 **TOCAL**: Калибровка таймаута FS

Число тактов PHY, добавляемых к стандартной задержке (от 16 до 18 времён бита) между передачами пакетов для компенсации работы PHY от разных производителей. Определяется на основе скорости переучёта. На каждый такт PHY добавляется 0.25 времени бита.

Таблица 203. Значения TRDT

Диапазон частот AHB (MHz)		Мин. значение TRDT
Min.	Max	
14.2	15	0xF
15	16	0xE
16	17.2	0xD
17.2	18.5	0xC
18.5	20	0xB
20	21.8	0xA
21.8	24	0x9
24	27.5	0x8
27.5	32	0x7
32	—	0x6

Регистр сброса (OTG_FS_GRSTCTL)

Смещение адреса: 0x010

По сбросу: 0x0000 0A00

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AHBIDL	Reserved																				TXFNUM					TXFFLSH	RXFFLSH	Reserved	FCRST	HSRST	CSRST
r																					rw					rs	rs		rs	rs	rs

— Бит 31 **AHBIDL**: Простой ведущего AHB

Машина состояний ведущего AHB в Простое.

NB: Доступен в обоих режимах.

— Биты 30:11 Резерв, не трогать.

— Биты 10:6 **TXFNUM**: Номер сливаемого TxFIFO

FIFO сливается установкой TxFIFO Flush. Пока бит TxFIFO Flush стоит, изменять это поле нельзя.

00000:

– Не-периодический TxFIFO в режиме хоста

– TxFIFO 0 в режиме устройства

00001:

– Периодический TxFIFO в режиме хоста

– TxFIFO 1 в режиме устройства

00010: TxFIFO 2 в режиме устройства

...

00101: TxFIFO 15 в режиме устройства

10000: Все передающие FIFO в обоих режимах.

NB: Доступен в обоих режимах.

— Бит 5 **TXFFLSH**: Слив TxFIFO

Команда на слив одного или всех FIFO, но не в середине передач с FIFO.

Отсутствие передач проверяется так:

Для чтения из TxFIFO—Было прерывание "NAK в действии"

Для записи в TxFIFO—Стоит бит AHBIDL в регистре OTG_FS_GRSTCTL.

NB: Доступен в обоих режимах.

— Бит 4 **RXFFLSH**: Слив RxFIFO

Команда на слив всего RxFIFO, но не в середине передач с RxFIFO.

Перед записью в этот бит надо убедиться, что RxFIFO ни читается ни пишется.

Перед запуском других операций надо дождаться снятия этого бита. Это требует 8 тактов (самый медленный из PHY и AHB).

NB: Доступен в обоих режимах.

— Бит 3 Резерв, не трогать.

— Бит 2 **FCRST**: Сброс счётчика фреймов хоста

Следующий посылаемый SOF будет иметь номер фрейма 0.

NB: Доступен только в режиме хоста.

— Бит 1 **HSRST**: Программный сброс HCLK

Слив управляющей логики конвейеров домена тактов AHB после завершения передач AHB.

Управляющие биты CSR машин состояния домена AHB снимаются.

Маски прерываний машин состояния домена AHB снимаются.

Биты состояния прерываний машин состояния домена AHB не снимаются.

Очистка может занять несколько тактов.

NB: Доступен в обоих режимах.

— Бит 0 **CSRST**: Программный сброс ядра

Сбрасывает домены HCLK и PCLK:

Чистит прерывания и все биты регистров CSR за исключением битов:

- RSTPDMODL в OTG_FS_PCGCCTL
- GAYEHCLK в OTG_FS_PCGCCTL
- PWRCLMP в OTG_FS_PCGCCTL
- STPPCLK в OTG_FS_PCGCCTL
- FSLSPCS в OTG_FS_HCFG
- DSPD в OTG_FS_DCFG

Все машины состояния модуля (кроме блока ведомого AHB) сбрасываются в Простой, передающие и приёмный FIFO сливаются.

Передачи ведущего AHB завершаются после конца последней фазы данных. Передачи по USB завершаются немедленно.

Бит можно писать когда угодно. Снимается сам через несколько тактов. После снятия бита перед доступом к PHY надо подождать не меньше 3 тактов PHY (задержка синхронизации) и дождаться установки бита 31 в этом регистре (Простой ведущего AHB).

Обычно программный сброс используется при разработке программ и динамическом изменении битов выбора PHY, после которого надо сбросить и домен PHY.

NB: Доступен в обоих режимах.

Регистр прерываний ядра (OTG_FS_GINTSTS)

Смещение адреса: 0x014

По сбросу: 0x0400 0020

Некоторые биты работают только в режиме хоста, иные только в режиме устройства. В регистре отмечен текущий режим. Изменяемые биты состояния прерванный снимаются записью 1.

Биты состояния прерываний FIFO только читаются, они изменяются автоматически при обращении к FIFO.

Во избежание излишних прерываний чистить регистр `OTG_FS_GINTSTS` при инициализации надо до снятия маски запрета прерываний.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WKUINT	SRQINT	DISCINT	CIDSCHG	Reserved	PTXFE	HCINT	HPRTINT	Reserved		IPXFR/INCOMPISOOUT	IISOIXFR	OEPINT	IEPINT	Reserved		EOPF	ISOODRP	ENUMDNE	USBRST	USBSUSP	ESUSP	Reserved		GONAKEFF	GINAKEFF	NPTXFE	RXFLVL	SOF	OTGINT	MMIS	CMOD
rc_w1					r	r	r	Res.		rc_w1		r	r			rc_w1								r	r	r	r	rc_w1	r	rc_w1	r

— Бит 31 **WKUPINT**: Прерывание удалённой побудки

В режиме устройства ставится при появлении на шине USB события восстановления.

В режиме хоста ставится при появлении на шине USB события удалённой побудки.

NB: Доступен в обоих режимах.

— Бит 30 **SRQINT**: Прерывание запроса/новой сессии

В режиме хоста ставится при появлении на шине USB запроса сессии от устройства.

В режиме устройства ставится при появлении рабочего уровня V_{BUS} для B-устройства.

NB: Доступен в обоих режимах.

— Бит 29 **DISCINT**: Прерывание отключения (вытыкания) внешнего устройства

NB: Доступен в обоих режимах.

— Бит 28 **CIDSCHG**: Состояние линии ID разъёма изменилось

NB: Доступен в обоих режимах.

— Бит 27 Резерв, не трогать.

— Бит 26 **PTXFE**: Периодический TxFIFO пуст

Ставится если в периодическом передающем FIFO есть куда писать. Полу- или полное опустение задаётся битом PTXFELVL в регистре OTG_FS_GAHBCFG.

NB: Доступен только в режиме хоста.

— Бит 25 **HCINT**: Прерывание каналов хоста

В режиме хоста ставится ядром при наличии запроса какого-либо канала хоста. Номер запросившего канала читается из регистра OTG_FS_HAINT, причина запроса читается из регистра OTG_FS_HCINTx. Снимается очисткой бита запроса в регистре OTG_FS_HCINTx.

NB: Доступен только в режиме хоста.

— Бит 24 **HPRTINT**: Прерывание изменения состояния порта Хоста

Событие запроса читается из регистра OTG_FS_HPRT. Снимается очисткой бита запроса в регистре OTG_FS_HPRT.

NB: Доступен только в режиме хоста.

— Биты 23:22 Резерв, не трогать.

— Бит 21 **IPXFR**: Незавершённая периодическая передача

В режиме хоста ядро извещает о незавершённой периодической передаче для текущего фрейма.

INCOMPISOOUT: Незавершённая изохронная передача OUT

В режиме устройства ядро извещает о незавершённой изохронной передаче OUT для текущего фрейма. Выставляется вместе с битом конца периодического фрейма (EOPF) в этом регистре.

— Бит 20 **IISOIXFR**: Незавершённая изохронная передача IN

В режиме устройства ядро извещает о незавершённой изохронной передаче IN для текущего фрейма. Выставляется вместе с битом конца периодического фрейма (EOPF) в этом регистре.

NB: Доступен только в режиме устройства.

— Бит 19 **OEPINT**: Прерывание конечной точки OUT

В режиме устройства ставится ядром при наличии запроса какой-либо конечной точки OUT. Номер запросившей точки читается из регистра OTG_FS_DAINT, причина запроса читается из регистра OTG_FS_DOEPINTx. Снимается очисткой бита запроса в регистре OTG_FS_DOEPINTx.

NB: Доступен только в режиме устройства.

— Бит 18 **IEPINT**: Прерывание конечной точки IN

В режиме устройства ставится ядром при наличии запроса какой-либо конечной точки IN. Номер запросившей точки читается из регистра OTG_FS_DAINT, причина запроса читается из регистра OTG_FS_DIEPINTx. Снимается очисткой бита запроса в регистре OTG_FS_DIEPINTx.

NB: Доступен только в режиме устройства.

— Биты 17:16 Резерв, не трогать.

— Бит 15 **EOPF**: Прерывание конца периодического фрейма

Возникает при истечении интервала периодического фрейма (поле PFIVL в OTG_FS_DCFG) в текущем фрейме.

NB: Доступен только в режиме устройства.

— Бит 14 **ISOODRP**: Прерывание выброса изохронного пакета OUT

Ставится ядром при нехватке места в RxFIFO для изохронного пакета OUT.

NB: Доступен только в режиме устройства.

— Бит 13 **ENUMDNE**: Переучёт завершён

После этого в регистре OTG_FS_DSTS лежит скорость переучёта.

NB: Доступен только в режиме устройства.

— Бит 12 **USBRST**: Сигнал сброса на шине USB

NB: Доступен только в режиме устройства.

— Бит 11 **USBSUSP**: Сигнал Приостановки на шине USB

NB: Доступен только в режиме устройства.

— Бит 10 **ESUSP**: Ранняя приостановка

На шине USB не было активности в течении 3 ms.

NB: Доступен только в режиме устройства.

— Биты 9:8 Резерв, не трогать.

— Бит 7 **GONAKEFF**: Глобальный OUT NAK работает

Бит SGONAK в регистре OTG_FS_DCTL сработал. Очищается установкой бита CGONAK в регистре OTG_FS_DCTL.

NB: Доступен только в режиме устройства.

— Бит 6 **GINAKEFF**: Глобальный NAK не-периодических IN работает

Бит SGINAK в регистре OTG_FS_DCTL сработал. Очищается установкой бита CGINAK в регистре OTG_FS_DCTL.

Это прерывание не означает, что по шине USB будет посылаться NAK, бит STALL старше.

NB: Доступен только в режиме устройства.

— Бит 5 **NPTXFE**: Не-периодический TxFIFO пуст

Ставится если в не-периодическом передающем FIFO есть куда писать. Полу- или полное опустение задаётся битом TXFELVL в регистре OTG_FS_GAHBCFG.

NB: Доступен только в режиме устройства.

— Бит 4 **RXFLVL**: RxFIFO не пуст

Из RxFIFO можно прочесть хоть один пакет.

NB: Доступен в обоих режимах.

— Бит 3 **SOF**: Старт фрейма

В режиме хоста ядро извещает что на шину USB послан SOF (FS) или Keep-Alive (LS). Снимается записью 1 в этот бит.

В режиме устройства ядро извещает, что на USB появился токен. Номер текущего фрейма читают в регистре Состояния устройства. Прерывание проявляется только при работе с FS.

NB: Доступен в обоих режимах.

— Бит 2 **OTGINT**: Прерывание OTG

Ядро извещает о событии протокола OTG. Само событие выявляют по регистру OTG_FS_GOTGINT. Бит снимается очисткой бита события в регистре OTG_FS_GOTGIN.

NB: Доступен в обоих режимах.

— Бит 1 **MMIS**: Прерывание несовпадения режима

Ядро ставит этот бит при попытке доступа:

– К регистрам режима хоста из режима устройства
– К регистрам режима устройства из режима хоста

NB: Доступен в обоих режимах.

— Бит 0 **CMOD**: Текущий режим

0: Режим устройства
1: Режим хоста

NB: Доступен в обоих режимах.

Регистр маски прерываний (OTG_FS_GINTMSK)

Смещение адреса: 0x018

По сбросу: 0x0000 0000

Изменение регистра маски на регистр запросов прерывание (OTG_FS_GINTSTS) не влияет.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WUIM	SRQIM	DISCINT	CIDSCHGM	Reserved	PTXFEM	HCIM	PRTIM	Reserved		IPXFRM/IISOOXFRM	IISOIXFRM	OEPINT	IEPINT	EPMISM	Reserved	EOPFM	ISOODRPM	ENUMDNEM	USBRST	USBSUSPM	ESUSPM	Reserved		GONAKEFFM	GINAKEFFM	NPTXFEM	RXFLVLM	SOFM	OTGINT	MMISM	Reserved
rw	rw	rw	rw		rw	rw	r			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	

Состояние битов:

0: Нельзя

1: Можно

— Бит 31 **WUIM**: Маска прерывания побудки
NB: Доступен в обоих режимах.

— Бит 30 **SRQIM**: Маска прерывания запроса/новой сессии
NB: Доступен в обоих режимах.

— Бит 29 **DISCINT**: Маска прерывания отключения устройства
NB: Доступен только в режиме устройства.

— Бит 28 **CIDSCHGM**: Маска прерывания изменения ID
NB: Доступен в обоих режимах.

— Бит 27 Резерв, не трогать.

— Бит 26 **PTXFEM**: Маска прерывания пустого периодического TxFIFO
NB: Доступен только в режиме хоста.

— Бит 25 **HCIM**: Маска прерывания каналов хоста
NB: Доступен только в режиме хоста.

— Бит 24 **PRTIM**: Маска прерывания порта хоста
NB: Доступен только в режиме хоста.

— Биты 23:22 Резерв, не трогать.

— Бит 21 **IPXFRM**: Маска прерывания незавершённой периодической передачи
NB: Доступен только в режиме хоста.
IISOOXFRM: Маска прерывания незавершённой изохронной передачи OUT
NB: Доступен только в режиме устройства.

— Бит 20 **IISOIXFRM**: Маска прерывания незавершённой изохронной передачи IN
NB: Доступен только в режиме устройства.

— Бит 19 **OEPINT:** Маска прерывания конечной точки OUT
NB: Доступен только в режиме устройства.

— Бит 18 **IEPINT**: Маска прерывания конечной точки IN
NB: Доступен только в режиме устройства.

— Бит 17 **EPMISM**: Маска прерывания несовпадения конечной точки
NB: Доступен только в режиме устройства.

— Бит 16 Резерв, не трогать.

— Бит 15 **EOPFM**: Маска прерывания конца периодического фрейма
NB: Доступен только в режиме устройства.

— Бит 14 **ISOODRPM**: Маска прерывания выброса изохронного пакета OUT
NB: Доступен только в режиме устройства.

— Бит 13 **ENUMDNEM**: Маска прерывания конца переучёта
NB: Доступен только в режиме устройства.

— Бит 12 **USBRST**: Маска прерывания сброса USB
NB: Доступен только в режиме устройства.

— Бит 11 **USBSUSPM**: Маска прерывания приостановки USB
NB: Доступен только в режиме устройства.

— Бит 10 **ESUSPM**: Маска прерывания ранней приостановки
NB: Доступен только в режиме устройства.

— Биты 9:8 Резерв, не трогать.

— Бит 7 **GONAKEFFM**: Маска прерывания Global OUT NAK effective
NB: Доступен только в режиме устройства.

— Бит 6 **GINAKEFFM**: Маска прерывания Global non-periodic IN NAK effective
NB: Доступен только в режиме устройства.
— Бит 5 **NPTXFEM**: Маска прерывания пустого не-периодического TxFIFO
NB: Доступен только в режиме хоста.
— Бит 4 **RXFLVLM**: Маска прерывания непустого RxFIFO
NB: Доступен в обоих режимах.
— Бит 3 **SOFM**: Маска прерывания старта фрейма
NB: Доступен в обоих режимах.
— Бит 2 **OTGINT**: Маска прерывания OTG
NB: Доступен в обоих режимах.
— Бит 1 **MMISM**: Маска прерывания несовпадения режима
NB: Доступен в обоих режимах.
— Бит 0 Резерв, не трогать.

Регистры Состояния приёма отладки/Состояния приёма и извлечения (OTG_FS_GRXSTSR/OTG_FS_GRXSTSP)

Смещение адреса регистра состояния приёма: 0x01C

Смещение адреса регистра извлечения: 0x020

По сбросу: 0x0000 0000

Чтение Состояния приёма отладки возвращает верхушку приёмного FIFO. Чтение Состояния приёма и извлечения дополнительно извлекает данные верхней строки RxFIFO.

The receive status contents must be interpreted differently in host and device modes. The core ignores the receive status pop/read when the receive FIFO is empty and returns a value of 0x0000 0000. The application must only pop the Receive Status FIFO when the Receive FIFO non-empty bit of the Core interrupt register (`RXFLVL` bit in `OTG_FS_GINTSTS`) is asserted.

Режим хоста

31–21	20–17	16–15	14–4	3–0
Reserved	PKTSTS	DPID	BCNT	CHNUM
	r	r	r	r

— Биты 31:21 Резерв, не трогать.
— Биты 20:17 **PKTSTS**: Статус принятого пакета
0010: Принят пакет данных IN
0011: Передача IN завершена (с прерыванием)
0101: Ошибка переключения данных (с прерыванием)
0111: Канал остановлен (с прерыванием)
Иные: Резерв
— Биты 16:15 **DPID**: PID данных принятого пакета
00: DATA0
10: DATA1
01: DATA2
11: MDATA
— Биты 14:4 **BCNT**: Счётчик байтов принятого пакета IN
— Биты 3:0 **CHNUM**: Номер канала текущего принятого пакета

Режим устройства

31–25	24–21	20–17	16–15	14–4	3–0
Reserved	FRMNUM	PKTSTS	DPID	BCNT	EPNUM
	r	r	r	r	r

— Биты 31:25 Резерв, не трогать.
— Биты 24:21 **FRMNUM**: Номер фрейма
Младшие 4 бита номера фрейма принятого пакета. Только при поддержке изохронных точек OUT.
— Биты 20:17 **PKTSTS**: Статус принятого пакета
0001: Глобальный OUT NAK (с прерыванием)
0010: Принят пакет данных OUT
0011: Передача OUT завершена (с прерыванием)

0100: Передача SETUP завершена (с прерыванием)
0110: Принят пакет данных SETUP
Others: Reserved

— Биты 16:15 **DPID**: PID данных принятого пакета OUT
00: DATA0
10: DATA1
01: DATA2
11: MDATA

— Биты 14:4 **BCNT**: Счётчик байтов принятого пакета

— Биты 3:0 **EPNUM**: Номер конечной точки текущего принятого пакета

Регистр размера приёмного FIFO (OTG_FS_GRXFSIZ)

Смещение адреса: 0x024

По сбросу: 0x0000 0200

31–16	15–0
Reserved	RXFD
	r/rw

— Биты 31:16 Резерв, не трогать.

— Биты 15:0 **RXFD**: Глубина RxFIFO в 32-бит словах
Минимальное значение 16
Максимальное значение 256
По сбросу питания пишется значение наибольшей глубины Rx FIFO данных.

Регистр размера передающего не-периодического FIFO хоста (OTG_FS_HNPTXFSIZ)
Размер передающего FIFO точки 0 (OTG_FS_DIEPTXF0)

Смещение адреса: 0x028

По сбросу: 0x0000 0200

31–16	15–0
NPTXFD/TX0FD	NPTXFSA/TX0FSA
r/rw	r/rw

Режим хоста

— Биты 31:16 **NPTXFD**: Глубина не-периодического TxFIFO в 32-бит словах.
Минимальное значение 16
Максимальное значение 256

— Биты 15:0 **NPTXFSA**: Адрес начала RAM не-периодических передач

Режим устройства

— Биты 31:16 **TX0FD**: Глубина TxFIFO конечной точки 0 в 32-бит словах.
Минимальное значение 16
Максимальное значение 256

— Биты 15:0 **TX0FSA**: Адрес начала RAM передач конечной точки 0

Регистр не-периодического передающего FIFO/состояния очереди (OTG_FS_HNPTXSTS)

Смещение адреса: 0x02C

По сбросу: 0x0008 0200

Содержит информацию о свободном месте в TxFIFO.

31	30–24	23–16	15–0
Reserved	NPTXQTOP	NPTQXSAV	NPTXFSAV
	r	r	r

— Бит 31 Резерв, не трогать.

— Биты 30:24 **NPTXQTOP**: Верх очереди не-периодических запросов передачи
Биты 30:27: Номер Канала/Конечной точки
Биты 26:25: Статус передачи
– 00: Токен IN/OUT
– 01: Пакет нулевой длины (устройство IN/хост OUT)
– 11: Команда остановки канала

Бит 24: Окончание (последняя строка этого Канала/Конечной точки)

— Биты 23:16 **NPTQXSAV**: Доступное место в очереди не-периодических запросов передачи
В режиме хоста очередь хранит запросы и IN и OUT, в режиме устройства только запросы. IN.
00: Очередь запросов не-периодических передач заполнена
01: 1 свободная строка
10: 2 свободных строки
bxn: *n* свободных строк (0 ≤ *n* ≤ 8)
Иные: Резерв

— Биты 15:0 **NPTXFSAV**: Доступное место очереди TxFIFO в 32-бит словах
00: Непериодический TxFIFO полон
01: 1 свободное слово
10: 2 свободных слова
0xn: *n* свободных слов (0 ≤ *n* ≤ 256)
Иные: Резерв

Общий регистр конфигурации ядра (OTG_FS_GCCFG)

Смещение адреса: 0x038

По сбросу: 0x0000 0000

31–21	20	19	18	17	16	15–0
Reserved	SOFOUTEN	VBUSBSEN	VBUSASEN	Reserved	.PWRDWN	Reserved
	rw	rw	rw		rw	

— Биты 31:21 Резерв, не трогать.

— Бит 20 **SOFOUTEN**: Разрешение выдачи SOF на PAD
0: Нельзя
1: Можно

— Бит 19 **VBUSBSEN:** Разрешение проверки V_{BUS} устройства “B”
0: Нельзя
1: Можно

— Бит 18 **VBUSASEN:** Разрешение проверки V_{BUS} устройства “A”
0: Нельзя
1: Можно

— Бит 17 Резерв, не трогать.

— Бит 16 **PWRDWN**: Выключение питания трансивера
0: Питание выключено
1: Трансивер включён

— Биты 15:0 Резерв, не трогать.

Регистр ID ядра (OTG_FS_CID)

Смещение адреса: 0x03C

По сбросу: 0x0000 1100

31–0
PRODUCT_ID
rw

— Биты 31:21 **PRODUCT_ID**: Программируемый ID продукта

Регистр размера периодического передающего FIFO хоста (OTG_FS_HPTXFSIZ)

Смещение адреса: 0x100

По сбросу: 0x0200 0600

31–16	15–0
PTXFSIZ	PTXSA
r/rw	r/rw

— Биты 31:16 **PTXFD**: Глубина периодического TxFIFO хоста в 32-бит словах
Минимальное значение 16

— Биты 15:0 **PTXSA**: Адрес начала периодического TxFIFO хоста
По сбросу питания содержит сумму наибольших глубин Rx FIFO данных и не-периодического Tx FIFO данных.

Регистр размера передающего FIFO конечной точки (OTG_FS_DIEPTXFx) (x = 1..3, где x это номер FIFO)

Смещение адреса: 0x104 + (FIFO_number – 1) × 0x04

По сбросу: 0x0200 0400

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INEPTXFD																INEPTXSA															
r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw

— Биты 31:16 **INEPTXFD**: Глубина TxFIFO конечной точки N в 32-бит словах
Минимальное значение 16
По сбросу питания это значение определено как наибольшая глубина FIFO точки IN.

— Биты 15:0 **INEPTXSA**: Адрес начала FIFOx передачи конечной точки IN
Должен быть выравнен на границу 32-бит слова.

34.16.3. Регистры режима хоста

В режиме устройства их содержимое не определено.

Регистр конфигурации хоста (OTG_FS_HCFG)

Смещение адреса: 0x400

По сбросу: 0x0000 0000

Конфигурация ядра по сбросу питания. После инициализации хоста изменять нельзя.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																													FSLSS	FSLSPCS	
																													r	rw	rw

— Биты 31:16 Резерв, не трогать.

— Бит 2 **FSLSS**: Поддержка только FS и LS
Определяет скорость переучёта. Пишется единожды.
1: Только FS/LS, даже для устройств с поддержкой HS (только чтение)

— Биты 1:0 **FSLSPCS**: Выбор тактов PHY FS/LS

В режиме FS
01: PHY на частоте 48 MHz
Иные: Резерв

В режиме LS
00: Резерв
01: PHY на частоте 48 MHz
10: PHY на частоте 6 MHz
11: Резерв

NB: Ставить надо после события подключения устройства в соответствии с его частотой (после изменения бита надо делать программный сброс).

Регистр интервала фреймов хоста (OTG_FS_HFIR)

Смещение адреса: 0x404

По сбросу: 0x0000 EA60

Значение интервала фреймов для текущей скорости переучёта.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																FRIVL															
																rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:16 Резерв, не трогать.

— Биты 15:0 **FRIVL**: Интервал фреймов

Программное определение интервала между двумя последовательными токенами SOF (FS) или Keep-Alive (LS) в тактах PHY. Писать можно только после установки бита PENA в регистре OTG_FS_HPRT. Если не задано, то ядро вычисляет его по значению поля FSLSPCS в регистре OTG_FS_HCFG. Писать надо единожды при начальной конфигурации.

1 ms × (частоту тактов PHY)

Регистр номера/остатка времени фрейма хоста (OTG_FS_HFNUM)

Смещение адреса: 0x408

По сбросу: 0x0000 3FFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FTREM																FRNUM															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

— Биты 31:16 **FTREM**: Остаток времени фрейма в тактах PHY

Декрементируется по каждому такту PHY. Перегружается из регистра интервала при достижении 0 и передаче нового SOF.

— Биты 15:0 **FRNUM**: Номер фрейма

Инкрементируется по новому SOF, обнуляется при достижении 0x3FFF.

Регистр состояния периодических передающего FIFO/очереди (OTG_FS_HPTXSTS)

Смещение адреса: 0x410

По сбросу: 0x0008 0100

Только чтение. Информация о свободном месте в TxFIFO и очереди периодических передач.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PTXQTOP								PTXQSAV								PTXFSAVL															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:24 **PTXQTOP**: Верх очереди запросов периодических передач

Используется при отладке.

Бит 31: Чётный/Нечётный фрейм

0: Чётный

1: Нечётный

Биты 30:27: Номер Канала/Конечной точки

Биты 26:25: Тип

00: IN/OUT

01: Пакет нулевой длины

11: Команда выключения канала

Бит 24: Последняя строка этого Канала/Конечной точки

— Биты 23:16 **PTXQSAV**: Свободное место в очереди периодических передач

Очередь содержит и IN и OUT запросы.

00: Очередь полна

01: 1 свободная строка

10: 2 свободных строки

bxn: n свободных строк (0 ≤ n ≤ 8)

Others: Резерв

— Биты 15:0 **PTXFSAVL**: Свободное место в TxFIFO периодических передач в 32-би словах

0000: TxFIFO полон

0001: 1 свободное слово

0010: 2 свободных слова

bxn: n свободных слов (0 ≤ n ≤ PTXFD)

Others: Резерв

Регистр всех прерываний хоста (OTG_FS_HAINT)

Смещение адреса: 0x414

По сбросу: 0x0000 0000

Отражает запросы прерываний каналов. Прерывание вызывается по общему биту HCINT в регистре OTG_FS_GINTSTS при разрешении в регистре маски всех каналов. Биты ставятся и снимаются в соответствии с битами регистров прерывания соответствующего канала x.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																HAINT															
																r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

— Биты 31:24 Резерв, не трогать.

— Биты 31:24 **HAINT**: Прерывания каналов

По одному биту на канал: бит 0 для канала 0, бит 15 для канала 15

Регистр маски всех прерываний хоста (OTG_FS_HAINTMSK)

Смещение адреса: 0x418

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																HAINTM															
																rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:24 Резерв, не трогать.

— Биты 31:24 **HAINTM**: Маска прерывания каналов

0: Нельзя

1: Можно

По одному биту на канал: бит 0 для канала 0, бит 15 для канала 15

Регистр управления и состояния портов хоста (OTG_FS_HPRT)

Смещение адреса: 0x440

По сбросу: 0x0000 0000

Сейчас хост OTG поддерживает только один порт. Регистр содержит информацию о сбросе USB, включении, остановке, восстановлении, подключении и поверке. Прерывания вызываются по биту HPRTINT в регистре OTG_FS_GINTSTS. Чтение этого регистра снимает бит запроса прерывания. Биты rc_w1 снимаются записью 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													PSPD		PTCTL				PPWR	PLSTS		Reserved	PRST	PSUSP	PRES	POCCHNG	POCA	PENCHNG	PENA	PCDET	PCSTS
													r	r	rw	rw	rw	rw	rw	r	r		rw	rs	rw	rc_w1	r	rc_w1	rc_w0	rc_w1	r

— Биты 31:19 Резерв, не трогать.

— Биты 18:17 **PSPD**: Скорость подключённого к порту устройства

01: FS

10: LS

11: Резерв

— Биты 16:13 **PTCTL**: Тест порта

0000: Режим теста выключен

0001: Режим Test_J

0010: Режим Test_K

0011: Режим Test_SE0_NAK

0100: Режим Test_Packet

0101: Режим Test_Force_Enable

Others: Резерв

— Бит 12 **PPWR**: Питание порта

0: Выключено

1: Включено

— Биты 11:10 **PLSTS**: Текущее состояние линий порта

Бит 10: Логический уровень OTG_FS_FS_DP

Бит 11: Логический уровень f OTG_FS_FS_DM

— Бит 9 Резерв, не трогать.

— Бит 8 **PRST**: Сброс порта

Ставится программно, снимается программно после завершения последовательности сброса.

0: Сбросу нету

1: Сброс есть

Ставится на период не менее 10 ms перед запуском сброса порта и ещё ждём 10 ms перед снятием этого бита. В стандарте USB этого нет.

— Бит 7 **PSUSP**: Приостановка порта

Ставится программно. Ядро просто перестаёт посылать SOF. Для остановки тактов PHY надо ставить свой бит остановки. При чтении выдаёт текущее состояние порта.

Снимается ядром после установки битов WKUINT или DISCINT в регистре OTG_FS_GINTSTS.

0: Порт работает

1: Порт стоит

— Бит 6 **PRES**: Восстановление порта

Программно подаёт на порт сигнал восстановления. При программном снятии бита снимается и сигнал. При обнаружении побудки (бит WKUINT в OTG_FS_GINTSTS) ядро выдаёт сигнал и снимает его после обнаружении отключения. Чтение выдаёт текущее состояние сигнала.

0: Сигнала восстановления нет

1: Сигнал есть

— Бит 5 **POCCHNG**: Сигнал перегрузки порта по току изменился

— Бит 4 **POCA**: Перегрузка порта по току

0: Нету

1: Есть

— Бит 3 **PENCHNG**: Включение порта изменилось

— Бит 2 **PENA**: Включение порта

Порт включается ядром в конце последовательности сброса, выключается при перегрузке, отключении или программном снятии этого бита. Программно бит не ставится. Прерывания не вызывает.

0: Выключен

1: Включён

— Бит 1 **PCDET**: Порт подключён

Ставится ядром при подключении устройства, прерывание вызывается битом HPRTINT в регистре OTG_FS_GINTSTS). Снимается записью 1.

— Бит 0 **PCSTS**: Состояние подключения порта

0: Устройства нет

1: Устройство есть

Регистр характеристик канала-x (OTG_FS_HCCHARx) (x = 0..7, где x = Номер_канала)

Смещение адреса: 0x500 + (Номер_канала × 0x20)

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
rs	rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Бит 31 **CHENA**: Включение канала

Ставится программно, снимается хостом OTG.

0: Выключен

1: Включён

— Бит 30 **CHDIS**: Выключение канала

Программная установка бита останавливает приём/передачу данных не дожидаясь их завершения. Далее надо дождаться подтверждения выключения прерыванием.

— Бит 29 **ODDFRM**: Нечётный фрейм

Программно задаёт передачу чётного/нечётного фрейма периодической (изохронной или прерывающей) передачи.

0: Чётный фрейм

1: Нечётный фрейм

— Биты 28:22 **DAD**: Адрес устройства

— Биты 21:20 **MCNT**: Многосчётчик

Число передач для этого фрейма периодической конечной точки. Для не-периодических точек не используется.

00: Резерв. Результат непредсказуем
01: 1 передача
10: 2 передачи на фрейм
11: 3 передачи на фрейм

— Биты 19:18 **EPTYP**: Тип конечной точки
00: Управляющая
01: Изохронная
10: Групповая
11: Прерывающая

— Бит 17 **LSDEV**: Низкоскоростное устройство

— Бит 16 Резерв. не трогать.

— Бит 15 **EPDIR**: Направление конечной точки
0: OUT
1: IN

— Биты 14:11 **EPNUM**: Номер конечной точки

— Биты 10:0 **MPSIZ**: Максимальный размер пакета

Регистр прерываний канала-x (OTG_FS_HCINTx) (x = 0..7, где x = Номер_канала)

Смещение адреса: 0x508 + (Номер_канала × 0x20)

По сбросу: 0x0000 0000

Читать регистр надо при стоящем бите HCINT в регистре OTG_FS_GINTSTS, но сначала надо узнать номер прервавшего канала из регистра OTG_FS_HAINT. Снятие бита в этом регистре чистит соответствующие биты в регистрах OTG_FS_HAINT и OTG_FS_GINTSTS.

31 … 11	10	9	8	7	6	5	4	3	2	1	0
Reserved	DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
	rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1	rc_w1		rc_w1	rc_w1

— Биты 31:11 Резерв. не трогать.

— Бит 10 **DTERR**: Ошибка переключения данных

— Бит 9 **FRMOR**: Переполнение фрейма

— Бит 8 **BBERR**: Ошибка перекрёстных шумов

— Бит 7 **TXERR**: Ошибка передачи
Сбой CRC
Таймаут
Ошибка бита заполнения
Дрянной EOP

— Бит 6 Резерв. не трогать.

— Бит 5 **ACK**: Прерывание принятого/переданного ACK

— Бит 4 **NAK**: Прерывание принятого ответа NAK

— Бит 3 **STALL**: Прерывание принятого ответа STALL

— Бит 2 Резерв. не трогать.

— Бит 1 **CHH**: Канал остановлен
Indicates the transfer completed abnormally either because of any USB transaction error or in response to disable request by the application.

— Бит 0 **XFRC**: Передача завершена без ошибок

Регистр маски прерываний канала-x (OTG_FS_HCINTMSKx) (x = 0..7, где x = Номер_канала)

Смещение адреса: 0x508 + (Номер_канала × 0x20)

По сбросу: 0x0000 0000

31–11	10	9	8	7	6	5	4	3	2	1	0
Reserved	DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

Маска прерывания события: 0- нельзя, 1 - можно.

— Биты 31:11 Резерв. не трогать.
— Бит 10 **DTERRM**: Ошибка переключения данных
— Бит 9 **FRMORM**: Переполнение фрейма
— Бит 8 **BBERRM**: Ошибка перекрытых шумов
— Бит 7 **TXERRM**: Ошибка передачи
— Бит 6 **NYET**: Ответ принят
— Бит 5 **ACKM**: Прерывание принятого/переданного ACK
— Бит 4 **NAKM**: Прерывание принятого ответа NAK
— Бит 3 **STALLM**: Прерывание принятого ответа STALL
— Бит 2 Резерв. не трогать.
— Бит 1 **CHHM**: Канал остановлен
— Бит 0 **XFRCM**: Передача завершена

Регистр размера передачи канала канала-x (OTG_FS_HCTSIZx) (x = 0..7, (x = 0..7, где x = Номер_канала)

Смещение адреса: 0x510 + (Номер_канала × 0x20)

По сбросу: 0x0000 0000

31	30–29	28–19	18–0
Reserved	DPID	PKTCNT	XFRSIZ
rw	rw	rw	rw

— Бит 31 Резерв. не трогать.
— Биты 30:29 **DPID:** PID данных начальной передачи

Программно задаёт PID данных начальной передачи. Для остальных передач управляется хостом.

00: DATA0
01: DATA2
10: DATA1
11: MDATA (non-control)/SETUP (control)

— Биты 28:19 **PKTCNT**: Счётчик пакетов

Программно пишется ожидаемое число передаваемых (OUT) или принимаемых (IN) пакетов передачи. Декрементируется хостом после каждой успешной передачи пакета OUT/IN. При достижении 0 выдаётся прерывание завершения.

— Биты 18:0 **XFRSIZ**: Размер передачи

Для OUT это число байтов данных передачи хоста.

Для IN это размер приёмного буфера. Должен быть кратен максимальному размеру пакета IN (периодического и не-периодического).

34.16.4. Регистры режима устройства

Регистр конфигурации устройства (OTG_FS_DCFG)

Смещение адреса: 0x800

По сбросу: 0x0220 0000

Конфигурация ядра по сбросу питания, некоторых команд или переучёта. После инициализации регистр изменять нельзя.

31–13	12–11	10–4	3	2	1–0
Reserved	PFIVL	DAD	Reserved	NZLSOHSK	DSPD
	rw	rw		rw	rw

— Биты 31:13 Резерв. не трогать.

— Биты 12:11 **PFIVL**: Интервал периодического фрейма

Время выдачи прерывания конца периодического фрейма до полного завершения его передач.

00: 80% интервала фрейма

01: 85% интервала фрейма

10: 90% интервала фрейма

11: 95% интервала фрейма

— Биты 10:4 **DAD**: Адрес устройства

Пишется после каждой команды **SetAddress**.

— Бит 3 Резерв. не трогать.

— Бит 2 **NZLSOHSK**: Ответ на посылку OUT не-нулевой длины фазы статуса

0: Отдать принятый пакет OUT программе (нулевой или не-нулевой длины) и послать ответ на основе битов NAK и STALL конечной точки в регистре управления устройства.

1: На передачу статуса OUT не-нулевой длины ответить STALL, пакет программе не отдавать.

— Биты 1:0 **DSPD**: Скорость устройства

Скорость, показываемая устройством во время переучёта, но реальная скорость определяется хостом USB после завершения "чик-чирик" последовательности.

00: Резерв

01: Резерв

10: Резерв

11: FS (USB 1.1 Частота 48 MHz)

Регистр управления устройства (OTG_FS_DCTL)

Смещение адреса: 0x804

По сбросу: 0x0000 0000

31–12	11	10	9	8	7	6–4	3	2	1	0
Reserved	POPRGDNE	CGONAK	SGONAK	CGINAK	SGINAK	TCTL	GONSTS	GINSTS	SDIS	RWUSIG
	rw	w	w	w	w	rw	r	r	rw	rw

— Биты 31:12 Резерв. не трогать.

— Бит 11 **POPRGDNE**: Побудка включения завершена

Ставится программно после конфигурации регистра.

— Бит 10 **CGONAK**: Очистка глобального OUT NAK записью 1 в этот бит.

— Бит 9 **SGONAK**: Установка глобального OUT NAK

Выдаёт ответ NAK по всем конечным точкам OUT.

Ставится программно только при снятом бите GONAKEFF в регистре OTG_FS_GINTSTS.

— Бит 8 **CGINAK**: Очистка глобального IN NAK записью 1 в этот бит.

— Бит 7 **SGINAK**: Установка глобального IN NAK

Выдаёт ответ NAK по всем не-периодическим конечным точкам IN.

Ставится программно только при снятом бите GINAKEFF в регистре OTG_FS_GINTSTS.

— Биты 6:4 **TCTL**: Управление тестом

000: Выключен

001: Режим Test_J

010: Режим Test_K

011: Режим Test_SE0_NAK

100: Режим Test_Packet

101: Режим Test_Force_Enable

Иные: Резерв

— Бит 3 **GONSTS**: Состояние глобального OUT NAK

0: Ответ посылается на основе заполнения RxFIFO и состояния битов NAK и STALL.

1: Данные в RxFIFO не пишутся, независимо от заполнения. На все пакеты, кроме SETUP, отсылается NAK. Изохронные пакеты OUT пропадают втуне.

— Бит 2 **GINSTS**: Состояние глобального IN NAK

0: Ответ отсылается на основе наличия данных в передающем FIFO.

1: По всем не-периодическим точкам IN отсылается ответ NAK, независимо от данных в FIFO.

— Бит 1 **SDIS**: Программное отключение

При стоящем бите хост в упор не видит подключённого устройства и устройство не принимает сигналы по USB. При очистке этого бита после программного отключения хосту посылается событие подключения. После переподключения устройства хост начинает его переучёт.

0: Нормальная работа.

1: Ядро выдаёт хосту событие подключения.

— Бит 0 **RWUSIG**: Удалённое пробуждение

Установка бита посылает хосту сигнал удалённой побудки. Очищать бит надо в интервале от 1 ms до 15 ms после установки.

Таблица 205. Минимальная длительность программного отключения

Рабочая скорость	Состояние устройства	Минимальная длительность
FS	Остановка	1 ms + 2.5 µs
FS	Простой	2.5 µs
FS	Работа	2.5 µs

Регистр состояния устройства (OTG_FS_DSTS)

Смещение адреса: 0x808

По сбросу: 0x0000 0010

Состояние событий ядра со стороны USB. Читать надо по прерываниям от регистра OTG_FS_DAINT).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										FNSOF														Reserved				EERR	ENUMSPD		SUSPSTS
										r	r	r	r	r	r	r	r	r	r	r	r	r	r					r	r	r	r

— Биты 31:22 Резерв. не трогать.

— Биты 21:8 **FNSOF**: Номер фрейма принятого SOF

— Биты 7:4 Резерв. не трогать.

— Бит 3 **EERR**: Непредсказуемая ошибка

Ставится ядром. Контроллер OTG_FS уходит в Останов и выдаётся прерывание раннего останова (бит ESUSP в OTG_FS_GINTSTS). Восстановить работу можно только через программное отключение.

— Биты 2:1 **ENUMSPD**: Скорость после переучёта

01: Резерв

10: Резерв

11: FS (частота PHY равна 48 MHz)

Иные: Резерв

— Бит 0 **SUSPSTS**: Состояние приостановки

В режиме устройства отражает сигнал Останова на шине USB. Ядро выходит из Останова при:

– Активности на линиях данных USB

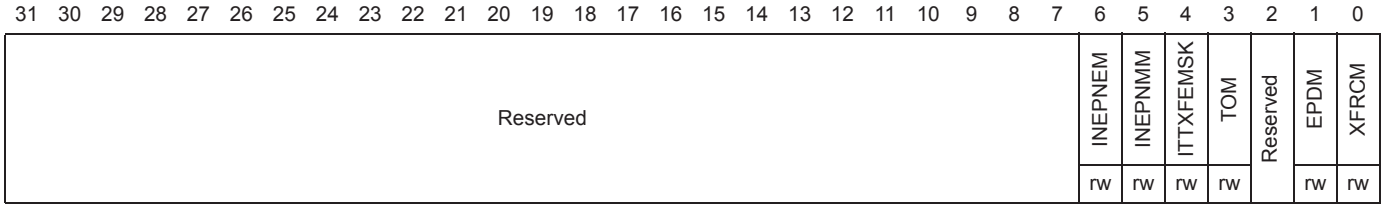
– Установке бита удалённой побудки RWUSIG в регистре OTG_FS_DCTL.

Общий регистр маски прерываний конечных точек IN устройства (OTG_FS_DIEPMSK)

Смещение адреса: 0x810

По сбросу: 0x0000 0000

Относится к выдаче прерываний всех регистров OTG_FS_DIEPINTx. По умолчанию всё замаскировано.



Маска прерывания: 0 - Нельзя, 1 - Можно.

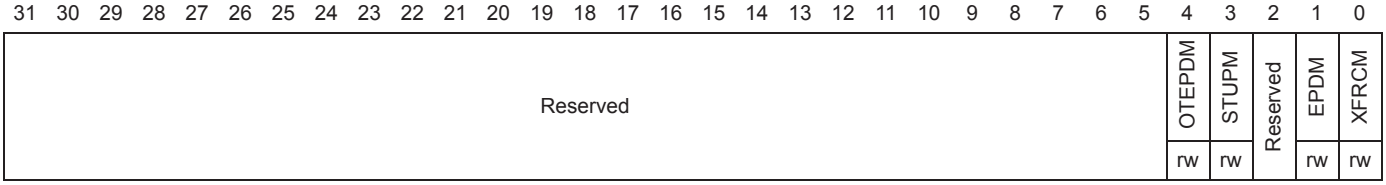
— Биты 31:7 Резерв. не трогать.
— Бит 6 **INEPNEM**: У конечной точки IN действует NAK
— Бит 5 **INEPNMM**: Принят токен IN с несовпадающей EP
— Бит 4 **ITTXFEMSK**: Принят токен IN при пустом TxFIFO
— Бит 3 **TOM**: Таймаут (Не-изохронные конечные точки)
— Бит 2 Резерв. не трогать.
— Бит 1 **EPDM**: Конечная точка выключена
— Бит 0 **XFRCM**: Передача завершена

Общий регистр маски прерываний конечных точек OUT устройства (OTG_FS_DOEPMSK)

Смещение адреса: 0x814

По сбросу: 0x0000 0000

Относится к выдаче прерываний всех регистров `OTG_FS_DOEPINTx` По умолчанию всё замаскировано.



Маска прерывания: 0 - Нельзя, 1 - Можно.

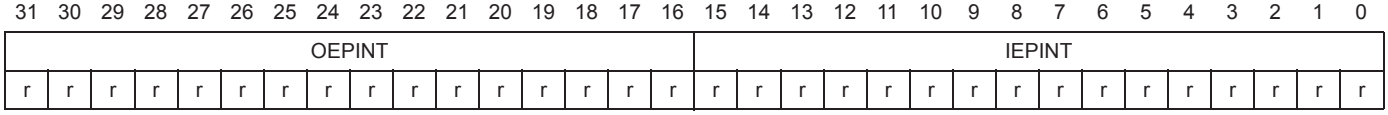
— Биты 31:5 Резерв. не трогать.
— Бит 4 **OTEPDM**: Принят токен OUT выключенной управляющей точки OUT.
— Бит 3 **STUPM**: Фаза SETUP управляющей точки завершена.
— Бит 2 Резерв. не трогать.
— Бит 1 **EPDM**: Конечная точка выключена
— Бит 0 **XFRCM**: Передача завершена

Регистр прерываний всех конечных точек устройства (OTG_FS_DAINT)

Смещение адреса: 0x818

По сбросу: 0x0000 0000

Регистр `OTG_FS_DAINT` содержит прерывания всех конечных OUT и IN точек устройства. Прерывания разрешаются битами `OEPINT` и `IEPINT` в регистре `OTG_FS_GINTSTS`. Для двунаправленных точек используются два бита. Биты в этом регистре ставятся и снимаются соответствующими битами в регистрах `OTG_FS_DIEPINTx`/`OTG_FS_DOEPINTx`.



— Биты 31:16 **OEPINT**: Прерывания конечных точек OUT
Для точек OUT:
Бит 16 для OUT конечной точки 0, бит 19 для OUT конечной точки 3
— Биты 15:0 **IEPINT**: Прерывания конечных точек IN
Для точек IN:
Бит 0 для IN конечной точки 0, бит 3 для IN конечной точки 3.

Регистр маски прерываний всех конечных точек устройства (OTG_FS_DAINTMSK)

Смещение адреса: 0x81C

По сбросу: 0x0000 0000

Маскирует прерывания по регистру OTG_FS_DAINT.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OEPM																IEPM															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Маска прерывания: 0 - Нельзя, 1 - Можно.

— Биты 31:16 **OEPINTM**: Прерывания конечных точек OUT

Для точек OUT:

Бит 16 для OUT конечной точки 0, бит 19 для OUT конечной точки 3

— Биты 15:0 **IEPINTM**: Прерывания конечных точек IN

Для точек IN:

Бит 0 для IN конечной точки 0, бит 3 для IN конечной точки 3.

Регистр времени разряда V_{BUS} устройства (OTG_FS_DVBUSDIS)

Смещение адреса: 0x828

По сбросу: 0x0000 17D7

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																VBUSDT															
																rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:16 Резерв, не трогать.

— Биты 15:0 **VBUSDT**: Время разряда V_{BUS} устройства после импульса во время SRP.

Вычисляется в тактах PHY / 1 024. Зависит от нагрузки V_{BUS}, можно подстраивать.

Регистр времени импульса V_{BUS} устройства (OTG_FS_DVBUSPULSE)

Смещение адреса: 0x82C

По сбросу: 0x0000 05B8

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				DVBUSP											
																				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:12 Резерв, не трогать.

— Биты 11:0 **DVBUSP:** Время импульса V_{BUS} устройства во время SRP.

Вычисляется в тактах PHY / 1 024.

Регистр маски прерываний пустого FIFO конечных точек IN (OTG_FS_DIEPEMPMSK)

Смещение адреса: 0x834

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																INEPTXFEM															
																rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Маска прерывания: 0 - Нельзя, 1 - Можно.

— Биты 31:16 Резерв, не трогать.

— Биты 15:0 **INEPTXFEM**: TxFIFO конечной точки IN пуст

Маскируют биты прерываний по TXFE регистров OTG_FS_DIEPINTx.

Бит 0 для IN конечной точки 0, бит 3 для IN конечной точки 3.

Регистр управления конечной точки IN 0 (OTG_FS_DIEPCTL0)

Смещение адреса: 0x900

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPENA	EPDIS	Reserved		SNAK	CNAK	TXFNUM				STALL	Reserved	EPTYP		NAKSTS	Reserved	USBAEP	Reserved													MPSIZ	
r	r			w	w	rw	rw	rw	rw	rs		r	r	r		r														rw	rw

— Бит 31 **EPENA**: Включение конечной точки
Установка бита запускает передачу данных конечной точки 0.
Снимается ядром перед выдачей прерываний:
– Точка выключена
– Передача завершена

— Бит 30 **EPDIS**: Выключение конечной точки
Установка бита останавливает даже незавершённую передачу данных конечной точки 0. Далее надо дождаться прерывания выключения точки. Снимается ядром перед выдачей прерывания.
Останавливать можно только включённую конечную точку.

— Биты 29:28 Резерв, не трогать.

— Бит 27 **SNAK**: Установка ответа NAK
Программно задаёт выдачу NAK этой точки. Ядро может ставить его после приёма пакета SETUP.

— Бит 26 **CNAK**: Очистка NAK записью в этот бит

— Биты 25:22 **TXFNUM**: Номер TxFIFO точки IN 0.

— Бит 21 **STALL**: Ответ STALL
Ставится программно, снимается ядром после приёма пакета SETUP. Имеет приоритет перед битами NAK, Глобального IN NAK и Глобального OUT NAK.

— Бит 20 Резерв, не трогать.

— Биты 19:18 **EPTYP**: Тип конечной точки, ’00’ для управляющей точки.

— Бит 17 **NAKSTS**: Состояние NAK
0: Не-NAK ответы на основании состояния FIFO
1: Ответы NAK.
При стоящем бите ядро останавливает передачи данных даже при их наличии в TxFIFO. На пакет SETUP ядро посылает ACK независимо от состояния этого бита.

— Бит 16 Резерв, не трогать.

— Бит 15 **USBAEP**: Активная конечная точка USB
Всегда стоит, точка 0 активна во всех конфигурациях и интерфейсах.

— Биты 14:2 Резерв, не трогать.

— Биты 1:0 **MPSIZ**: Максимальный размер пакета
00: 64 байта
01: 32 байта
10: 16 байт
11: 8 байт

Регистр управления конечной точки-x устройства (OTG_FS_DIEPCTLx) (x = 1..3, где x = Номер_конечной_точки)

Смещение адреса: 0x900 + (Endpoint_number × 0x20)

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				Stall	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ											
rs	rs	w	w	w	w	rw	rw	rw	rw	rw/ rs		rw	rw	r	r	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Бит 31 **EPENA**: Включение конечной точки
Установка бита запускает передачу данных конечной точки.
Снимается ядром перед выдачей прерываний:
– Фаза SETUP завершена
– Точка выключена
– Передача завершена

— Бит 30 **EPDIS**: Выключение конечной точки
Установка бита останавливает даже незавершённую передачу данных конечной точки. Далее надо дождаться прерывания выключения точки. Снимается ядром перед выдачей прерывания.
Останавливать можно только включённую конечную точку.

— Бит 29 **SODDFRM**: Нечётный фрейм изохронной передачи IN и OUT (поле EONUM).

— Бит 28 **SD0PID:** Установить PID DATA0 прерывающих/групповых точек IN (поле DPID).

SEVNFRM: Чётный фрейм изохронной передачи IN (поле EONUM)

— Бит 27 **SNAK**: Установить NAK

Программно задаёт выдачу NAK этой точки. Ядро может ставить его после завершения передачи точки OUT или приёма пакета SETUP.

— Бит 26 **CNAK:** Очистить бит NAK

— Биты 25:22 **TXFNUM**: Номер TxFIFO конечной точки IN.

Активные точки IN должны иметь разные FIFO.

— Бит 21 **STALL**: Ответ STALL

Для не-управляющих, не-изохронных точек IN (доступ rw).

Программно останавливает передачи от хоста точке. Имеет приоритет перед битами NAK, Глобального IN NAK и Глобального OUT NAK. Снимается только программно.

Для управляющих точек (доступ rs).

Ставится программно, снимается ядром после приёма пакета SETUP. Имеет приоритет перед битами NAK, Глобального IN NAK и Глобального OUT NAK. На пакет SETUP ядро посылает ACK независимо от состояния этого бита.

— Бит 20 Резерв, не трогать.

— Биты 19:18 **EPTYP**: Тип конечной точки

00: Управляющая

01: Изохронная

10: Групповая

11: Прерывающая

— Бит 17 **NAKSTS**: Состояние NAK

0: Не-NAK ответы на основании состояния FIFO

1: Ответы NAK.

При стоящем бите ядро останавливает передачи данных даже при их наличии в TxFIFO. На пакет SETUP ядро посылает ACK независимо от состояния этого бита.

При стоящем бите:

Для не-изохронных точек IN: Ядро останавливает передачи по точке IN даже при данных в TxFIFO.

Для изохронных точек IN: Ядро посылает пакет данных нулевой длины даже при данных в TxFIFO.

На пакет SETUP ядро посылает ACK независимо от состояния этого бита.

— Бит 16 **EONUM**: Чётный/Нечётный фрейм изохронных точек IN.

Устанавливается записью в биты SEVNFRM и SODDFRM этого регистра.

0: Чётный

1: Нечётный

DPID: PID первого пакета данных прерывающей/групповой конечной точки

Устанавливается записью в бит SD0PID:

0: DATA0

1: DATA1

— Бит 15 **USBAEP**: Точка USB активна

По сбросу USB ядро снимает этот бит для всех точек, кроме EP 0. Ставить надо после завершения исполнения команд **SetConfiguration** и **SetInterface**.

— Биты 14:11 Резерв, не трогать.

— Биты 10:0 **MPSIZ**: Максимальный размер пакета в байтах

Регистр управления конечной точкой OUT 0 (OTG_FS_DOEPCTL0)

Смещение адреса: 0xB00

По сбросу: 0x0000 8000

31	30	29 28	27	26	25 24 23 22	21	20	19 18	17	16	15	14 13 12 11 10 9 8 7 6 5 4 3 2	1 0
EPENA	EPDIS	Reserved	SNAK	CNAK	Reserved	Stall	SNPM	EPTYP	NAKSTS	Reserved	USBAEP	Reserved	MPSIZ
w	r		w	w		rs	rw	r r	r		r		r r

— Бит 31 **EPENA**: Включить конечную точку

Установка бита запускает передачу данных конечной точки 0.

Снимается ядром перед выдачей прерываний:

– Фаза SETUP завершена
– Точка выключена
– Передача завершена

— Бит 30 **EPDIS**: Выключение конечной точки
Программно не выключается.
— Биты 29:28 Резерв, не трогать.
— Бит 27 **SNAK**: Установка NAK
Программно задаёт выдачу NAK этой точки. Ядро может ставить его после завершения передачи точки OUT или приёма пакета SETUP.
— Бит 26 **CNAK**: Очистка NAK
— Биты 25:22 Резерв, не трогать.
— Бит 21 **STALL**: Ответ STALL
Ставится программно, снимается ядром после приёма пакета SETUP. Имеет приоритет перед битами NAK и Глобального OUT NAK. На пакет SETUP ядро посылает ACK независимо от состояния этого бита.
— Бит 20 **SNPM**: Режим Snoop (Подглядывания)
В этом режиме ядро не проверяет правильность пакетов OUT перед записью в память.
— Биты 19:18 **PTYP**: Тип конечной точки, ноль для управляющей.
— Бит 17 **NAKSTS**: Состояние NAK
0: Не-NAK ответы на основании состояния FIFO
1: Ответы NAK.
При стоящем бите ядро останавливает приём данных даже при наличии места в RxFIFO. На пакет SETUP ядро посылает ACK независимо от состояния этого бита.
— Бит 16 Резерв, не трогать.
— Бит 15 **USBAEP**: Активная конечная точка, всегда равен 1.
— Биты 14:2 Резерв, не трогать.
— Бит 1:0 **MPSIZ**: Максимальный размер пакета точки OUT
00: 64 байта
01: 32 байта
10: 16 байт
11: 8 байт

Регистр управления конечной точки-x (OTG_FS_DOEPCTLx)
(x = 1..3, где x = Номер_конечной_точки)

Смещение адреса точки OUT: 0xB00 + (Endpoint_number × 0x20)

По сбросу: 0x0000 0000

31	30	29	28	27	26	25–22	21	20	19–18	17	16	15	14–11	10–0
EPENA	EPDIS	SODDFRM/SD1PID	SD0PID/SEVNFRM	SNAK	CNAK	Reserved	Stall	SNPM	EPTYP	NAKSTS	EONUM/DPID	USBAEP	Reserved	MPSIZ
rs	rs	w	w	w	w		rw/ rs	rw	rw	r	r	rw		rw

— Бит 31 **EPENA**: Включение конечной точки IN и OUT.
Установка бита запускает передачу данных конечной точки.
Снимается ядром перед выдачей прерываний:
– Фаза SETUP завершена
– Точка выключена
– Передача завершена
— Бит 30 **EPDIS**: Выключение конечной точки
Установка бита останавливает даже незавершённую передачу/приём данных конечной точки. Далее надо дождаться прерывания выключения точки. Снимается ядром перед выдачей прерывания. Останавливать можно только включённую конечную точку.
— Бит 29 **SD1PID**: Установить PID DATA1 (DPID) прерывающих/групповых точек IN и OUT.
SODDFRM: Установить нечётный фрейм изохронных точек IN и OUT (поле EONUM)
— Бит 28 **SD0PID**: Установить PID DATA0 (DPID) прерывающих/групповых точек OUT

SEVNFRM: Установить чётный фрейм изохронных точек OUT (поле EONUM)

— Бит 27 **SNAK**: Установка NAK

Программно задаёт выдачу NAK этой точки. Ядро может ставить его после завершения передачи точки OUT или приёма пакета SETUP.

— Бит 26 **CNAK**: Очистка NAK

— Биты 25:22 Резерв, не трогать.

— Бит 21 **STALL**: Ответ STALL

Для не-управляющих, не-изохронных точек IN (доступ rw).

Программно останавливает все передачи от хоста точке. Имеет приоритет перед битами NAK, Глобального IN NAK и Глобального OUT NAK. Снимается только программно.

Для управляющих точек (доступ rs).

Ставится программно, снимается ядром после приёма пакета SETUP. Имеет приоритет перед битами NAK, Глобального IN NAK и Глобального OUT NAK. На пакет SETUP ядро посылает ACK независимо от состояния этого бита.

— Бит 20 **SNPM**: Режим Snoop (Подглядывания)

В этом режиме ядро не проверяет правильность пакетов OUT перед записью в память.

— Биты 19:18 **EPTYP**: Тип конечной точки

00: Управляющая

01: Изохронная

10: Групповая

11: Прерывающая

— Бит 17 **NAKSTS**: Состояние NAK

0: Не-NAK ответы на основании состояния FIFO

1: Ответы NAK.

При стоящем бите ядро останавливает приём данных даже при наличии места в RxFIFO. На пакет SETUP ядро посылает ACK независимо от состояния этого бита.

— Бит 16 **EONUM**: Чётный/Нечётный фрейм изохронных точек IN.

Устанавливается записью в биты SEVNFRM и SODDFRM этого регистра.

0: Чётный

1: Нечётный

DPID: PID первого пакета данных прерывающей/групповой конечной точки

Устанавливается записью в бит SD0PID:

0: DATA0

1: DATA1

— Бит 15 **USBAEP**: Активная конечная точка USB

По сбросу USB ядро снимает этот бит для всех точек, кроме EP 0. Ставить надо после завершения исполнения команд **SetConfiguration** и **SetInterface**.

— Биты 14:11 Резерв, не трогать.

— Биты 10:0 **MPSIZ**: Максимальный размер пакета в байтах

Регистр прерываний конечной точки-x (OTG_FS_DIEPINTx) (x = 0..3, где x = Номер_конечной_точки)

Смещение адреса: 0x908 + (Endpoint_number × 0x20)

По сбросу: 0x0000 0080

Читать можно только при стоящем бите `IEPINT` в `OTG_FS_GINTSTS`, но сначала надо узнать номер точки по регистру `OTG_FS_DAINT`. Снятие бита в этом регистре чистит соответствующие биты в регистрах `OTG_FS_DAINT` и `OTG_FS_GINTSTS`.

31 … 8	7	6	5	4	3	2	1	0
Reserved	TXFE	INEPNE	Reserved	ITTXFE	TOC	Reserved	EPDISD	XFRC
	r	rc_w1/rw		rc_w1	rc_w1		rc_w1	rc_w1

— Биты 31:8 Резерв, не трогать.

— Бит 7 **TXFE**: Передающий FIFO полу- или полностью пуст

Уровень пустоты TxFIFO задаётся битом TXFELVL в регистре OTG_FS_GAHBCFG.

— Бит 6 **INEPNE**: NAK конечной точки IN в действии
Снимается записью бита CNAK в регистре OTG_FS_DIEPCTLx. Установка бита вызывает прерывание. По шине USB не обязательно пойдёт NAK, бит STALL приоритетнее.
— Бит 5 Резерв, не трогать.
— Бит 4 **ITTXFE**: Принят токен IN при пустом не-периодическом TxFIFO
Выставляется прерывание этой точки.
— Бит 3 **TOC**: Таймаут управляющей точки IN.
— Бит 2 Резерв, не трогать.
— Бит 1 **EPDISD**: Прерывание выключенной конечной точки
— Бит 0 **XFRC**: Прерывание конца передачи по AHB или USB.

Регистр прерываний конечной точки-x (OTG_FS_DOEPINTx) (x = 0..3, где x = Номер_конечной_точки)

Смещение адреса: 0xB08 + (Endpoint_number × 0x20)

По сбросу: 0x0000 0080

Читать можно только при стоящем бите `OEPINT` в регистре `OTG_FS_GINTSTS`, но сначала надо узнать номер точки по регистру `OTG_FS_DAINT`. Снятие бита в этом регистре чистит соответствующие биты в регистрах `OTG_FS_DAINT` и `OTG_FS_GINTSTS`.

31 … 7	6	5	4	3	2	1	0
Reserved	B2BSTUP	Reserved	OTEPDIS	STUP	Reserved	EPDISD	XFRC
	rc_w1 /rw		rc_w1	rc_w1		rc_w1	rc_w1

— Биты 31:7 Резерв, не трогать.
— Бит 6 **B2BSTUP**: Принята череда (больше 3) пакетов SETUP управляющей точки OUT.
— Бит 5 Резерв, не трогать.
— Бит 4 **OTEPDIS**: Принят пакет OUT выключенной управляющей точки (с прерыванием)
— Бит 3 **STUP**: Фаза SETUP управляющей точки OUT завершена (с прерыванием)
— Бит 2 Резерв, не трогать.
— Бит 1 **EPDISD**: Прерывание выключения конечной точки
— Бит 0 **XFRC**: Прерывание конца передачи по AHB или USB.

Регистр размера передачи конечной точки IN 0 (OTG_FS_DIEPTSIZ0)

Смещение адреса: 0x910

По сбросу: 0x0000 0000

Писать надо до включения точки 0. После этого можно только читать.

31 … 21	20	19	18 … 7	6	5	4	3	2	1	0
Reserved	PKTCNT		Reserved	XFRSIZ						
	rw	rw		rw	rw	rw	rw	rw	rw	rw

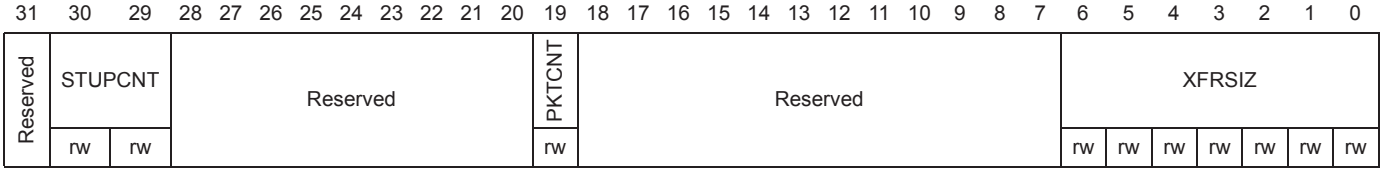
— Биты 31:21 Резерв, не трогать.
— Биты 20:19 **PKTCNT**: Счётчик общего числа пакетов передачи
Декрементируется при каждом чтении из TxFIFO.
— Биты 18:7 Резерв, не трогать.
— Биты 6:0 **XFRSIZ**: Размер передачи в байтах (с прерыванием после исчерпания)
Уменьшается при каждой записи в TxFIFO.

Регистр размера передачи конечной точки OUT 0 (OTG_FS_DOEPTSIZ0)

Смещение адреса: 0xB10

По сбросу: 0x0000 0000

Писать надо до включения точки 0. После этого можно только читать.



— Бит 31 Резерв, не трогать.

— Биты 30:29 **STUPCNT**: Счётчик череды пакетов SETUP

01: 1 пакет

10: 2 пакета

11: 3 пакета

— Биты 28:20 Резерв, не трогать.

— Бит 19 **PKTCNT**: Счётчик пакетов

Обнуляется при записи в RxFIFO.

— Биты 18:7 Резерв, не трогать.

— Биты 6:0 **XFRSIZ**: Размер передачи в байтах (с прерыванием после исчерпания)

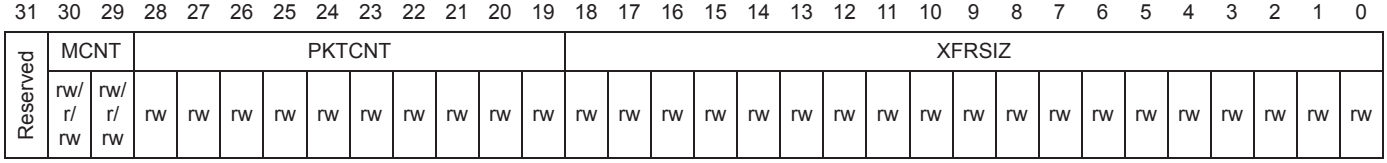
Уменьшается при каждом чтении из RxFIFO.

Регистр размера передачи конечной точки-x (OTG_FS_DIEPTSIZx) (x = 1..3, где x = Номер_конечной_точки)

Смещение адреса: 0xB10 + (Endpoint_number × 0x20)

По сбросу: 0x0000 0000

Писать надо до включения точки 0. После этого можно только читать.



— Бит 31 Резерв, не трогать.

— Биты 30:29 **MCNT**: Многосчётчик пакетов в периодическом фрейме IN.

01: 1 пакет

10: 2 пакета

11: 3 пакета

— Биты 28:19 **PKTCNT**: Счётчик общего числа пакетов передачи

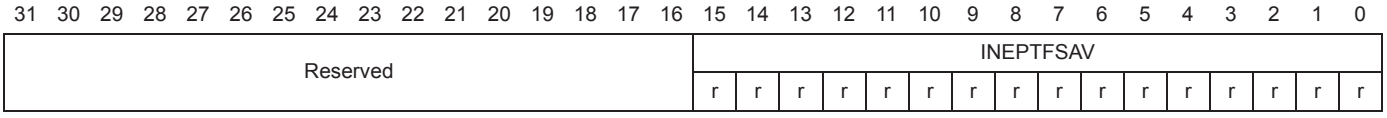
Декрементируется при каждом чтении из TxFIFO.

— Биты 18:0 **XFRSIZ**: Размер передачи в байтах

Уменьшается при каждой записи в TxFIFO.

Регистр состояния FIFO конечной точки IN (OTG_FS_DTXFSTSx) (x = 0..3, где x = Номер_конечной_точки)

Смещение адреса точки IN: 0x918 + (Endpoint_number × 0x20)



— Биты 31:16 Резерв, не трогать.

— Биты 15:0 **INEPTFSAV**: Доступное место в TxFIFO точки IN в 32-бит словах

0x0: TxFIFO полон

0x1: 1 слово

0x2: 2 слова

0xn: n слов

Иное: Резерв

Регистр размера передачи конечной точки-x OUT (OTG_FS_DOEPTSIZx) (x = 1..3, где x = Номер_конечной_точки)

Смещение адреса: 0xB10 + (Endpoint_number × 0x20)

По сбросу: 0x0000 0000

Писать надо до включения точки 0. После этого можно только читать.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	RXDPID/STUPCNT		PKTCNT										XFRSIZ																		
	rw/r/rw	rw/r/rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Бит 31 Резерв, не трогать.

— Биты 30:29 **RXDPID**: PID принятых данных изохронной точки OUT.

00: DATA0
01: DATA2
10: DATA1
11: MDATA

STUPCNT: Максимальное число пакетов SETUP управляющей точки OUT.

01: 1 пакет
10: 2 пакета
11: 3 пакета

— Биты 28:19 **PKTCNT**: Счётчик общего числа пакетов передачи
Декрементируется при каждой записи в RxFIFO.

— Биты 18:0 **XFRSIZ**: Размер передачи в байтах (с прерыванием после исчерпания)
Уменьшается при каждом чтении из RxFIFO.

34.16.5. Регистр питания и тактов OTG_FS (OTG_FS_PCGCCTL)

Смещение адреса: 0xE00

По сбросу: 0x0000 0000

Доступен в обоих режимах.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																											PHYSUSP	Reserved			GATEHCLK	STPPCLK
																											rw				rw	rw

— Биты 31:5 Резерв, не трогать.

— Бит 4 **PHYSUSP**: PHY остановлен

— Биты 3:2 Резерв, не трогать.

— Бит 1 **GATEHCLK**: Разрешение подачи HCLK
Ставится и снимается программно.

— Бит 0 **STPPCLK**: Остановка тактов PHY
Ставится и снимается программно.

34.16.6. Карта регистров OTG_FS

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**OTG_FS_GOTGCTL**	Reserved												BSVLD	ASVLD	DBCT	CIDSTS	Reserved				DHNPEN	HSHNPEN	HNPRQ	HNGSCS	Reserved						SRQ	SRQSCS
	Reset value													0	0	0	1					0	0	0	0							0	0
0x004	**OTG_FS_GOTGINT**	Reserved												DBCDNE	ADTOCHG	HNGDET	Reserved							HNSSCHG	SRSSCHG	Reserved					SEDET	Res.	
	Reset value													0	0	0								0	0						0		
0x008	**OTG_FS_GAHBCFG**	Reserved																							PTXFELVL	TXFELVL	Reserved						GINTMSK
	Reset value																								0	0							0
0x00C	**OTG_FS_GUSBCFG**	CTXPKT	FDMOD	FHMOD	Reserved									TRDT										HNPCAP	SRPCAP	Reserved	PHYSEL	Reserved				TOCAL		
	Reset value													0	1	0	1							0	0		1					0	0	0
0x010	**OTG_FS_GRSTCTL**	AHBIDL	Reserved																					TXFNUM					RXFFLSH	Reserved	FCRST	HSRST	CSRST
	Reset value	1																						0	0	0		0	0	0	0	0	0
0x014	**OTG_FS_GINTSTS**	WKUINT	SRQINT	DISCINT	CIDSCHG	Reserved	PTXFE	HCINT	HPRTINT	Reserved		IPXFR/INCOMPISOOUT	IISOIXFR	OEPINT	IEPINT	Reserved		EOPF	ISOODRP	ENUMDNE	USBRST	USBSUSP	ESUSP	Reserved		GONAKEFF	GINAKEFF	NPTXFE	RXFLVL	SOF	OTGINT	MMIS	CMOD
	Reset value	0	0	0	0		1	0	0			0	0	0	0			0	0	0	0	0	0			0	0	1	0	0	0	0	0
0x018	**OTG_FS_GINTMSK**	WUIM	SRQIM	DISCINT	CIDSCHGM	Reserved	PTXFEM	HCIM	PRTIM	Reserved		IPXFRM/IISOOXFRM	IISOIXFRM	OEPINT	IEPINT	EPMISM	Reserved	EOPFM	ISOODRPM	ENUMDNEM	USBRST	USBSUSPM	ESUSPM	Reserved		GONAKEFFM	GINAKEFFM	NPTXFEM	RXFLVLM	SOFM	OTGINT	MMISM	Reserved
	Reset value	0	0	0	0		0	0	0			0	0	0	0	0		0	0	0	0	0	0			0	0	0	0	0	0	0	
0x01C	**OTG_FS_GRXSTSR (host mode)**	Reserved												PKTSTS				DPID				BCNT									CHNUM		
	Reset value													0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	OTG_FS_GRXSTSR (Device mode)	Reserved							FRMNUM					PKTSTS				DPID				BCNT									EPNUM		
	Reset value								0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x020	**OTG_FS_GRXSTSR (host mode)**	Reserved											PKTSTS				DPID		BCNT											CHNUM			
	Reset value												0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	OTG_FS_GRXSTSPR (Device mode)	Reserved							FRMNUM				PKTSTS				DPID		BCNT											EPNUM			
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	**OTG_FS_GRXFSIZ**	Reserved																	RXFD														
	Reset value																		0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0x028	**OTG_FS_HNPTXFSIZ/ OTG_FS_DIEPTXF0**	NPTXFD/TX0FD																	NPTXFSA/TX0FSA														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0x02C	**OTG_FS_HNPTXSTS**	Res.	NPTXQTOP							NPTQXSAV									NPTXFSAV														
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	1	0	0		0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0x038	**OTG_FS_GCCFG**	Reserved											NOVBUSSENS	SOFOUTEN	VBUSBSEN	VBUSASEN	Reserved	.PWRDWN	Reserved															
	Reset value												0	0	0	0		0															
0x03C	**OTG_FS_CID**	PRODUCT_ID																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	1	0	0	1	0	0	0	0	0	0	0	0
0x100	**OTG_FS_HPTXFSIZ**	PTXFSIZ																	PTXSA														
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0		0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x104	**OTG_FS_DIEPTXF1**	INEPTXFD																	INEPTXSA														
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0x108	**OTG_FS_DIEPTXF2**	INEPTXFD																	INEPTXSA														
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0		0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x10C	**OTG_FS_DIEPTXF3**	INEPTXFD																	INEPTXSA														
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0		0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x400	**OTG_FS_HCFG**	Reserved																													FSLSS	FSLSPCS	
	Reset value																														0	0	0
0x404	**OTG_FS_HFIR**	Reserved																	FRIVL														
	Reset value																		1	1	1	0	1	0	1	0	0	1	1	0	0	0	0	0
0x408	**OTG_FS_HFNUM**	FTREM																	FRNUM														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	1	1	1	1	1	1	1	1	1	1	1	1	1

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x410	**OTG_FS_HPTXSTS**	PTXQTOP								PTXQSAV								PTXFSAVL															
	Reset value	0	0	0	0	0	0	0	0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
0x414	**OTG_FS_HAINT**	Reserved																HAINT															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x418	**OTG_FS_HAINTMSK**	Reserved																HAINTM															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x440	**OTG_FS_HPRT**	Reserved													PSPD	PTCTL					PPWR	PLSTS	Reserved	PRST	PSUSP	PRES	POCCHNG	POCA	PENCHNG	PENA	PCDET	PCSTS	
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x500	**OTG_FS_HCCHAR0**	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x520	**OTG_FS_HCCHAR1**	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x540	**OTG_FS_HCCHAR2**	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x560	**OTG_FS_HCCHAR3**	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x580	**OTG_FS_HCCHAR4**	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5A0	**OTG_FS_HCCHAR5**	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C0	**OTG_FS_HCCHAR6**	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5E0	**OTG_FS_HCCHAR7**	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x508	**OTG_FS_HCINT0**	Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x528	**OTG_FS_HCINT 1**	Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x548	**OTG_FS_HCINT 2**	Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x568	**OTG_FS_HCINT 3**	Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x588	**OTG_FS_HCINT 4**	Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x5A8	**OTG_FS_HCINT 5**	Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x5C8	**OTG_FS_HCINT 6**	Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x5E8	**OTG_FS_HCINT 7**	Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x50C	**OTG_FS_HCINT MSK0**	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	Reserved	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	Reset value																						0	0	0	0		0	0	0		0	0
0x52C	**OTG_FS_HCINT MSK1**	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	Reserved	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	Reset value																						0	0	0	0		0	0	0		0	0
0x54C	**OTG_FS_HCINT MSK2**	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	Reserved	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	Reset value																						0	0	0	0		0	0	0		0	0
0x56C	**OTG_FS_HCINT MSK3**	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	Reserved	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	Reset value																						0	0	0	0		0	0	0		0	0
0x58C	**OTG_FS_HCINT MSK4**	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	Reserved	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	Reset value																						0	0	0	0		0	0	0		0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x5AC	**OTG_FS_HCINTMSK5**	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	Reserved	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	Reset value																						0	0	0	0		0	0	0		0	0
0x5CC	**OTG_FS_HCINTMSK6**	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	Reserved	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	Reset value																						0	0	0	0		0	0	0		0	0
0x5EC	**OTG_FS_HCINTMSK7**	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	Reserved	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	Reset value																						0	0	0	0		0	0	0		0	0
0x510	**OTG_FS_HCTSIZ0**	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x530	**OTG_FS_HCTSIZ1**	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x550	**OTG_FS_HCTSIZ2**	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x570	**OTG_FS_HCTSIZ3**	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x590	**OTG_FS_HCTSIZ4**	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5B0	**OTG_FS_HCTSIZ5**	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5D0	**OTG_FS_HCTSIZ6**	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5F0	**OTG_FS_HCTSIZ7**	Reserved	DPID		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x800	**OTG_FS_DCFG**	Reserved																					PFIVL			DAD					Reserved	NZLSOHSK	DSPD	
	Reset value																						0	0	0	0	0	0	0	0		0	0	0
0x804	**OTG_FS_DCTL**	Reserved																					POPRGDNE	CGONAK	SGONAK	CGINAK	SGINAK	TCTL			GONSTS	GINSTS	SDIS	RWUSIG
	Reset value																						0	0	0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x808	**OTG_FS_DSTS**	Reserved										FNSOF														Reserved				EERR	ENUMSPD		SUSPSTS
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0
0x810	**OTG_FS_DIEPMSK**	Reserved																		NAKM	Reserved							INEPNEM	INEPNMM	ITTXFEMSK	TOM	Reserved	EPDM	XFRCM
	Reset value																			0								0	0	0	0		0	0
0x814	**OTG_FS_DOEPMSK**	Reserved																		NAKMSK	BERRM	Reserved				OUTPKTERRM	Reserved		STSPHSRXM	OTEPDM	STUPM	Reserved	EPDM	XFRCM
	Reset value																			0	0					0			0	0	0		0	0
0x818	**OTG_FS_DAINT**	OEPINT																IEPINT															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x81C	**OTG_FS_DAINTMSK**	OEPM																IEPM															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x828	**OTG_FS_DVBUSDIS**	Reserved																VBUSDT															
	Reset value																	0	0	0	1	0	1	1	1	1	1	0	1	0	1	1	1
0x82C	**OTG_FS_DVBUSPULSE**	Reserved																				DVBUSP											
	Reset value																					0	1	0	1	1	0	1	1	1	0	0	0
0x834	**OTG_FS_DIEPEMPMSK**	Reserved																INEPTXFEM															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x900	**OTG_FS_DIEPCTL0**	EPENA	EPDIS	Reserved			SNAK	CNAK	TXFNUM				STALL	Reserved	EPTYP		NAKSTS	Reserved	USBAEP	Reserved												MPSIZ	
	Reset value	0	0				0	0	0	0	0	0	0		0	0	0		1													0	0
0x918	**TG_FS_DTXFSTS0**	Reserved																INEPTFSAV															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x920	**OTG_FS_DIEPCTL1**	EPENA	EPDIS	SODDFRM/SD1PID	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM					STALL	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved												MPSIZ	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0							0	0	0	0	0	0	0	0
0x938	**TG_FS_DTXFSTS1**	Reserved																INEPTFSAV															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x940	**OTG_FS_DIEPCTL2**	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				STALL	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0x958	**TG_FS_DTXFSTS2**	Reserved																INEPTFSAV															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x960	**OTG_FS_DIEPCTL3**	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				STALL	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0x978	**TG_FS_DTXFSTS3**	Reserved																INEPTFSAV															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0xB00	**OTG_FS_DOEPCTL0**	EPENA	EPDIS	Reserved		SNAK	CNAK	Reserved				STALL	SNPM	EPTYP		NAKSTS	Reserved	USBAEP	Reserved													MPSIZ	
	Reset value	0	0			0	0					0	0	0	0	0		1														0	0
0xB20	**OTG_FS_DOEPCTL1**	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	Reserved				STALL	SNPM	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0xB40	**OTG_FS_DOEPCTL2**	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	Reserved				STALL	SNPM	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0xB60	**OTG_FS_DOEPCTL3**	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	Reserved				STALL	SNPM	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0x908	**OTG_FS_DIEPINT0**	Reserved																		NAK	Reserved	PKTDRPSTS	Reserved			TXFE	INEPNE	INEPNM	ITTXFE	TOC	Reserved	EPDISD	XFRC
	Reset value																			0		0				1	0	0	0	0		0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x928	**OTG_FS_DIEPINT1**	Reserved																		NAK	Reserved	PKTDRPSTS	Reserved			TXFE	INEPNE	INEPNM	ITTXFE	TOC	Reserved	EPDISD	XFRC
	Reset value																			0		0				1	0	0	0	0		0	0
0x948	**OTG_FS_DIEPINT2**	Reserved																		NAK	Reserved	PKTDRPSTS	Reserved			TXFE	INEPNE	INEPNM	ITTXFE	TOC	Reserved	EPDISD	XFRC
	Reset value																			0		0				1	0	0	0	0		0	0
0x968	**OTG_FS_DIEPINT3**	Reserved																		NAK	Reserved	PKTDRPSTS	Reserved			TXFE	INEPNE	INEPNM	ITTXFE	TOC	Reserved	EPDISD	XFRC
	Reset value																			0		0				1	0	0	0	0		0	0
0xB08	**OTG_FS_DOEPINT0**	Reserved																		NAK	BERR	Reserved			OUTPKTERR	Reserved		STSPHSRX	OTEPDIS	STUP	Reserved	EPDISD	XFRC
	Reset value																			0	0				0			0	0	0		0	0
0xB28	**OTG_FS_DOEPINT1**	Reserved																		NAK	BERR	Reserved			OUTPKTERR	Reserved		STSPHSRX	OTEPDIS	STUP	Reserved	EPDISD	XFRC
	Reset value																			0	0				0			0	0	0		0	0
0xB48	**OTG_FS_DOEPINT2**	Reserved																		NAK	BERR	Reserved			OUTPKTERR	Reserved		STSPHSRX	OTEPDIS	STUP	Reserved	EPDISD	XFRC
	Reset value																			0	0				0			0	0	0		0	0
0xB68	**OTG_FS_DOEPINT3**	Reserved																		NAK	BERR	Reserved			OUTPKTERR	Reserved		STSPHSRX	OTEPDIS	STUP	Reserved	EPDISD	XFRC
	Reset value																			0	0				0			0	0	0		0	0
0x910	**OTG_FS_DIEPTSIZ0**	Reserved											PKTCNT		Reserved												XFRSIZ						
	Reset value												0	0													0	0	0	0	0	0	0
0x930	**OTG_FS_DIEPTSIZ1**	Reserved			PKTCNT										XFRSIZ																		
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x950	**OTG_FS_DIEPTSIZ2**	Reserved			PKTCNT										XFRSIZ																		
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x970	**OTG_FS_DIEPTSIZ3**	Reserved			PKTCNT										XFRSIZ																		
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xB10	**OTG_FS_DOEPTSIZ0**	Reserved	STUPCNT		Reserved									PKTCNT	Reserved												XFRSIZ						
	Reset value		0	0										0													0	0	0	0	0	0	0
0xB30	**OTG_FS_DOEPTSIZ1**	Reserved	RXDPID/STUPCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB50	**OTG_FS_DOEPTSIZ2**	Reserved	RXDPID/STUPCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB70	**OTG_FS_DOEPTSIZ3**	Reserved	RXDPID/STUPCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xE00	**OTG_FS_PCGCCTL**	Reserved																											PHYSUSP	Reserved		GATEHCLK	STPPCLK
	Reset value																																

34.17. Программная модель OTG_FS

34.17.1. Инициализация ядра

Сторона подключения кабеля (A или B) и режим контроллера показывается битом CMOD в регистре OTG_FS_GINTSTS).

Независимо от режимов хоста или устройства сначала пишем глобальные регистры:

1. В регистре OTG_FS_GAHBCFG:
   - Бит маски глобального прерывания GINTMSK = 1
   - RxFIFO не пуст (бит RXFLVL в OTG_FS_GINTSTS)
   - Уровень пустоты периодического TxFIFO
2. В регистре OTG_FS_GUSBCFG:
   - Бит разрешения HNP
   - Бит разрешения SRP
   - Поле калибровки таймаута FS
   - Поле времени обратной связи USB
3. Разрешаем маски в регистре OTG_FS_GINTMSK:
   - Маска прерывания OTG
   - Маска прерывания несовпадения режимов
4. По биту CMOD в OTG_FS_GINTSTS определяем режим контроллера OTG_FS (хост или устройство).

34.17.2. Инициализация хоста

Исполняем следующие шаги:

1. Демаскируем HPRTINT в регистре OTG_FS_GINTMSK
2. В регистре OTG_FS_HCFG ставим полную скорость
3. Битом PPWR в OTG_FS_HPRT включаем V_{BUS} на USB.
4. Ждём подключения устройства к хосту по прерыванию PCDET в OTG_FS_HPRT0.
5. Запускаем сброс битом PRST в регистре OTG_FS_HPRT.
6. Не меньше 10 ms ждём его завершения.
7. Снимаем бит PRST в регистре OTG_FS_HPRT.
8. Ждём прерывания PENCHNG в регистре OTG_FS_HPRT.

9. Узнаём скорость переучёта по биту PSPD в регистре OTG_FS_HPRT.
10. В регистр HFIR пишем нужное число для 1 такта PHY.
11. В соответствии с полученной скоростью устройства пишем поле FSLSPCS регистра OTG_FS_HCFG. Если оно изменилось, то надо сбросить порт.
12. В регистр OTG_FS_GRXFSIZ пишем размер приёмного FIFO.
13. В регистр OTG_FS_HNPTXFSIZ пишем размер и адрес начала не-периодического передающего FIFO.
14. В регистр OTG_FS_HNPTXFSIZ пишем размер и адрес начала периодического передающего FIFO.

Для работы с устройствами нужно инициализировать хоть один канал.

34.17.3. Инициализация устройства

После сброса по включению питания или изменении режима на устройство надо:

1. В регистре OTG_FS_DCFG определить:
   - Скорость устройства
   - Состояние ответа OUT ненулевой длины.
2. В регистре OTG_FS_GINTMSK демаскировать прерывания:
   - Сброс USB
   - Переучёт завершён
   - Ранний останов
   - Останов USB
   - SOF
3. Битом VBUSBSEN в OTG_FS_GCCFG разрешить контроль V_{BUS} устройства “B” и резистор подпорки линии DP.
4. Дождаться прерывания USBRST в регистре OTG_FS_GINTSTS. После получения этого прерывания сброс длится ещё 10 ms.

Ждём прерывания ENUMDNE в регистре OTG_FS_GINTSTS. Это конец сброса USB. Теперь по регистру OTG_FS_DSTS определяем скорость переучёта и инициализируем конечные точки.

Всё, устройство готово принимать пакеты SOF и выдавать управляющие передачи по точке 0.

34.17.4. Программная модель хоста

Инициализация канала

Выполняем шаги:

1. В регистре OTG_FS_GINTMSK демаскируем:
2. Прерывания канала
   - Пустого не-периодического передающего FIFO передач OUT или
   - Полупустого не-периодического передающего FIFO передач OUT.
3. В регистре OTG_FS_GINTMSK демаскируем прерывания выбранных каналов.
4. В регистре OTG_FS_GINTMSK демаскируем прерывания нужных условий передач.
5. В регистры OTG_FS_HCTSIZx пишем общий размер передач и ожидаемое число пакетов, включая короткие. В поле PID пишем начальный PID (первой для OUT или ожидаемой от IN).
6. В регистре OTG_FS_HCCHARx выбранного канала пишем характеристики устройства, вроде типа, скорости, направления и т.д.

Канал можно включать только при готовности к передаче или приёму данных.

Остановка канала

Выключить канал можно установкой битов CHDIS и CHENA регистра OTG_FS_HCCHARx. Хост сливает стоящие запросы, не прерывая запущенные передачи, и выдаёт прерывание остановки канала. Перед выделением канала для другой работы надо дождаться прерывания CHH в OTG_FS_HCINTx.

Перед выключением канала нужно убедиться в наличии хоть одной свободной строки в соответствующей очереди запросов (периодической или не-периодической). Запросы в очереди можно слить установкой бита CHDIS и очисткой бита CHENA регистра OTG_FS_HCCHARx.

Канал выключают при:

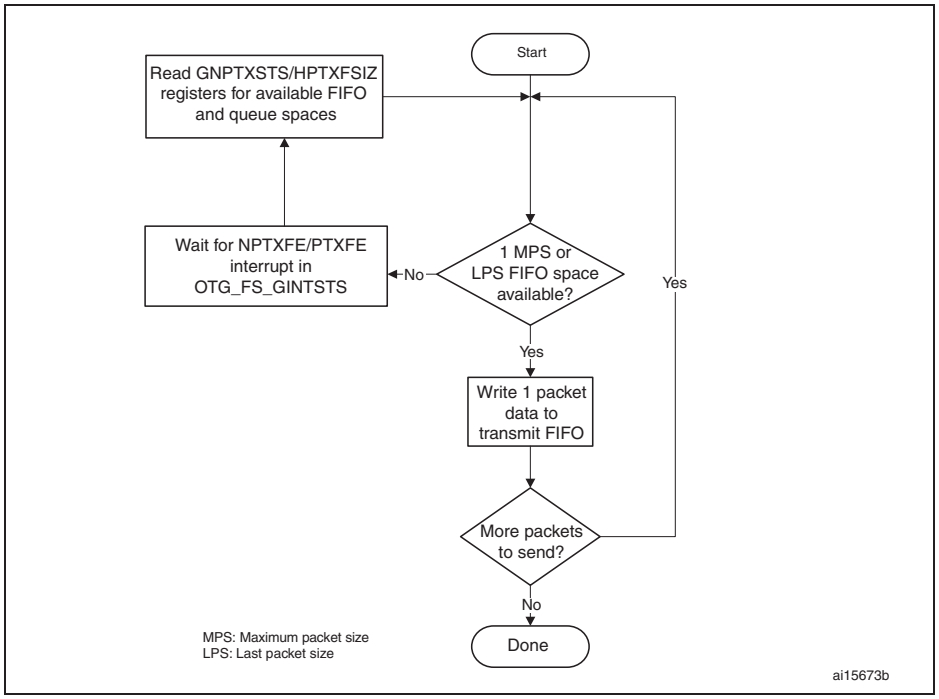
1. Прерываниях STALL, TXERR, BBERR или DTERR регистра OTG_FS_HCINTx. При этом такой же канал должен уметь принимать прерывания DTERR, NAK, DATA, TXERR.
2. Прерывании DISCINT регистра OTG_FS_GINTSTS. (выключаются все каналы).
3. При прекращении передач до их нормального завершения.

Модель работы

Всё делается с уже инициализированным каналом.

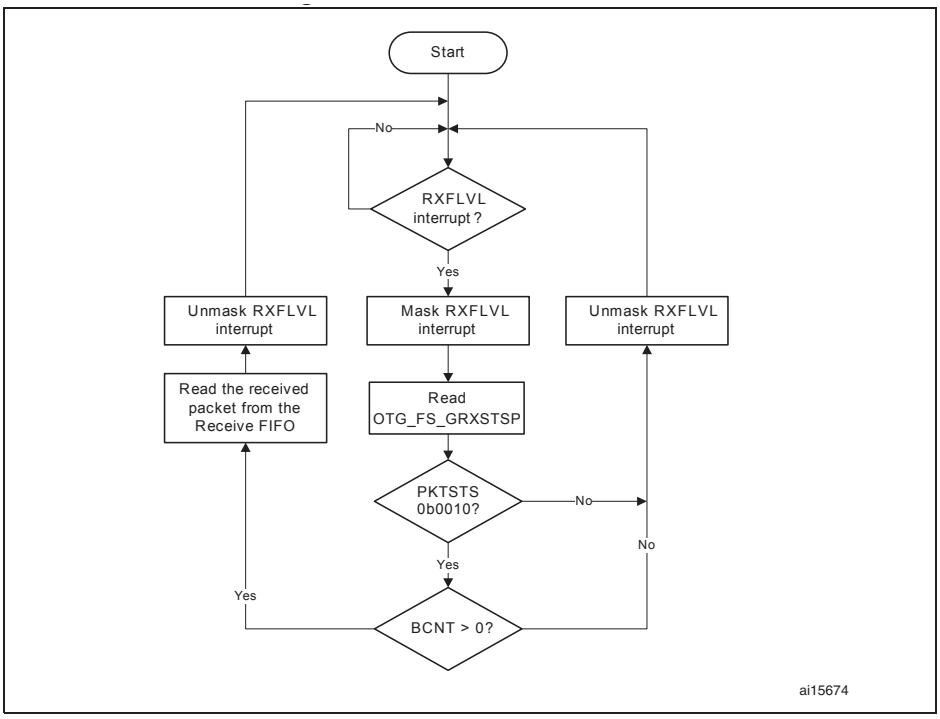
- **Запись в передающий FIFO**

Одновременно с записью последнего слова пакета, хост OTG_FS автоматически пишет запрос OUT в периодическую/не-периодическую очередь. Писать надо только при наличии свободного места в передающем FIFO. В FIFO пишут только словами. При некратной длине пакета он дополняется. Хост OTG_FS определяет конкретную длину пакета на основе установленных максимальной длины пакета и размера передачи.



- **Чтение приёмного FIFO**

Пакеты, отличные от пакетов данных IN (bx0010) должны игнорироваться.



• **Групповые и управляющие передачи OUT/SETUP**

На рисунке ниже передаются два пакета OUT и один пакет SETUP. Полагаем, что:

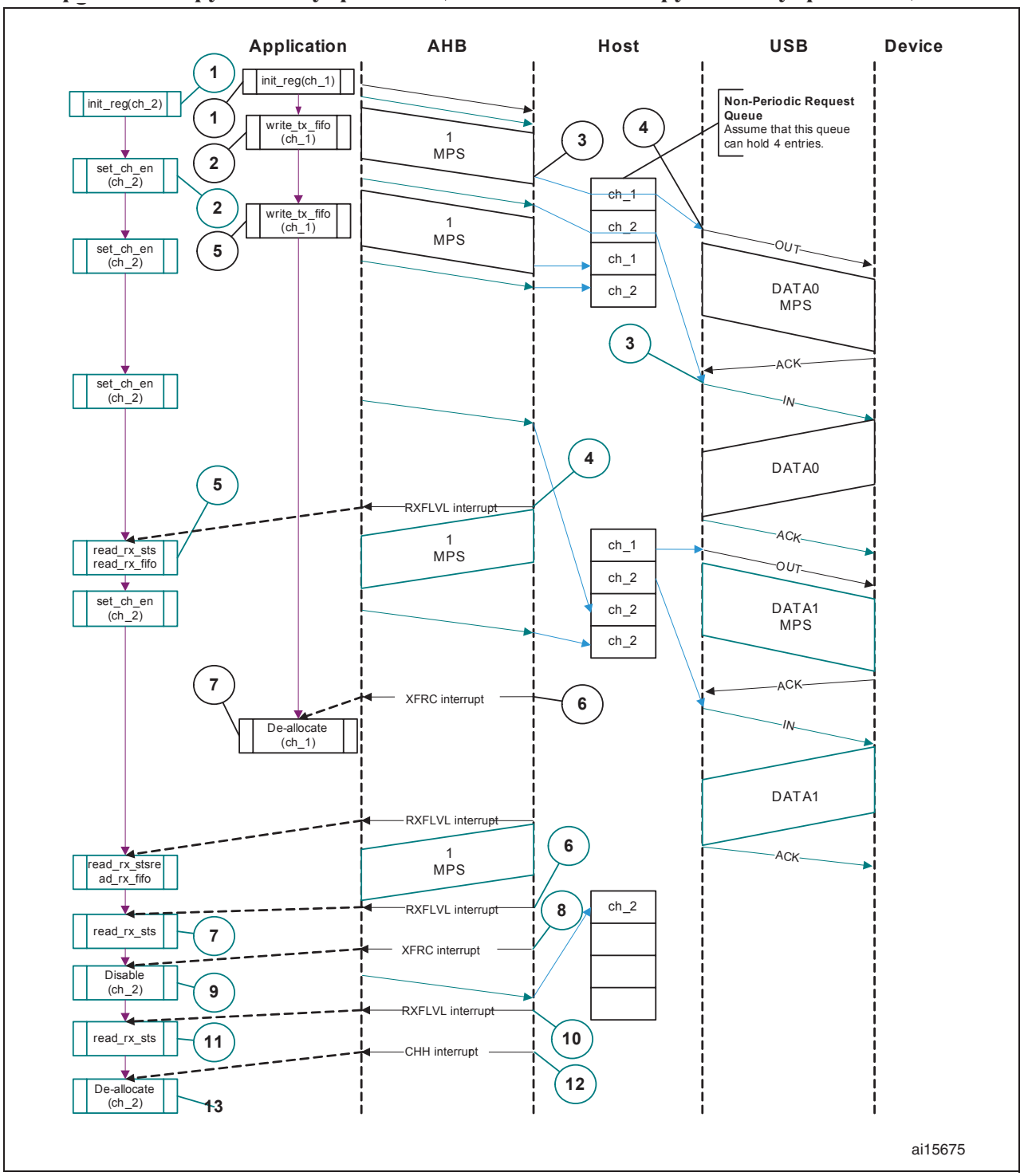
– Передаются два максимальных пакета (размер = 1024 байта).
– Передающий не-периодический FIFO может хранить два пакета (128 байт для FS).
– Глубина очереди не-периодических запросов = 4.

• **Нормальные групповые и управляющие операции OUT/SETUP**

Последовательность действий для канала 1 такова:

a) Инициализируем канал 1
b) Пишем первый пакет канала 1
c) Вместе с записью последнего слова хост пишет не-периодический запрос в очередь
d) После появления запроса в очереди ядро пытается послать токен OUT в текущем фрейме
e) Пишем второй (последний) пакет канала 1
f) После успешного завершения последней передачи выдаётся прерывание `XFRC`
g) По прерыванию `XFRC` освобождает канал для других передач
h) Обрабатываем не-ACK ответы

Нормальные групповые/управляющие OUT/SETUP и групповые/управляющие IN



• **Обработчик прерываний групповых/управляющих передач OUT/SETUP и IN**

а) Групповые/Управляющие OUT/SETUP

Наличие свободного места в FIFO и в очереди запросов определяют по прерыванию `NPTXFE` регистра `OTG_FS_GINTSTS`.

```
Unmask (NAK/TXERR/STALL/XFRC)
if (XFRC) {
    Reset Error Count
    Mask ACK
    De-allocate Channel
    }
else if (STALL)    {
    Transfer Done = 1
    Unmask CHH
    Disable Channel
}
    else if (NAK or TXERR ) {
    Rewind Buffer Pointers
```

```
        Unmask CHH
Disable Channel
if (TXERR)   {
            Increment Error Count
            Unmask ACK
    }
        else {
            Reset Error Count
            }
        }
    else if (CHH){
        Mask CHH
        if (Transfer Done or (Error_count == 3)) {
            De-allocate Channel
            }
        else  {
            Re-initialize Channel
            }
        }
    else if (ACK) {
        Reset Error Count
        Mask ACK
        }
```

b) Групповые/Управляющие IN

Запросы выдаются при наличии свободного места в очереди.

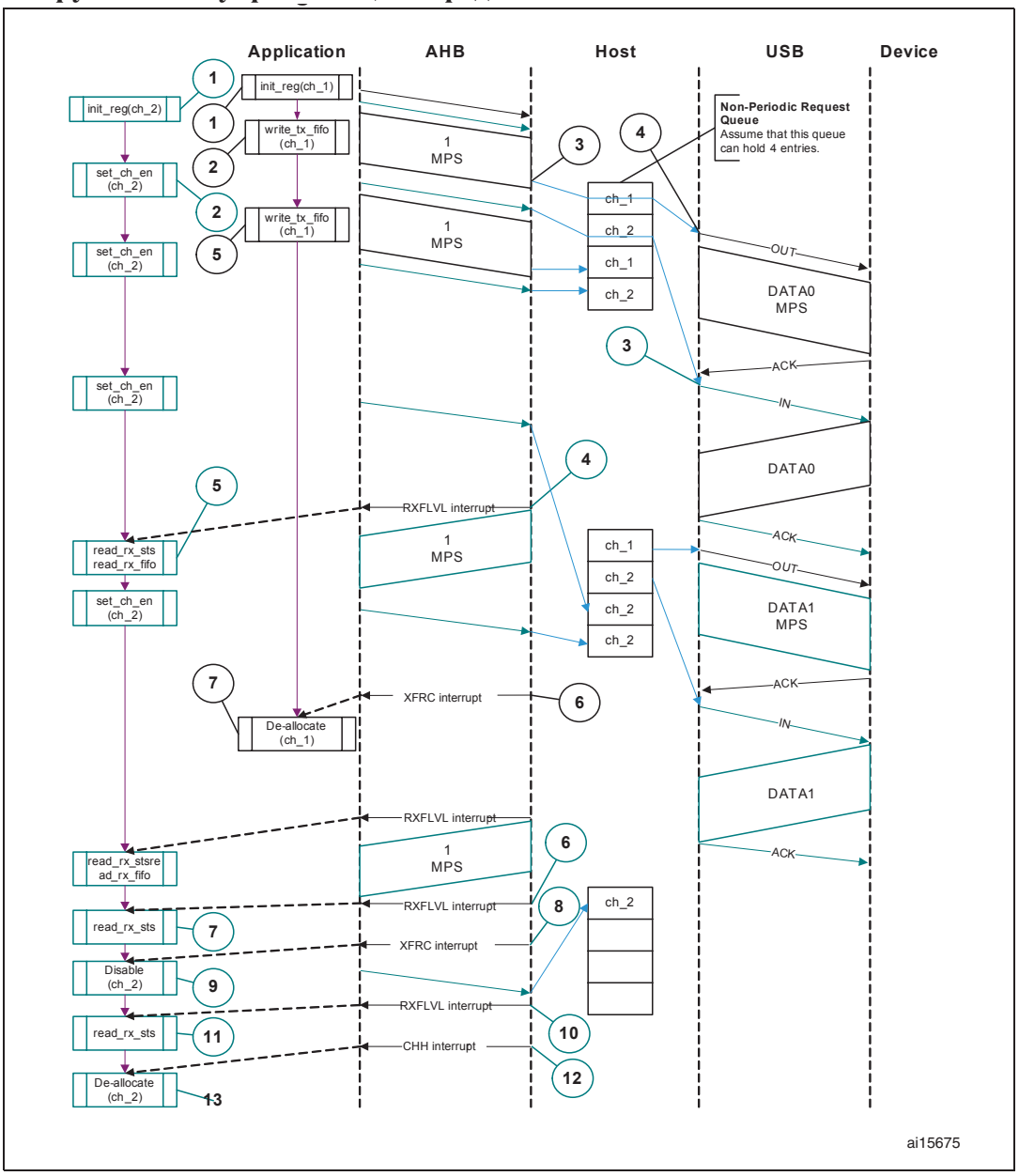
```
    Unmask (TXERR/XFRC/BBERR/STALL/DTERR)
    if (XFRC){
        Reset Error Count
        Unmask CHH
        Disable Channel
        Reset Error Count
        Mask ACK
}
    else if (TXERR or BBERR or STALL) {
        Unmask CHH
        Disable Channel
        if (TXERR) {
            Increment Error Count
            Unmask ACK
            }
        }
    else if (CHH) {
        Mask CHH
        if (Transfer Done or (Error_count == 3)) {
            De-allocate Channel
            }
        else {
            Re-initialize Channel
            }
        }
    else if (ACK) {
        Reset Error Count
        Mask ACK
}
    else if (DTERR) {
        Reset Error Count
        }
```

• Групповые и управляющие передачи IN

На рисунке ниже принимаем групповые или управляющие передачи IN по каналу 2. Полагаем, что:

– Принимаются два максимальных пакета (размер = 1024 байта).
– Приёмный FIFO может хранить хоть один максимальный пакет и два слова состояния на пакет (72 байта для FS).
– Глубина не-периодической очереди = 4.

Групповые и управляющие передачи IN



Последовательность операций такова:

a) Инициализируем канал 2.
b) Ставим бит CHENA в HCCHAR2 для записи запроса IN в не-периодическую очередь.
c) Ядро пробует послать токен IN после завершения текущей передачи OUT.
d) После приёма пакета в FIFO ядро выдаёт прерывание RXFLVL.
e) Приняв прерывание RXFLVL маскируем его, читаем число принятых байтов из состояния принятого пакета и читаем FIFO. Затем демаскируем прерывание RXFLVL.
f) Ядро выдаёт прерывание RXFLVL по каждой строке состояния конца приёма в FIFO.
g) Принятые не-IN пакеты данных (PKTSTS в GRXSTSR ≠ 0b0010) игнорируются.
h) Ядро выдаёт прерывание XFRC по каждому чтению состояния принятого пакета.
i) По прерыванию XFRC выключаем канал и стопорим запись дальнейших запросов в OTG_FS_HCCHAR2. Ядро пишет запрос выключения канала в не-периодическую очередь сразу после записи в регистр OTG_FS_HCCHAR2.
j) Ядро выдаёт прерывание RXFLVL сразу записи состояния останова в FIFO.

k) Читаем и игнорируем состояния приёма пакета.

l) Ядро выдаёт прерывание CHH сразу после извлечения состояния останова из FIFO.

m) По прерыванию CHH отдаём канал для новых операций.

n) Обрабатываем не-ACK ответы

Управляющие передачи

Управляющие передачи Setup, Data и Status передаются раздельно. Передача OUT Setup, Data или Status подобна групповой передаче OUT. Передача IN Data или Status подобна групповой передаче IN. Для всех трёх передач в поле EPTYP регистра OTG_FS_HCCHAR1 пишут "Управляющая". Для передач Setup в поле PID регистра OTG_FS_HCTSIZ1 пишут SETUP.

Прерывающие передачи OUT

Они показаны на рисунке ниже. Полагаем, что:

– Передаётся 1 пакет в каждом фрейме (до 1 максимального размера), начиная с нечётного фрейма (размер = 1024 байта)

– Периодический передающий FIFO может хранить один пакет (1 KB)

– Глубина периодической очереди равна = 4

Последовательность операций такова:

a) Инициализируем и включаем канал 1. Бит ODDFRM в OTG_FS_HCCHAR1 должен стоять.

b) Пишем первый пакет канала 1.

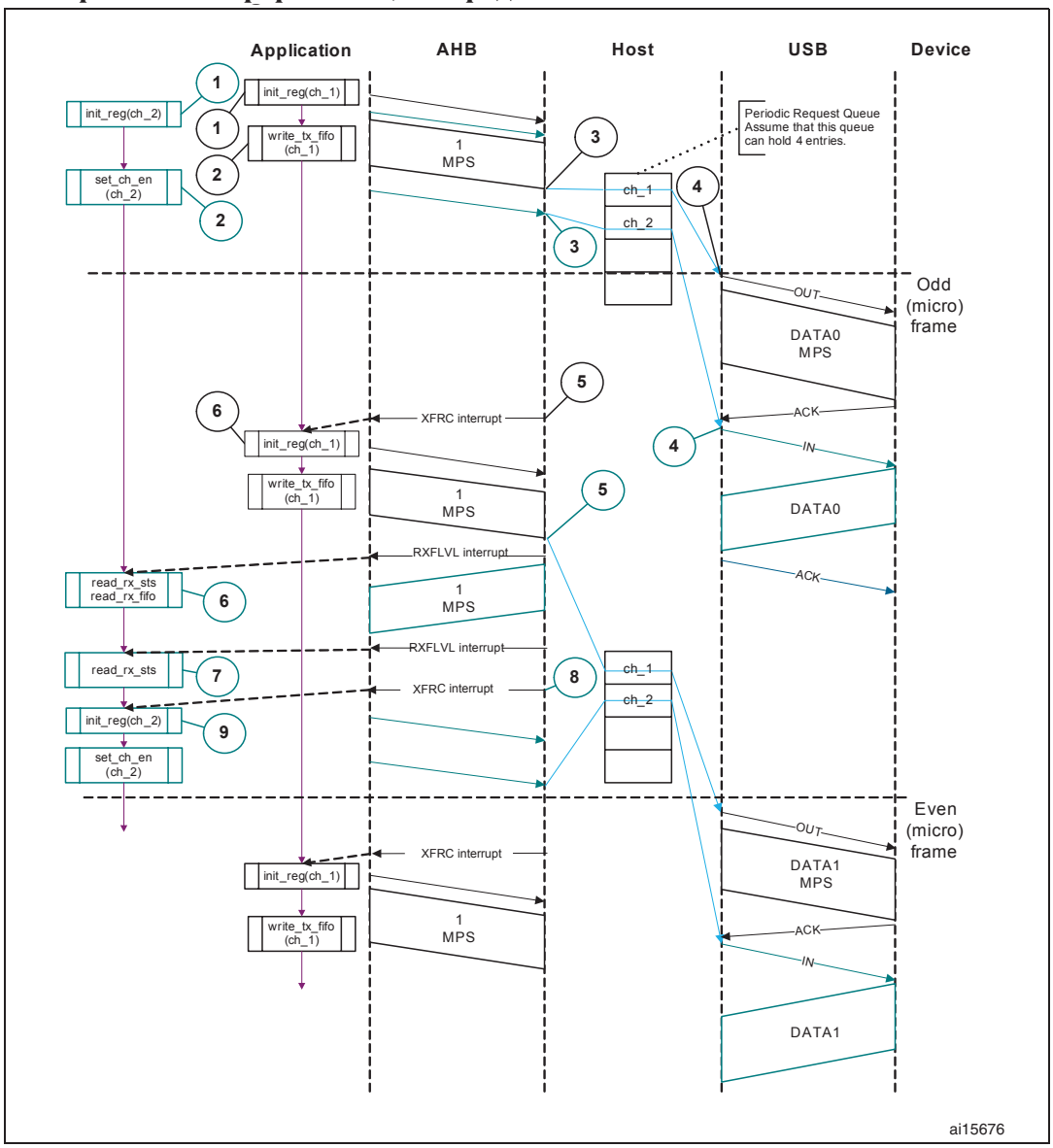
c) Вместе с записью последнего слова каждого пакета хост OTG_FS пишет запрос в периодическую очередь.

d) Хост OTG_FS пытается послать токен OUT в следующем (нечётном) фрейме.

e) Прерывание XFRC выдаётся после успешной передачи последнего пакета.

f) После прерывания XFRC инициализируем канал для следующей передачи.

Нормальные прерывающие передачи OUT/IN



• **Программа обработки прерываний для прерывающих передач OUT/IN**

а) Прерывающие OUT

Место в передающем FIFO определяется по прерыванию `NPTXFE` в `OTG_FS_GINTSTS`.

```
Unmask (NAK/TXERR/STALL/XFRC/FRMOR)
if (XFRC) {
    Reset Error Count
    Mask ACK
    De-allocate Channel
    }
else
    if (STALL or FRMOR) {
        Mask ACK
        Unmask CHH
        Disable Channel
        if (STALL) {
            Transfer Done = 1
             }
        }
    else
        if (NAK or TXERR) {
            Rewind Buffer Pointers
            Reset Error Count
            Mask ACK
    Unmask CHH
            Disable Channel
```

```
        }
    else
        if (CHH) {
        Mask CHH
        if (Transfer Done or (Error_count == 3)) {
                    De-allocate Channel
                    }
        else  {
                Re-initialize Channel (in next b_interval - 1 Frame)
                }
        }
    else
        if (ACK)  {
        Reset Error Count
        Mask ACK
        }
```

b) Прерывающие IN

```
Unmask (NAK/TXERR/XFRC/BBERR/STALL/FRMOR/DTERR)
if (XFRC) {
    Reset Error Count
    Mask ACK
    if (OTG_FS_HCTSIZx.PKTCNT == 0 {
        De-allocate Channel
        }
    else {
        Transfer Done = 1
        Unmask CHH
        Disable Channel
        }
    }
else
    if (STALL or FRMOR or NAK or DTERR or BBERR)  {
        Mask ACK
        Unmask CHH
        Disable Channel
        if (STALL or BBERR)  {
            Reset Error Count
            Transfer Done = 1
            }
        else
            if (!FRMOR) {
            Reset Error Count
            }
        }
else
    if (TXERR)  {
        Increment Error Count
        Unmask ACK
        Unmask CHH
        Disable Channel
        }
    else
        if (CHH)  {
            Mask CHH
            if (Transfer Done or (Error_count == 3))  {
            De-allocate Channel
            }
            else
            Re-initialize Channel (in next b_interval - 1 /Frame)
            }
        }
    else
        if (ACK)  {
            Reset Error Count
            Mask ACK  }
```

- **Прерывающие передачи IN**

Полагаем, что:

- Принимаем один пакет максимальной длины в каждом фрейме, начиная с нечётного (размер = 1024 байта).
- Приёмный FIFO может хранить один максимальный пакет и два слова состояния на один пакет (1031 байт).
- Глубина периодической очереди = 4.

- **Нормальная прерывающая операция IN**

Последовательность операций такова:

a) Инициализируем канал 2. Бит ODDFRM в OTG_FS_HCCHAR2 должен стоять.
b) Ставим бит CHENA в OTG_FS_HCCHAR2 для записи запроса IN в периодическую очередь.
c) Хост пишет запрос IN в очередь по каждой установке бита CHENA в OTG_FS_HCCHAR2.
d) Хост OTG_FS пробует послать токен IN в следующем (нечётном) фрейме.
e) После приёма пакета IN в FIFO, хост OTG_FS выдаёт прерывание RXFLVL.
f) По прерыванию RXFLVL в слове состояния узнаём число принятых байтов и читаем FIFO. Перед чтением FIFO прерывание RXFLVL надо маскировать и открывать его после извлечения всего пакета.
g) После получения в FIFO строки статуса выдаётся прерывание RXFLVL. Принятые не-IN пакеты данных (PKTSTS в GRXSTSR ≠ 0b0010) надо читать и игнорировать.
h) После чтения строки статуса выдаётся прерывание XFRC.
i) По прерыванию XFRC читаем поле PKTCNT регистра OTG_FS_HCTSIZ2. Если бит PKTCNT в OTG_FS_HCTSIZ2 стоит, то выключаем канал, иначе инициализируем канал для следующей передачи. При этом бит ODDFRM в OTG_FS_HCCHAR2 надо сбросить.

- **Изохронные передачи OUT**

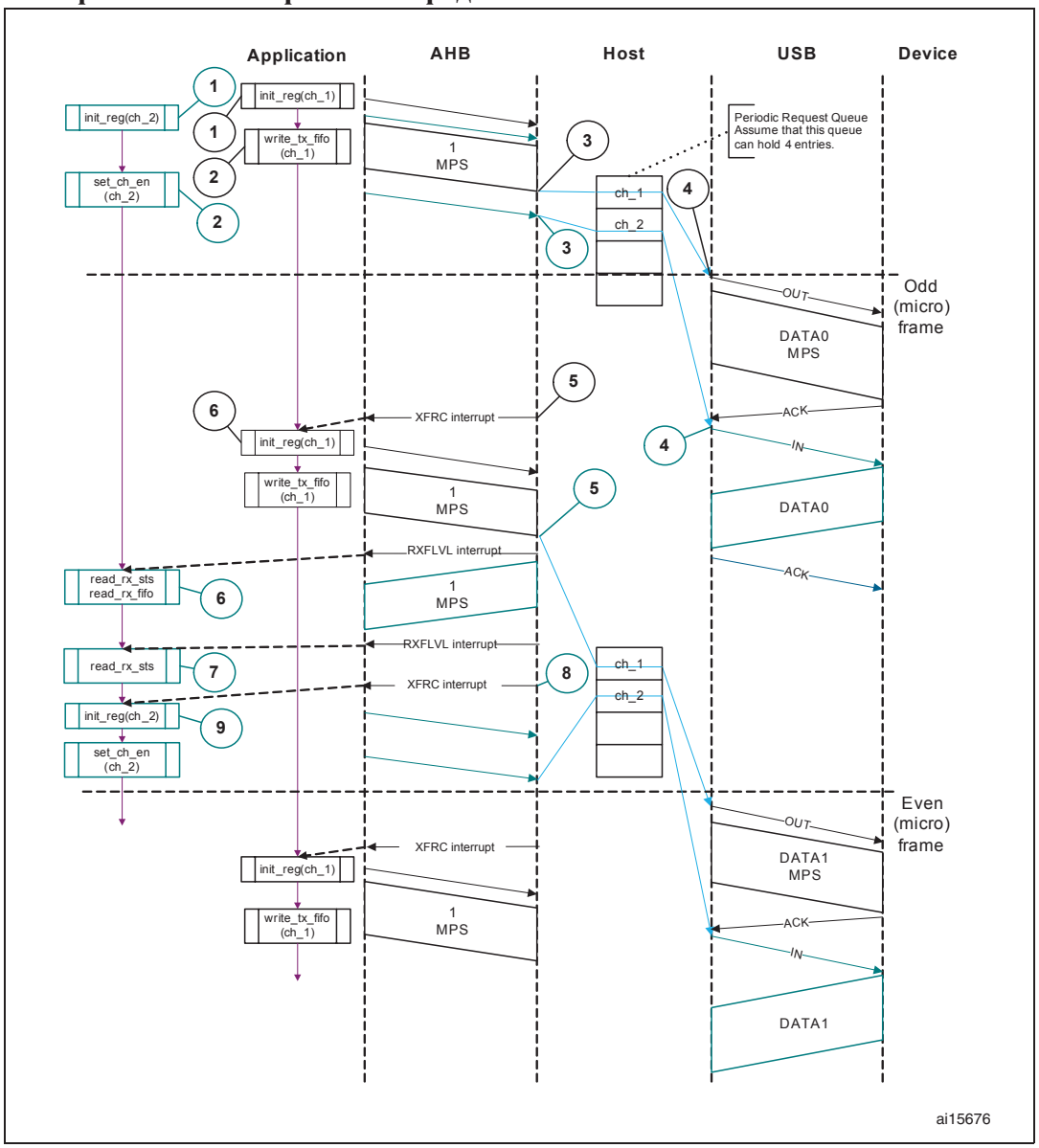
Пример показан на рисунке ниже. Полагаем, что:

- Посылается один пакет максимальной длины в каждом фрейме, начиная с нечётного (размер = 1024 байта).
- Периодический передающий FIFO может содержать один пакет (1 KB).
- Глубина периодической очереди = 4.

Последовательность операций такова:

a) Инициализируем канал 1. Бит ODDFRM в OTG_FS_HCCHAR1 должен стоять.
b) Пишем первый пакет канала 1.
c) Вместе с записью последнего слова пакета хост OTG_FS пишет запрос в периодическую очередь.
d) Хост OTG_FS пробует послать токен OUT в следующем (нечётном) фрейме.
e) Прерывание XFRC выдаётся после успешной передачи последнего пакета.
f) После него инициализируем канал для следующей передачи.
g) Обрабатываем не-ACK ответы.

Нормальные изохронные передачи OUT/IN



- **Программа обработки прерываний для изохронных передач OUT/IN**

Изохронные OUT

```
Unmask (FRMOR/XFRC)
if (XFRC) {
    De-allocate Channel
    }
else
    if (FRMOR) {
        Unmask CHH
        Disable Channel
        }
    else
        if (CHH) {
            Mask CHH
            De-allocate Channel
            }
```

Изохронные IN

```
Unmask (TXERR/XFRC/FRMOR/BBERR)
if (XFRC or FRMOR) {
if (XFRC and (OTG_FS_HCTSIZx.PKTCNT == 0)) {
        Reset Error Count
        De-allocate Channel
        }
    else {
        Unmask CHH
        Disable Channel
```

```
    }
}
else
    if (TXERR or BBERR) {
        Increment Error Count
        Unmask CHH
Disable Channel
}
    else
if (CHH) {
    Mask CHH
    if (Transfer Done or (Error_count == 3)) {
                De-allocate Channel
                }
        else {
         Re-initialize Channel
         }
         }
```

- **Изохронные передачи IN**

Полагаем, что:

- Принимаем один пакет максимальной длины в каждом фрейме, начиная с нечётного (размер = 1024 байта).
- Приёмный FIFO может содержать 1 максимальный пакет и два слова статуса на каждый пакет (1031 байт).
- Глубина периодической очереди = 4.

Последовательность операций такова:

a) Инициализируем канал 2. Бит ODDFRM в OTG_FS_HCCHAR2 должен стоять.
b) Ставим CHENA в OTG_FS_HCCHAR2 для записи запроса IN в периодическую очередь.
c) Запрос IN пишется в очередь по каждой установке бита CHENA в OTG_FS_HCCHAR2.
d) Хост пробует послать токен IN в следующем нечётном фрейме.
e) После приёма пакета IN в FIFO выдаётся прерывание RXFLVL.
f) По нему из состояния принятого пакета берём число полученных байтов и читаем FIFO. Перед чтением FIFO маскируем прерывание RXFLVL и открываем его после извлечения всего пакета.
g) Прерывание RXFLVL выдаётся после приёма строки состояния пакета в FIFO. Строки состояния не-IN пакетов данных (бит PKTSTS и OTG_FS_GRXSTSR ≠ 0b0010) читаются и игнорируются.
h) Прерывание XFRC выдаётся после чтения из FIFO строки состояния пакета.
i) По нему читаем поле PKTCNT регистра OTG_FS_HCTSIZ2. Если PKTCNT≠ 0, то выключаем канал, иначе инициализируем его для следующей передачи. В это время нужно сбросит бит ODDFRM в регистре OTG_FS_HCCHAR2.

- **Выбор глубины очереди**

Глубина периодической и не-периодической очередей должна соответствовать числу одновременно работающих соответствующих конечных точек.

Чем глубже не-периодической очереди (с соответствующим размером FIFO), тем чаще возможны не-периодические передачи.

Глубокая периодическая очередь позволяет планировать периодические передачи. Если глубина меньше запланированного числа периодических передач в микрофрейме, то возникает переполнение фрейма.

- **Обработка ошибки перекрёстных шумов**

Две типа ошибки: пакета и порта.

Ошибка пакета появляется когда устройство посылает данных больше, чем максимальный размер пакета канала. Ошибка порта появляется когда ядро продолжает принимать данные от устройства после EOF2 (конец фрейма 2, очень близкого к SOF).

Когда контроллер OTG_FS обнаруживает ошибку пакета, он перестаёт писать данные в буфер Rx и ждёт конца пакета (EOP). После обнаружения EOP он сливает уже записанные в Rx данные и выдаёт прерывание ошибки перекрёстных шумов.

Когда контроллер OTG_FS обнаруживает ошибку порта, он сливает RxFIFO и выключает порт. Выдаётся прерывание выключения порта (HPRTINT в OTG_FS_GINTSTS, PENCHNG в OTG_FS_HPRT). Программа по биту POCA in OTG_FS_HPRT, определяет не было ли это вызвано перегрузкой порта и выполняет программный сброс порта. После ошибки порта токены больше не посылаются.

34.17.5. Программная модель устройства

Инициализация по сбросу USB

1. Ставим бит NAK для всех конечных точек OUT
   - SNAK = 1 в OTG_FS_DOEPCTLx
2. Демаскируем прерывания:
   - INEP0 = 1 в OTG_FS_DAINTMSK (точки 0 IN)
   - OUTEP0 = 1 в OTG_FS_DAINTMSK (точки 0 OUT)
   - STUP = 1 в DOEPMSK
   - XFRC = 1 в DOEPMSK
   - XFRC = 1 в DIEPMSK
   - TOC = 1 в DIEPMSK
3. Определяем RAM для всех FIFO данных
   - В регистре OTG_FS_GRXFSIZ обеспечиваем возможность принимать управляющие OUT и SETUP. Если контроль порога не включён, то как минимум размер должен быть равен 1 максимальному пакету точки 0 + 2 слова (для состояния управляющего пакета данных OUT) + 10 слов (для пакетов SETUP).
   - В регистре OTG_FS_TX0FSIZ обеспечиваем возможность (в зависимости от номера выбранного FIFO) передавать управляющие данные IN. Как минимум размер должен быть равен 1 максимальному пакету конечной точки 0.
4. Пишем регистры конечной точки 0 OUT для приёма пакетов SETUP
   - STUPCNT = 3 в OTG_FS_DOEPTSIZ0 (для приёма череды 3 пакетов SETUP)

Всё, пакеты SETUP можно принимать.

Инициализация конечной точки по завершению переучёта

1. По прерыванию ENUMDNE в регистре OTG_FS_GINTSTS, из регистра OTG_FS_DSTS выясняем скорость переучёта.
2. В поле MPSIZ регистра OTG_FS_DIEPCTL0 пишем максимальный размер пакета точки 0. Он зависит от скорости переучёта.

Отныне устройство может принимать пакеты SOF и передавать по точке 0.

Инициализация конечной точки по команде SetAddress

После приёма команды **SetAddress** в пакете SETUP:

1. В регистр OTG_FS_DCFG пишем полученный в команде **SetAddress** адрес
2. Вынуждаем ядро послать пакет состояния IN.

Инициализация конечной точки по команде SetConfiguration/SetInterface

После приёма команд **SetConfiguration** или **SetInterface** в пакете SETUP:

1. По команде **SetConfiguration** в регистрах конечных точек пишем новую конфигурацию.
2. По команде **SetInterface** пишем регистры затронутых командой конечных точек.
3. Некоторые точки в новой конфигурации и установках становятся нерабочими. Деактивируем такие точки.
4. В регистре the interrupt for each active endpoint and mask the interrupts for all inactive endpoints in the OTG_FS_DAINTMSK демаскируем прерывания активных точек и маскируем прерывания неактивных конечных точек.
5. Определяем RAM для всех FIFO.

6. Вынуждаем ядро послать пакет состояния IN.

Отныне устройство может принимать и передавать любые пакеты данных.

Активация конечной точки

Новый тип конечной точки задаём так:

1. В регистре OTG_FS_DIEPCTLx (для IN или двунаправленных точек) или регистре OTG_FS_DOEPCTLx (для OUT или двунаправленных точек) пишем поля:
   - Максимальный размер пакета
   - Активная точка USB = 1
   - Начальное значение переключения данных точки (прерывающие и групповые)
   - Тип точки
   - Номер TxFIFO

Теперь точка активирована и ядро начинает рассматривать получаемые точкой токены и отсылать нужные ответы.

Деактивация конечной точки

Делаем так:

1. Снимаем бит активной точки USB в В регистре OTG_FS_DIEPCTLx (для IN или двунаправленных точек) или регистре OTG_FS_DOEPCTLx (для OUT или двунаправленных точек).

Теперь ядро игнорирует направленные этой точке токены, что приводит к таймауту USB.

NB: Программа должна снять биты NPTXFEM и RXFLVLM в регистре OTG_FS_GINTMSK.

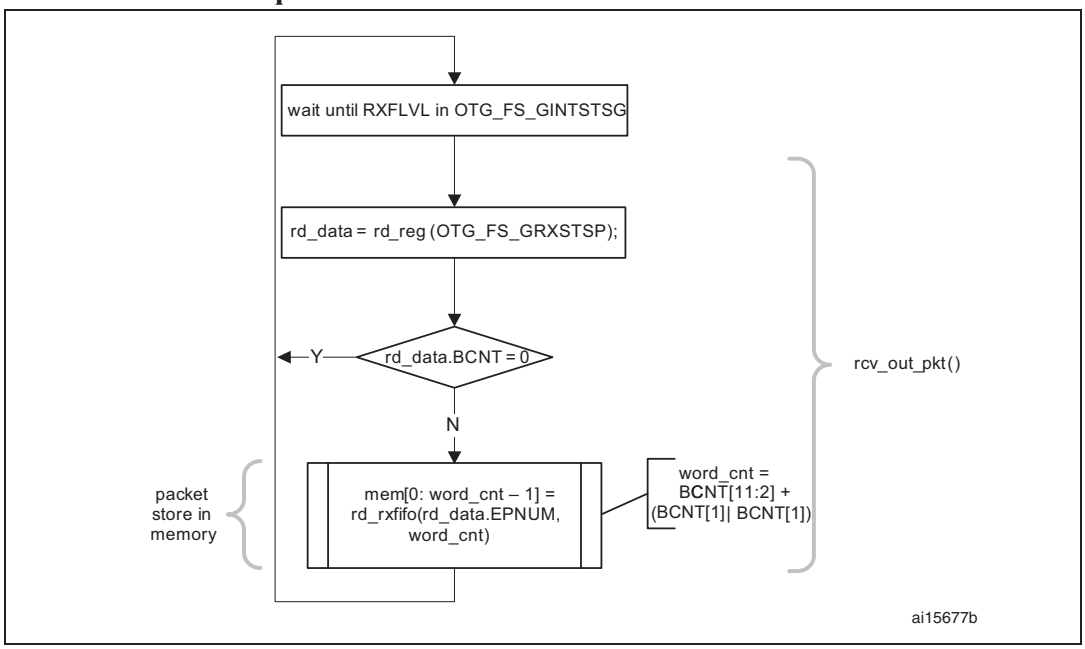
34.17.6. Модель работы

Передачи SETUP и OUT данных

• **Чтение данных пакета**

1. По прерыванию RXFLVL регистра OTG_FS_GINTSTS читаем регистр состояния приёма (OTG_FS_GRXSTSP).
2. Записью RXFLVL = 0 (в OTG_FS_GINTMSK) маскируем прерывание RXFLVL регистра OTG_FS_GINTSTS вплоть до извлечения пакета из FIFO.
3. Если счётчик байтов пакета не равен 0, то извлекаем байты из FIFO в память.
4. Статус принятого в FIFO пакета может показывать:
   a) Действует Глобальный OUT NAK:
   PKTSTS = Глобальный OUT NAK, BCNT = 0x000, EPNUM = Всё Равно (0x0), DPID = Всё Равно (0b00).
   b) Пакет SETUP:
   PKTSTS = SETUP, BCNT = 0x008, EPNUM = Номер управляющей EP, DPID = D0.
   c) Конец фазы SETUP:
   PKTSTS = Конец фазы SETUP, BCNT = 0x0, EPNUM = Номер управляющей EP, DPID = Всё Равно (0b00).
   После чтения этой строки из FIFO выдаётся прерывание SETUP этой конечной точки.
   d) Пакет данных OUT:
   PKTSTS = DataOUT, BCNT = Размер пакета (0 ≤ BCNT ≤ 1 024), EPNUM = Номер точки, DPID = PID данных.
   e) Передача данных завершена:
   PKTSTS = Передача данных завершена, BCNT = 0x0, EPNUM = Номер точки, DPID = Всё Равно (0b00).
   После чтения этой строки из FIFO выдаётся прерывание Конца передачи этой точки.
5. После извлечения данных пакета из FIFO надо демаскировать прерывание RXFLVL в регистре (OTG_FS_GINTSTS).
6. Шаги 1–5 повторяются по каждому прерыванию RXFLVL в OTG_FS_GINTSTS. Чтение пустого приёмного FIFO сводит ядро с ума.

Чтение пакета приёмного FIFO



- **Передачи SETUP**

Программные требования

1. При ненулевом поле STUPCNT в регистре OTG_FS_DOEPTSIZx принимаемые пакеты SETUP точки OUT пишутся в FIFO независимо от состояния NAK и бита EPENA в регистре OTG_FS_DOEPCTLx. Поле STUPCNT декрементируется после каждого приёма пакета SETUP. При неверном значении поля STUPCNT пакеты принимаются и поле декрементируется, но определить число присланных пакетов невозможно.
   - STUPCNT = 3 в OTG_FS_DOEPTSIZx
2. В приёмном FIFO данных управляющей точки всегда надо отводить место для приёма трёх пакетов SETUP.
   - Резервируется 10 слов. Три слова для первого пакета SETUP, 1 слово для статуса Конца фазы SETUP и 6 слов для ещё двух пакетов SETUP всех управляющих точек.
   - В 3 словах пакета SETUP хранятся 8 байтов данных и 4 байта статуса.
   - Это место используется ядром только для данных SETUP.
3. Из FIFO надо читать 2 слова пакета SETUP.
4. Слово статуса Конца фазы SETUP надо выбрасывать.

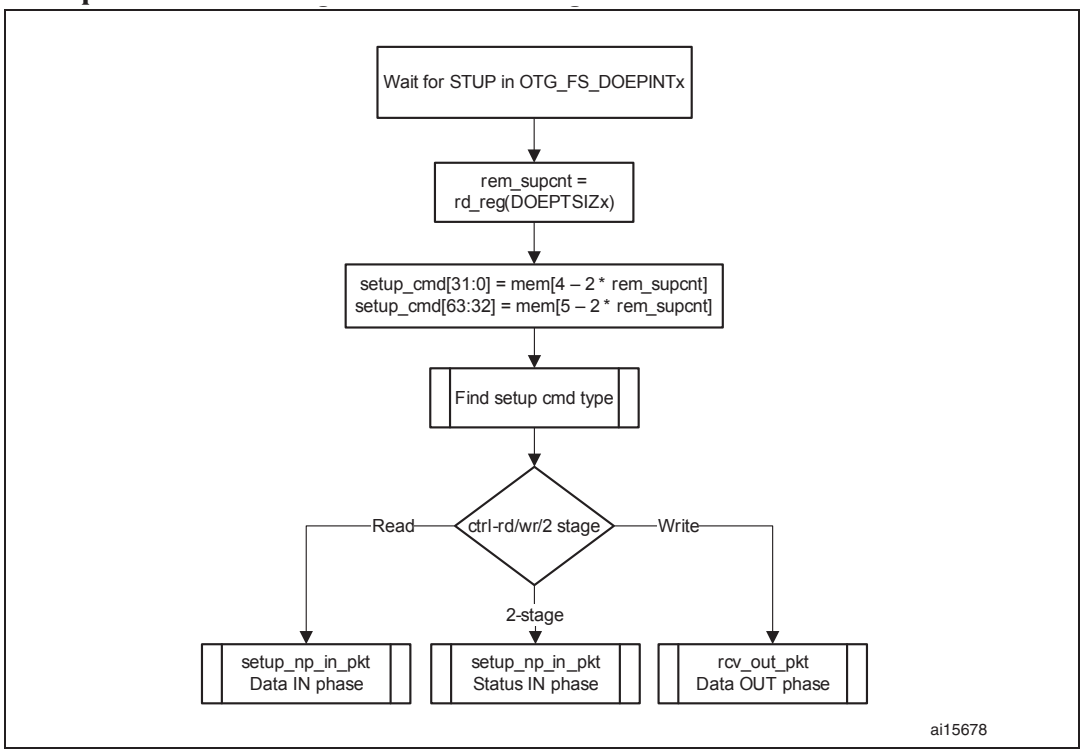
Внутренний ход данных

1. Принимаемые пакеты SETUP пишутся в FIFO без проверки доступного места и независимо от состояния NAK и бита STALL.
   - В точках IN/OUT, принявших пакет SETUP, ядро ставит биты IN NAK и OUT NAK.
2. По каждому пакету SETUP декрементируется поле STUPCNT и в FIFO пишутся 3 слова:
   - В первом слове информация для ядра
   - Во втором слове первые 4 байта команды SETUP
   - В третьем слове последние 4 байта команды SETUP
3. При изменении фазы SETUP на IN/OUT данных ядро пишет в FIFO слово Конца фазы SETUP.
4. На стороне AHB пакеты SETUP опустошаются программой.
5. По чтению из FIFO слова Конца фазы SETUP выдаётся прерывание STUP OTG_FS_DOEPINTx.
   - Ядро снимает бит разрешения управляющей точки OUT.

Программная последовательность

1. В регистр OTG_FS_DOEPTSIZx пишем STUPCNT = 3
2. Ждём прерывания RXFLVL (OTG_FS_GINTSTS) и читаем данные пакета из FIFO.
3. Прерывание STUP (OTG_FS_DOEPINTx) отмечает завершение передачи данных SETUP.
   - По нему из регистра OTG_FS_DOEPTSIZx узнаём число принятых пакетов SETUP и обрабатываем последний.

Обработка пакета SETUP



Обработка череды более 3 пакетов SETUP

Обычно при ошибке пакета SETUP хост не посылает подряд более трёх пакетов SETUP одной точке, но при нормальной работе это не исключено. В этом случае контроллер OTG_FS выдаёт прерывание B2BSTUP в регистре OTG_FS_DOEPINTx.

- **Установка Глобального OUT NAK**

Внутренний ход данных

1. При установке бита SGONAK в регистре OTG_FS_DCTL в FIFO пишутся только пакеты SETUP, не-изохронные пакеты OUT получают ответ NAK, изохронные пакеты OUT игнорируются.
2. В приёмный пишется слово Глобального OUT NAK. Места должно хватать.
3. При его чтении из FIFO выдаётся прерывание GONAKEFF в регистре OTG_FS_GINTSTS.
4. Теперь можно снять бит SGONAK в регистре OTG_FS_DCTL.

Программная последовательность

1. Пишем SGONAK = 1 в регистре OTG_FS_DCTL
2. Ждём прерывания GONAKEFF в регистре OTG_FS_GINTSTS.
3. Между установкой бита SGONAK в OTG_FS_DCTL и прерыванием GONAKEFF (OTG_FS_GINTSTS) нормальные пакеты OUT принимаются.
4. Это прерывание можно временно маскировать записью GINAKEFFM = 0 в регистре OTG_FS_GINTMSK
5. Из режима Глобального OUT NAK выходят снятием бита SGONAK в регистре OTG_FS_DCTL. Заодно очищается прерывание GONAKEFF (OTG_FS_GINTSTS).
6. Если оно было ранее маскировано, то демаскируем его.

- **Выключение точки OUT**

Программная последовательность

1. Сначала включаем Глобальный OUT NAK. SGONAK = 1 в OTG_FS_DCTL
2. Ждём прерывания GONAKEFF (OTG_FS_GINTSTS)
3. Выключаем точку OUT записью:
   - EPDIS = 1 в OTG_FS_DOEPCTLx
   - SNAK = 1 в OTG_FS_DOEPCTLx
4. Ждём прерывания EPDIS (OTG_FS_DOEPINTx), автоматически снимаются биты EPDIS = 0 и EPENA = 0 в регистре OTG_FS_DOEPCTLx

5. Чтобы принимать данные по другим точка OUT выходим из режима Глобального OUT NAK снятием бита SGONAK в регистре OTG_FS_DCTL.

• Общие не-изохронные передачи данных OUT

Программная последовательность

1. Для начала размещаем приёмный буфер в памяти.
2. Размер передачи точки должен быть кратен максимальному размеру пакета точки, выравненному на границу слова.
   - размер передачи[EPNUM] = $n \times$ (MPSIZ[EPNUM] + 4 – (MPSIZ[EPNUM] mod 4))
   - счётчик пакетов[EPNUM] = n
   - $n>0$
3. Размер переданных данных вычисляется по регистру размера передачи.
   - Принято данных = начальный размер – остаток в регистре
   - Число пакетов с принятыми данными = начальное число – остаток в регистре

Внутренний ход данных

1. Сначала в регистрах точки ставим размер передачи и счётчик пакетов, снимаем бит NAK и включаем точку на приём.
2. После снятия бита NAK ядро начинает писать данные в свободное место FIFO. Вместе с данными в FIFO пишется статус пакета и декрементируется счётчик пакетов.
   - Пакеты OUT с ошибкой CRC сливаются из приёмного FIFO автоматически.
   - Повторно посылаемые чередой не-изохронные пакеты OUT той же точке с тем же PID от хоста, который не воспринимает ответ ACK, отбрасываются ядром и счётчик пакетов не декрементируется.
   - При нехватке места в FIFO изохронные и не-изохронные пакеты данных игнорируются, на не-изохронные токены OUT выдаётся ответ NAK.
   - Во всех трёх случаях счётчик пакетов не декрементируется.
3. При обнулении счётчика пакетов или по приёму короткого пакета ставится бит NAK конечной точки и изохронные и не-изохронные пакеты данных игнорируются, на не-изохронные токены OUT выдаётся ответ NAK.
4. Из приёмного FIFO данные извлекается пакетами, по одному пакету за раз.
5. Размер передачи конечной точки уменьшается на размер пакета после его извлечения из FIFO.
6. Строка конца передачи данных OUT пишется в приёмный FIFO после:
   - Обнулении размера передачи и счётчика пакетов
   - Последний принятый в FIFO пакет OUT был коротким (0 ≤ размер < максимума)
7. При извлечении этой строки из FIFO выдаётся прерывание и точка выключается.

Программная последовательность

1. В регистр OTG_FS_DOEPTSIZx пишем размер передачи и счётчик пакетов.
2. В регистре OTG_FS_DOEPCTLx пишем характеристки точки и ставим биты EPENA и CNAK.
3. Ждём прерывания RXFLVL (в OTG_FS_GINTSTS) и извлекаем данные из FIFO.
   - Этот шаг может повторяться несколько раз в зависимости от размера передачи.
4. Прерывание XFRC (OTG_FS_DOEPINTx) отмечает конец не-изохронной передачи данных OUT.
5. По регистру OTG_FS_DOEPTSIZx определяем длину принятых данных.

• Общие изохронные передачи данных OUT

Программные требования

1. Требования такие же, как и для не-изохронных передач OUT.
2. Размер передачи и счётчик пакетов всегда должны быть равны ожидаемому числу пакетов максимальной длины в одном фрейме и не больше. Изохронные передачи OUT не могут занимать более 1 фрейма.
3. Данные и статус изохронных передач OUT надо прочесть из FIFO до конца периодического фрейма (прерывание EOPF в регистре OTG_FS_GINTSTS).
4. Для приёма следующего фрейма изохронную точку OUT надо включить после EOPF (OTG_FS_GINTSTS) и до SOF (OTG_FS_GINTSTS).

Внутренний ход данных

1. Он немного отличается от хода данных не-изохронных точек OUT.
2. При включении изохронной точки OUT установкой бита EPENA и очисткой бита NAK, бит Чётного/Нечётного фрейма должен быть установлен правильно. Ядро принимает изохронные данные отдельным фреймом только при:
   - EONUM (в OTG_FS_DOEPCTLx) = SOFFN[0] (в OTG_FS_DSTS)
3. После чтения данных и статуса пакета из FIFO в поле RXDPID регистра OTG_FS_DOEPTSIZx пишется PID данных последнего принятого пакета.

Программная последовательность

1. В регистр OTG_FS_DOEPTSIZx пишем размер передачи и счётчик пакетов.
2. В регистре OTG_FS_DOEPTSIZx ставим характеристики точки и пишем EPENA = 1, CNAK=1 и EONUM = (0: Чётный/1: Нечётный)
3. Ждём прерывания RXFLVL (в OTG_FS_GINTSTS) и извлекаем данные пакета из FIFO
   - Этот шаг может повторяться несколько раз в зависимости от размера передачи.
4. Прерывание XFRC (OTG_FS_DOEPINTx) отмечает конец изохронной передачи данных OUT. При этом данные не обязательно будут достоверными.
5. Для изохронных передач OUT это прерывание выдаётся не всегда, лучше использовать прерывание IISOOXFRM в регистре OTG_FS_GINTSTS.
6. По регистру OTG_FS_DOEPTSIZx определяем длину принятых данных. Данные достоверны когда:
   - RXDPID = D0 (в OTG_FS_DOEPTSIZx) и число пакетов с данными = 1
   - RXDPID = D1 (в OTG_FS_DOEPTSIZx) и число пакетов с данными = 2
   - RXDPID = D2 (в OTG_FS_DOEPTSIZx) и число пакетов с данными = 3

   Число пакетов с данными = Начальное число пакетов – Конечное число пакетов.

   Недостоверные пакеты можно выбрасывать.

• Незавершённые изохронные передачи данных OUT

Внутренний ход данных

1. У изохронных точек OUT прерывание XFRC (в OTG_FS_DOEPINTx) не выдаётся когда:
   - Приёмный FIFO не вмещает весь пакет данных ISO OUT и он отбрасывается,
   - Возникла ошибка CRC принятого пакета,
   - Принятый токен OUT нарушен,
   - Программа не успевает читать данные из FIFO.
2. При появлении конца периодического фрейма до завершения передач всех изохронных точек OUT ядро выдаёт прерывание IISOOXFRM в регистре OTG_FS_GINTSTS, говоря, что не для всех точек выдавалось прерывание XFRC (в OTG_FS_DOEPINTx). Точка с незавершённой передачей остаётся включённой, но передачи по ней нет.

Программная последовательность

1. Прерывание IISOOXFRM (OTG_FS_GINTSTS) означает, что есть хоть одна изохронная точка OUT с незавершенной передачей.
2. Если оно возникло из-за не прочитанного FIFO, то из FIFO надо извлечь все данные и статус. При этом может выдаться прерывание XFRC (OTG_FS_DOEPINTx). В этом случае надо снова включить точку для приёма изохронных данных OUT в следующем фрейме.
3. По прерыванию IISOOXFRM (OTG_FS_GINTSTS) в регистрах OTG_FS_DOEPCTLx надо определить конечные точки с незавершёнными передачами. При этом:
   - Бит EONUM (в OTG_FS_DOEPCTLx) = SOFFN[0] (в OTG_FS_DSTS)
   - EPENA = 1 (в OTG_FS_DOEPCTLx)
4. Предыдущий шаг надо выполнить до прерывания SOF (в OTG_FS_GINTSTS), чтобы номер текущего фрейма не изменился.
5. У изохронных точек OUT с незавершёнными передачами надо отбросить данные и выключить точку установкой бита EPDIS в регистре OTG_FS_DOEPCTLx.
6. Ждём прерывания EPDIS (в OTG_FS_DOEPINTx) и включаем точку для следующего фрейма.

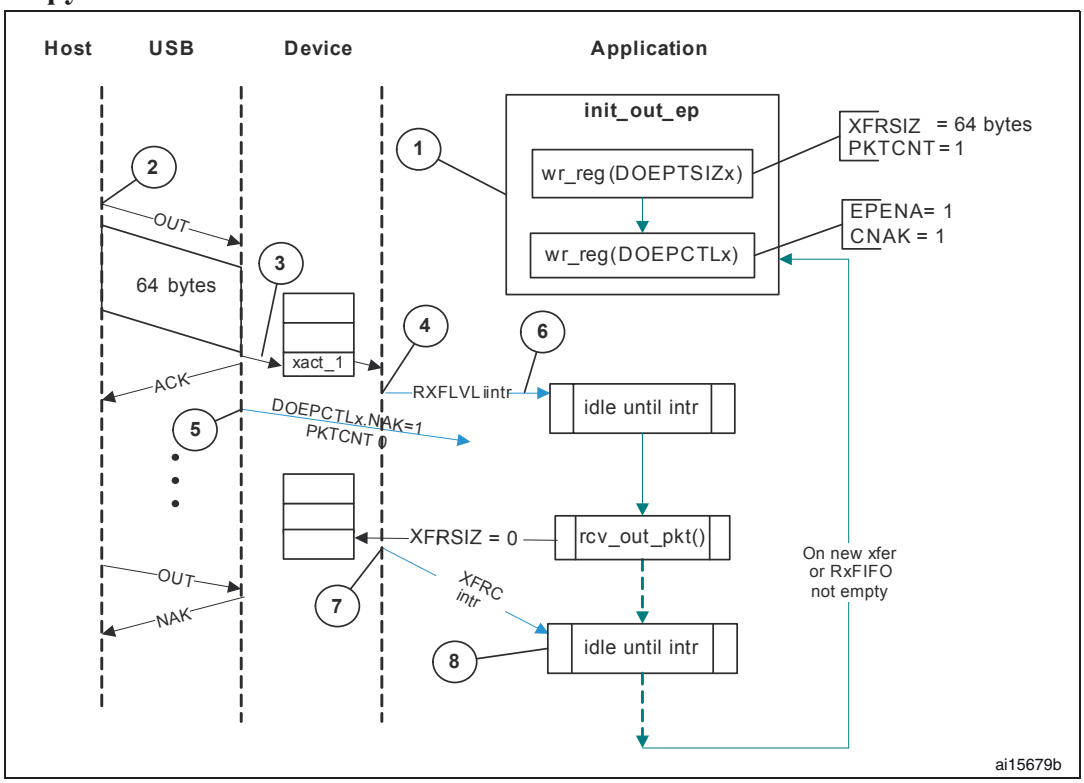
- Из-за задержки выключения точки следующий фрейм может быть не получен.

• Остановка не-изохронной точки OUT

1. Ставим ядро в режим Глобального OUT NAK.
2. Выключаем точку
   - Вместо установки бита SNAK в OTG_FS_DOEPCTL, ставим STALL = 1 (OTG_FS_DOEPCTL). Бит STALL старше бита a NAK.
3. При готовности отмены ответов STALL снимаем бит STALL в OTG_FS_DOEPCTLx.
4. При изменении бита STALL по командам **SetFeature.Endpoint Halt** или **ClearFeature.Endpoint Halt** сделать это надо до фазы передачи Статуса управляющей конечной точки.

Примеры

• Групповая точка OUT



После команд **SetConfiguration/SetInterface** надо инициировать все конечные точки OUT установкой CNAK = 1 и EPENA = 1 (в OTG_FS_DOEPCTLx) и записью подходящих XFRSIZ и PKTCNT в регистрах OTG_FS_DOEPTSIZx.

1. Хост пробует передавать токен OUT по конечной точке.
2. По нему ядро пишет пакет в доступное место RxFIFO.
3. По завершению записи ядро выдаёт прерывание RXFLVL (в OTG_FS_GINTSTS).
4. После приёма PKTCNT пакетов ядро ставит точке бит NAK для задержки приёма.
5. Программа обрабатывает прерывание и читает данные из RxFIFO.
6. По прочтению всех данных (равного XFRSIZ), ядро выдаёт прерывание XFRC в регистре OTG_FS_DOEPINTx.
7. Программа обрабатывает конец передачи.

Передачи данных IN

• Запись пакета

1. Работать можно по прерываниям или опросом.
   - При опросе программа по регистру OTG_FS_DTXFSTSx следит за наличием свободного места в передающем FIFO данных.
   - В режиме прерываний программа ждёт прерывания TXFE (в OTG_FS_DIEPINTx) и по регистру OTG_FS_DTXFSTSx определяет наличие места в FIFO данных для пакета ненулевой длины.
   - Для пакета нулевой длины места в FIFO не требуется.

2. Перед записью в FIFO данных надо записать регистр управления точки (OTG_FS_DIEPCTLx) командой чтение-модификация-запись во избежание изменения регистра (исключая установку бита включения точки).

При наличии места в передающем FIFO можно писать несколько пакетов для одной точки. Для периодических точек IN надо писать по одному пакету на микрофрейм. Новый пакет можно писать только после подтверждения завершения предыдущей передачи.

- **Установка NAK точки IN**

Внутренний ход данных

1. При установке IN NAK точки ядро прекращает передачи независимо от наличия места в наличии данных в передающем FIFO точки.
2. Не-изохронные токены IN получают ответ NAK
   - Изохронные токены IN получают ответ с нулевой длиной данных
3. В ответ на бит SNAK в OTG_FS_DIEPCTLx ядро выдаёт прерывание INEPNE (действует NAK точки IN) в OTG_FS_DIEPINTx.
4. Снимается режим IN NAK установкой бита CNAK в регистре OTG_FS_DIEPCTLx.

Программная последовательность

1. Для остановки передач по точке IN программа пишет SNAK = 1 в OTG_FS_DIEPCTLx
2. Ждём подтверждения остановки по прерыванию INEPNE в OTG_FS_DIEPINTx.
3. В промежутке между установкой бита NAK и прерыванием INEPNE ядро может передавать данные IN.
4. Программа маскировать это прерывание записью INEPNEM = 0 в регистр DIEPMSK
5. Из режима NAK точки выходят снятием бита NAKSTS в OTG_FS_DIEPCTLx (запись CNAK = 1 in OTG_FS_DIEPCTLx). Этим также чистится прерывание INEPNE (в OTG_FS_DIEPINTx).
6. Ранее маскированное прерывание демаскируют записью INEPNEM = 1 in DIEPMSK.

- **Выключение точки IN**

Программная последовательность

1. Программа перестаёт писать данные для выключаемой точки IN.
2. Ставим режим NAK записью SNAK = 1 в OTG_FS_DIEPCTLx.
3. Ждём прерывания INEPNE в OTG_FS_DIEPINTx.
4. В регистр OTG_FS_DIEPCTLx пишем EPDIS = 1 и SNAK = 1.
5. Прерывание EPDISD в OTG_FS_DIEPINTx подтверждает выключение точки. Вместе с этим ядро снимает биты EPENA и EPDIS в регистре OTG_FS_DIEPCTLx.
6. У периодических IN EP число переданных данных узнают по регистру OTG_FS_DIEPTSIZx.
7. Программа должна слить данные из передающего FIFO точки записью номера точки в TXFNUM, и TXFFLSH = 1 регистра OTG_FS_GRSTCTL

Конец слива определяют по снятию ядром бита TXFFLSH опросом регистра OTG_FS_GRSTCTL. Новые данные можно передавать после включения точки.

- **Общие не-периодические передачи данных IN**

Программные требования

1. Сначала надо убедиться, что все передаваемые данные лежат в одном буфере.
2. Для передач IN размер передачи включает несколько пакетов максимальной длины и один короткий пакет, передаваемый последним.
   - Для передачи нескольких пакетов:
     Размер передачи[EPNUM] = $x \times$ MPSIZ[EPNUM] + sp
     Если (sp > 0), то Счётчик пакетов[EPNUM] = x + 1. Иначе, Счётчик пакетов[EPNUM] = x
   - Для передачи пакета без данных:
     Размер передачи[EPNUM] = 0
     Счётчик пакетов[EPNUM] = 1
   - Для передачи нескольких максимальных пакетов и пакета без данных передача разбивается на две части: пакеты с данными и отдельно пакет без данных.

Первая передача: Размер передачи[EPNUM] = x × MPSIZ[EPNUM]; Счётчик пакетов = n;
Вторая передача: Размер передачи[EPNUM] = 0; Счётчик пакетов = 1;

3. После включения точки на передачу ядро изменяет регистр размера передачи. В конце передачи IN по нему определяют сколько данных было передано из FIFO.
4. В FIFO извлечено = Начальный размер передачи – Остаток размера передачи
   - Передано по USB = (Начальный счётчик пакетов – Остаток счётчика пакетов) × MPSIZ[EPNUM]
   - Ещё передавать по USB = (Начальный размер передачи – Передано по USB)

Внутренний ход данных

1. В регистрах точки надо записать размер передачи и счётчик пакетов и включить точку на передачу.
2. Конечно же, данные надо записать в FIFO точки.
3. После записи пакета в FIFO размер передачи уменьшается на размер пакета. Данные пишутся в FIFO до обнуления размера передачи. После записи данных в FIFO инкрементируется "счётчик пакетов в FIFO" (Это 3-бит счётчик, управляемый ядром для каждого FIFO). Для пакетов без данных в FIFO ставится особый флаг, данные не пишутся.
4. Данные из FIFO передаются ядром по токену IN. По каждому ответу ACK не-изохронного пакета IN декрементируется счётчик пакетов вплоть до его обнуления. При таймауте счётчик пакетов не декрементируется.
5. Пакет нулевой длины высылается по токену IN и счётчик пакетов декрементируется.
6. По приёму токена IN при пустом FIFO и нулевом счётчике пакетов ядро выдаёт прерывание ITTXFE, извещая, что бит не стоит NAK. Не-изохронные точки отсылают ответ NAK.
7. Ядро само отматывает указатели FIFO и прерывание таймаута не выдаётся.
8. При нулевых размере передачи и счётчике пакетов выдаётся прерывание конца передачи (XFRC) и точка выключается.

Программная последовательность

1. В регистр OTG_FS_DIEPTSIZx пишем размер передачи и счётчик пакетов.
2. В регистр OTG_FS_DIEPCTLx пишем характеристики точки и ставим биты CNAK и EPENA.
3. При передаче пакета без данных надо опрашивать регистр OTG_FS_DTXFSTSx для определения наличия места в FIFO, или использовать TXFE (в OTG_FS_DIEPINTx).

• Общие периодические передачи данных IN

Программные требования

1. Они совпадают с пунктами 1, 2, 3 и 4 *Общих не-периодических передач данных IN* с некоторым изменением пункта 2.
   - Передавать можно только несколько максимальных пакетов данных с возможным коротким пакетом с данными в конце передачи. При этом:
     Размер передачи[EPNUM] = x × MPSIZ[EPNUM] + sp (где $x \geq 0$, и $0 \leq sp < MPSIZ[EPNUM]$)
     Если ($sp > 0$), то Счётчик пакетов[EPNUM] = $x + 1$ Иначе, Счётчик пакетов[EPNUM] = x;
     MCNT[EPNUM] = Счётчик пакетов[EPNUM]
   - Пакеты без данных передаются отдельно. При этом:
   - Размер передачи[EPNUM] = 0
     Счётчик пакетов[EPNUM] = 1; MCNT[EPNUM] = Счётчик пакетов[EPNUM]
2. Программа может планировать передачу только по одному фрейму за раз.
   - (MCNT – 1) × MPSIZ ≤ XFERSIZ ≤ MCNT × MPSIZ
   - PKTCNT = MCNT (in OTG_FS_DIEPTSIZx)
   - Если XFERSIZ < MCNT × MPSIZ, то последний пакет короткий.
   - **NB**: MCNT, PKTCNT и XFERSIZ лежат в OTG_FS_DIEPTSIZx, MPSIZ лежит в OTG_FS_DIEPCTLx.
3. В FIFO должны быть записаны все передаваемые данные до приёма токена IN. При нехватке в FIFO хоть одного слова он будет считаться пустым. Тогда:
   - Изохронная точка IN передаст пакет без данных

- Прерывающая точка IN передаст ответ NAK.

Внутренний ход данных

1. В регистрах точки пишем размер передачи и счётчик пакетов и включаем точку на передачу.
2. Также пишем в FIFO точки передаваемые данные.
3. При записи очередного пакета в FIFO размер передачи уменьшается на размер пакета. Донные пишутся до обнуления размера передачи.
4. По приёму пакета IN периодическая точка передаёт данные из FIFO. Если там нет полного пакета, то ядро выдаёт прерывание пустого TxFIFO точки.
   - Изохронная точка IN передаст пакет без данных
   - Прерывающая точка IN передаст ответ NAK.
5. Счётчик пакетов точки декрементируется когда:
   - Изохронная точка IN передаст пакет без данных
   - Прерывающая точка IN передаст ответ NAK.
   - Обнулятся размер передачи и счётчик пакетов, будет выдано прерывание завершения передачи и точка выключится.
6. Если по "Интервалу периодического фрейма" (`PFIVL` в `OTG_FS_DCFG`) ядро найдёт непустые FIFO изохронных точек IN для текущего фрейма, то оно выдаст прерывание `IISOIXFR` в `OTG_FS_GINTSTS`.

Программная последовательность

1. В регистр `OTG_FS_DIEPCTLx` пишем характеристики точки и ставим биты `CNAK` `EPENA`.
2. Пишем передаваемые данные следующего фрейма в FIFO.
3. Прерывание `ITTXFE` (в `OTG_FS_DIEPINTx`) покажет, что в FIFO записаны ещё не все данные.
4. Если прерывающая точка включена, то игнорируем прерывание, иначе включаем её, чтобы она передала данные по следующему токену IN.
5. Прерывание `XFRC` (в `OTG_FS_DIEPINTx`) без прерывания `ITTXFE` покажет успешное завершение изохронной передачи IN. При этом размер передачи и счётчик пакетов в регистре `OTG_FS_DIEPTSIZx` должны быть нулевыми.
6. Прерывание `XFRC` (в `OTG_FS_DIEPINTx`) независимо от наличия прерывания `ITTXFE` interrupt (in `OTG_FS_DIEPINTx`) покажет успешное завершение прерывающей передачи IN. При этом размер передачи и счётчик пакетов в регистре `OTG_FS_DIEPTSIZx` должны быть нулевыми.
7. Прерывание `IISOIXFR` в `OTG_FS_GINTSTS` без упомянутых прерываний покажет, что ядро не приняло хоть один периодический токен IN в текущем фрейме.

- **Незавершённые изохронные передачи данных IN**

Внутренний ход данных

1. Изохронная передача IN считается незавершённой когда:
   a) Ядро принимает нарушенный токен IN по какой-либо изохронной точке. В этом случае выдаётся прерывание `IISOIXFR` в `OTG_FS_GINTSTS`.
   b) Программа не успевает записать в FIFO все передаваемые данные до получения токена IN. В этом случае выдаётся прерывание пустого TxFIFO в `OTG_FS_DIEPINTx`. Его можно игнорировать, поскольку оно приведёт к прерыванию `IISOIXFR` в `OTG_FS_GINTSTS` в конце периодического фрейма. В ответ на токен IN ядро пошлёт пакет без данных.
2. Запись данных в передающий FIFO надо немедленно остановить.
3. Ставим бит `NAK` и бит выключения точки.
4. Ядро выключает точку, снимает бит выключения и выдаёт прерывание Выключения точки.

Программная последовательность

1. Прерывание пустого TxFIFO изохронной точки можно игнорировать, поскольку оно приведёт к прерыванию `IISOIXFR` в `OTG_FS_GINTSTS`.
2. Оно извещает о незавершённой передаче на какой-либо изохронной точке IN.
3. Точка с незавершённой передачей определяется чтением Управляющих регистров всех изохронных точек IN.

4. Останавливаем запись в периодические передающие FIFO этих точек.
5. Выключаем точку записью SNAK = 1 и EPDIS = 1 в OTG_FS_DIEPCTLx
6. Выключение точки подтверждается соответствующим прерыванием в регистре OTG_FS_DIEPINTx.
   - В этот момент надо слить данные из FIFO с помощью регистра OTG_FS_GRSTCTL или переписать их включением точки для новой передачи в следующем микрофрейме.

- **Остановка не-изохронной точки IN**

Программная последовательность

1. Выключаем точку IN и затем ставим бит STALL в регистре OTG_FS_DIEPCTLx
   - Бит STALL всегда старше бита NAK
2. Прерывание Выключения точки (в OTG_FS_DIEPCTLx) извещает об этом.
3. Сливаем ассоциированный FIFO. В случае не-периодической точки надо повторно включить остальные не-периодические точки.
4. При готовности отмены ответов STALL для точки снимаем бит STALL в OTG_FS_DIEPCTLx.
5. При изменении бита STALL по командам **SetFeature.Endpoint Halt** или **ClearFeature.Endpoint Halt** изменяем его до фазы Статуса управляющей конечной точки.

Особый случай: остановка управляющей точки OUT

Если хост посылает токенов IN/OUT больше, чем предусмотрено а пакете SETUP, то их надо остановить. В этом случае, в фазе данных, после передачи предусмотренных пакетом SETUP данных надо разрешить прерывания ITTXFE в OTG_FS_DIEPINTx и OTEPDIS в OTG_FS_DOEPINTx. Затем при обработке прерываний ставим STALL точки и чистим прерывание.

34.17.7. Худшее время ответа

Это ответ на токен, следующий за изохронным OUT при работе OTG_FS устройством. Время ответа зависит от частоты тактов AHB (7 тактов PHY при равных частотах тактов PHY и AHB, и меньше при более высокой частоте AHB).

Регистры ядра лежат в домене AHB и оно не принимает новых токенов до обновления регистров, а изохронные передачи OUT не требуют ответа, и следующий групповой/прерывающий токен может получить ответ NAK. Изохронные токены отбрасываются и выдаётся прерывание IISOIXFR.

Хост воспринимает отсутствие ответа на токен SETUP как таймаут и повторяет его передачу.

Выбор значения TRDT в OTG_FS_GUSBCFG

Поле TRDT (OTG_FS_GUSBCFG) это число тактов PHY, отведённое MAC для получения статуса FIFO и чтения первого данного из блока PFC после получения токена IN. Это включает задержку синхронизации тактов PHY и AHB (в худшем случае 5 тактов).

Информация о получении MAC токена IN синхронизируется с частотой PFC (такты AHB). Затем PFC читает данные из SPRAM и пишет их в буфер с двойным тактированием (глубиной 4), откуда их читает MAC.

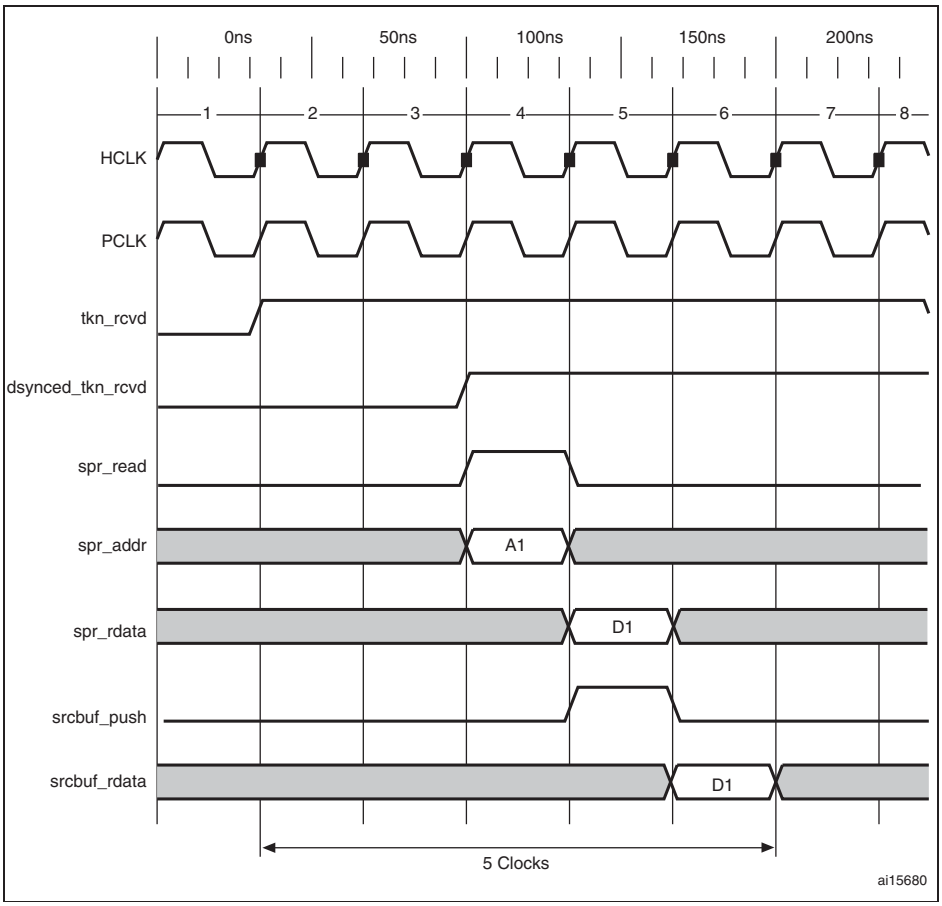
Если частота AHB выше частоты PHY, то можно использовать меньшие значения TRDT (в OTG_FS_GUSBCFG).

На рисунке ниже есть сигналы:

- tkn_rcvd: Сигнал PFC от MAC о получении токена
- dynced_tkn_rcvd: Двухчастотный tkn_rcvd, от PCLK в домен HCLK
- spr_read: Чтение в SPRAM
- spr_addr: Адрес в SPRAM
- spr_rdata: Чтение данных из SPRAM
- srcbuf_push: Запись в буфер
- srcbuf_rdata: Чтение данных из буфера. Данные видны для MAC

См. *Таблицу 203: Значения TRDT*.

Максимальные значения TRDT

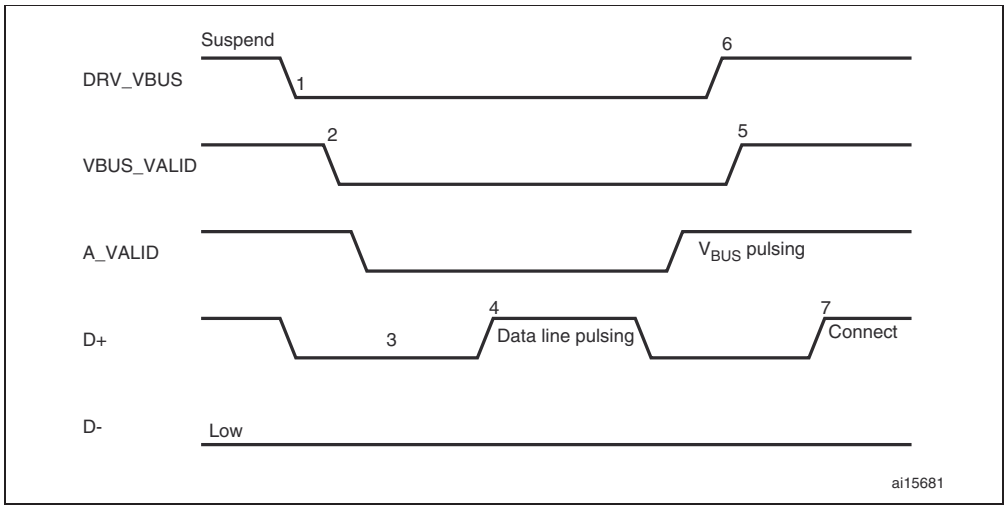


34.17.8. Программная модель OTG

Контроллер OTG_FS поддерживает протоколы HNP и SRP. Если ядро подключено к концу “A”, то именуется A-устройством, если к концу “B” то B-устройством. В режиме хоста контролер OTG_FS выключает V_{BUS}. Протокол SRP позволяет B-устройству попросить A-устройство включить V_{BUS}. Устройство выдаёт импульсы по линии данных и V_{BUS}, но при SRP хост обнаруживает импульс только на линии данных или V_{BUS}. Протокол HNP позволяет B-устройству переговорить и стать хостом. По переговорам после HNP B-устройство приостанавливает шину и оборачивается в периферию.

Протокол запроса сессии A-устройства

Программа ставит бит разрешения SRP в регистре Конфигурации Ядра и контроллер становится A-устройством.



DRV_VBUS = Сигнал PHY подачи V_{BUS}

VBUS_VALID = Сигнал рабочего V_{BUS} от PHY

A_VALID = Сигнал рабочего уровня V_{BUS} от A-периферии к PHY

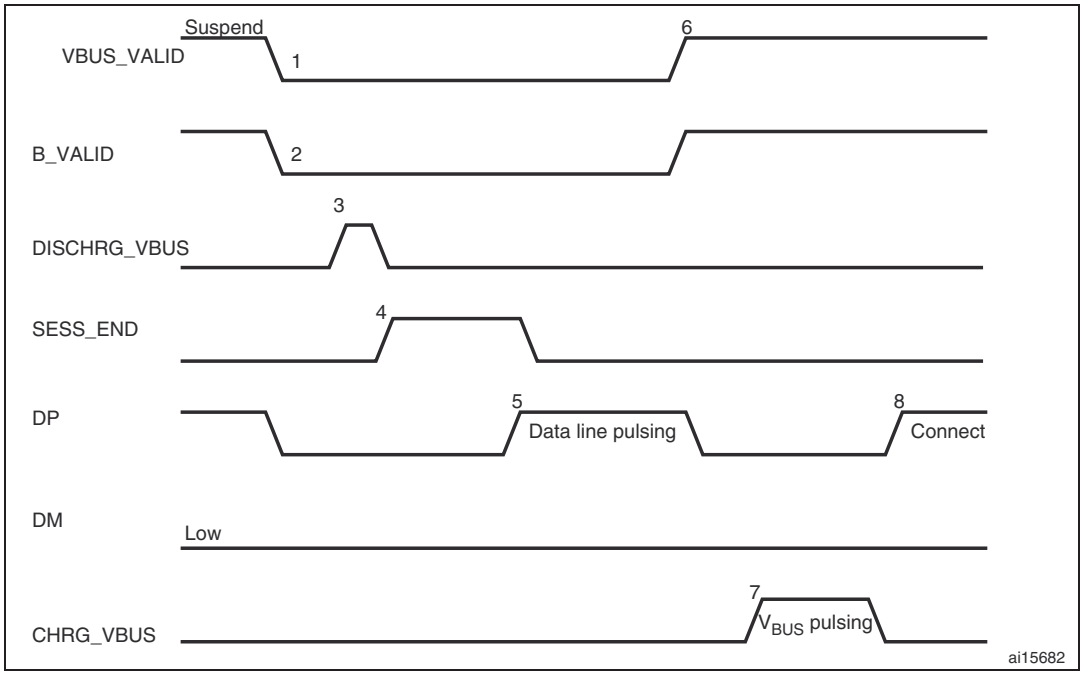
D+ = Линия данных плюс

D- = Линия данных минус

1. Программа останавливает порт и выключает питание простаивающей шины установкой битов остановки порта и выключения питания в регистре управления хоста.
2. PHY показывает выключение питания снятием сигнала VBUS_VALID.
3. Устройство должно обнаружить SE0 не менее чем за 2 ms до пуска SRP при выключенном V_{BUS}.
4. Для запуска SRP устройство подключает свой резистор подпорки линии данных на время от 5 до 10 ms. Контроллер OTG_FS обнаруживает этот импульс.
5. Устройство поднимает V_{BUS} выше порога (2.0 V минимум). Контроллер OTG_FS ставит бит запроса сессии (SRQINT в OTG_FS_GINTSTS) и выдаёт прерывание.
6. По прерыванию программа включает питание установкой бита в регистре управления хоста. PHY показывает включение питания выдачей сигнала VBUS_VALID.
7. USB запитан, устройство подключено, процесс SRP завершён.

Протокол запроса сессии B-устройства

Программа ставит бит разрешения SRP в регистре Конфигурации Ядра и контроллер становится B-устройством. Можно запросить сессию от хоста.



VBUS_VALID = Сигнал рабочего V_{BUS} от PHY
B_VALID = B-peripheral valid session to PHY
DISCHRG_VBUS = Сигнал разрядки для PHY
SESS_END = Сигнал конца сессии для PHY
CHRG_VBUS = Сигнал зарядки V_{BUS} для PHY
DP = Линия данных плюс
DM = Линия данных минус

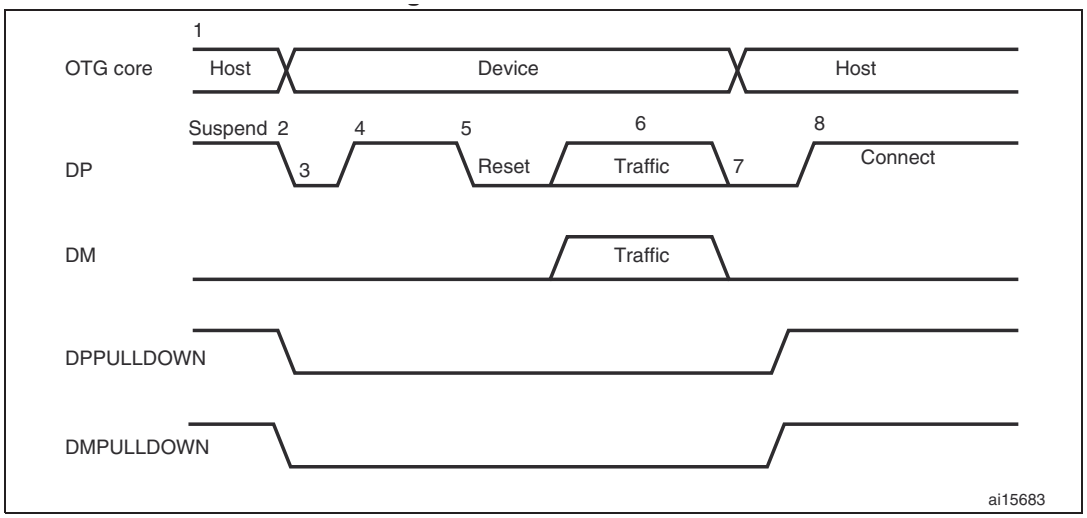
1. При простое шины хост останавливает порт и выключает питание. Через 3 ms простоя контроллер OTG_FS ставит бит ранней остановки в регистре прерываний ядра. Затем он там же ставит бит остановки USB и подаёт PHY сигнал разрядки V_{BUS}.
2. PHY выдаёт устройству конец сессии. Это исходное состояние SRP. Контроллеру OTG_FS нужно 2 ms SE0 перед запуском SRP.
   Для полноскоростных трансиверов USB 1.1, после снятия BSVLD (в OTG_FS_GOTGCTL) программе надо дождаться разряда V_{BUS} до 0.2 V. Время указывает производитель.
3. Ядро USB OTG говорит PHY ускорить разряд V_{BUS}.
4. Программа пускает SRP записью бита запроса а регистре Управления OTG. Контроллер OTG_FS выдаёт импульс на линии данных с последующим импульсом на V_{BUS}.
5. Хост обнаруживает SRP по одному из импульсов и включает V_{BUS}. PHY показывает устройству включение V_{BUS}.
6. Контроллер OTG_FS выдаёт импульс V_{BUS}.
   Хост начинает новую сессию включением V_{BUS}. Контроллер OTG_FS выдаёт прерывание

изменения состояния запроса сессии в регистре состояния прерываний OTG. Программа бит успешной сессии в регистре Управления OTG.

7. USB запитан, контроллер OTG_FS подключен, процесс SRP завершён.

Протокол беседы хоста A-устройства (HNP)

Протокол HNP переключает хост USB из A-устройства в B-устройство. Программа ставит бит разрешения HNP в регистре конфигурации ядра USB и контроллер становится A-устройством.

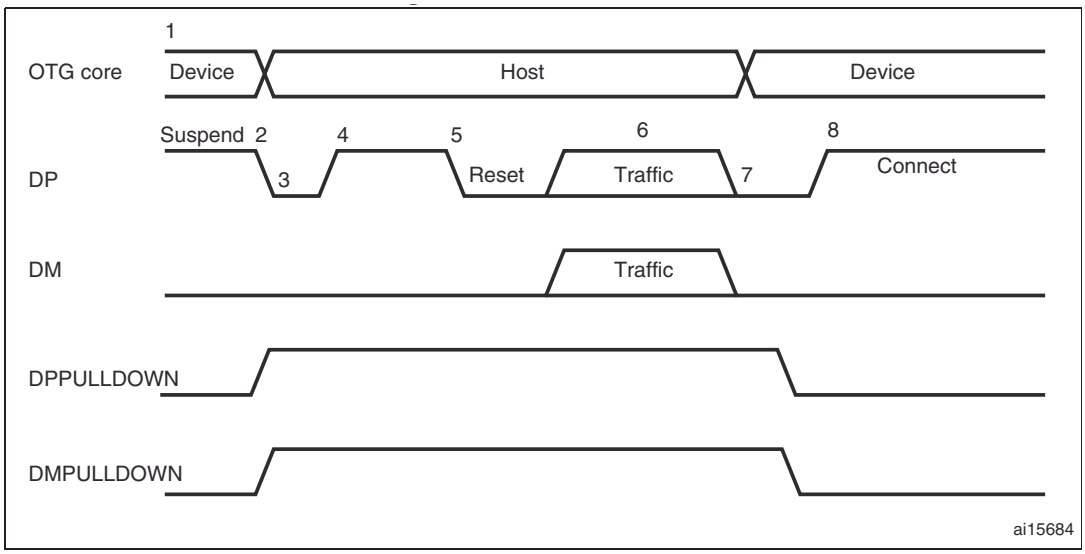


DPPULLDOWN = signal from core to PHY to enable/disable the pull-down on the DP line inside the PHY.
DMPULLDOWN = signal from core to PHY to enable/disable the pull-down on the DM line inside the PHY.

1. The OTG_FS controller sends the B-device a SetFeature b_hnp_enable descriptor to enable HNP support. The B-device's ACK response indicates that the B-device supports HNP. The application must set host Set HNP Enable bit in the OTG Control and status register to indicate to the OTG_FS controller that the B-device supports HNP.
2. When it has finished using the bus, the application suspends by writing the Port suspend bit in the host port control and status register.
   When the B-device observes a USB suspend, it disconnects, indicating the initial condition for HNP. The B-device initiates HNP only when it must switch to the host role; otherwise, the bus continues to be suspended.
   The OTG_FS controller sets the host negotiation detected interrupt in the OTG interrupt status register, indicating the start of HNP.
   The OTG_FS controller deasserts the DM pull down and DM pull down in the PHY to indicate a device role. The PHY enables the OTG_FS_DP pull-up resistor to indicate a connect for B-device. The application must read the current mode bit in the OTG Control and status register to determine device mode operation.
4. The B-device detects the connection, issues a USB reset, and enumerates the OTG_FS controller for data traffic.
5. The B-device continues the host role, initiating traffic, and suspends the bus when done.
   The OTG_FS controller sets the early suspend bit in the Core interrupt register after 3 ms of bus idleness. Following this, the OTG_FS controller sets the USB Suspend bit in the Core interrupt register.
6. In Negotiated mode, the OTG_FS controller detects the suspend, disconnects, and switches back to the host role. The OTG_FS controller asserts the DM pull down and DM pull down in the PHY to indicate its assumption of the host role.
7. The OTG_FS controller sets the Connector ID status change interrupt in the OTG Interrupt Status register. The application must read the connector ID status in the OTG Control and Status register to determine the OTG_FS controller operation as an A- device. This indicates the completion of HNP to the application. The application must read the Current mode bit in the OTG control and status register to determine host mode operation.
8. The B-device connects, completing the HNP process.

HNP B-устройства

HNP switches the USB host role from B-device to A-device. The application must set the HNP-capable bit in the Core USB configuration register to enable the OTG_FS controller to perform HNP as a B-device.



DPPULLDOWN = сигнал от ядра к PHY включения/выключения резистора подтяжки линии DP.
DMPULLDOWN = сигнал от ядра к PHY t включения/выключения резистора подтяжки линии DM.

1. A-устройство посылает дескриптор **SetFeature b_hnp_enable** для включения поддержки HNP. ACK от контроллера OTG_FS подтверждает сей факт. Программа ставит бит разрешения HNP устройства в регистре управления OTG извещая о поддержке HNP.
   Программа ставит бит запроса HNP в регистре управления OTG извещая контроллер о необходимости запуска HNP.
2. По завершению работы шины A-устройство останавливает порт в регистре Управления портом хоста. После 3 ms простоя шины контроллер OTG_FS ставит бит прерывания Раннего останова в регистре прерываний ядра. Затем контроллер OTG_FS ставит бит останова USB в регистре прерываний ядра.
   Контроллер OTG_FS отключается и A-устройство обнаруживает на шине SE0, показывающий HNP. Контроллер OTG_FS включает подтяжку DP и DM в PHY, подтверждая роль хоста.
   A-устройство в ответ включает свой резистор подтяжки OTG_FS_DP в течении 3 ms обнаружения SE0. Контроллер OTG_FS понимает это как подключение.
   Контроллер OTG_FS ставит прерывание изменения состояния беседы хоста в регистре состояния прерываний OTG, показывая статус HNP. Программа определяет его по биту в регистре Управления OTG. Текущий режим выясняют в регистре `OTG_FS_GINTSTS`.
3. Программа ставит бит сброса `PRST` в `OTG_FS_HPRT` и контроллер OTG_FS выдаёт сброс USB и выполняет переучёт A-устройства.
4. По завершению работы хостом контроллер OTG_FS останавливает шину записью бита останова порта в регистр Управления портом хоста.
5. В режиме Беседы, когда A-устройство обнаруживает останов, оно отключается и переключается в хост. Контроллер OTG_FS отключает резисторы подтверждая роль устройства.
6. Текущий режим читают в регистре `OTG_FS_GINTSTS`.
7. Контроллер OTG_FS подключается, завершая процесс HNP.

35. Высокоскоростной автономный USB (OTG_HS)

35.1. Введение в OTG_HS

Использованы сокращения:

HS	Высокая скорость
LS	Низкая скорость
USB	Универсальная последовательная шина
OTG	On-the-go (здесь "*автономный*")
PHY	Физический уровень
MAC	Контроллер доступа среды
PFC	Контроллер FIFO пакетов
UTMI	USB 2.0 transceiver macrocell interface (UTMI)
ULPI	UTMI+ Low Pin Interface

OTG_ HS это двухфункциональный (DRD) контроллер, поддерживающий режимы периферийного устройства и хоста USB, в соответствии с *On-The-Go Supplement to the USB 2.0 Specification*. Может работать только хостом и только устройством в соответствии с *USB 2.0 Specification*. В режиме хоста OTG_ HS поддерживает высокоскоростные (HS, 480 Мбит/с), полноскоростные (FS, 12 Мбит/с) и низкоскоростные (LS, 1.5 Мбит/с) передачи, а в режиме устройства только высокоскоростные (HS, 480 Мбит/с) и полноскоростные (FS, 12 Мбит/с) передачи. OTG_ HS поддерживает и HNP и SRP. В режиме OTG нужно только внешнее питание V_{BUS}.

35.2. Основные свойства OTG_HS

Их можно разделить на три категории: общие, хоста и устройства.

35.2.1. Общие свойства OTG_HS

Они таковы:

- Интерфейс USB-IF, сертифицированный для *Universal Serial Bus Specification Rev 2.0*
- Поддержка 3 интерфейсов PHY
  - Встроенный полноскоростной PHY
  - Интерфейс ULPI для внешнего высокоскоростного PHY.
- Встроенная поддержка хостом Протокола Сообщений (HNP) и Протокола Запроса Сессии SRP
- Выключение хостом V_{BUS} для экономии батарей систем OTG
- Поддержка контроля OTG уровня V_{BUS} внутренними компараторами
- Поддержка динамического переключения режимов хост/устройство
- Программная конфигурация в режиме:

— Устройство USB HS/FS с SRP (устройство B)
— Хост USB HS/FS/LS с SRP (устройство A)
— Двухфункциональный OTG_ FS USB

- Поддержка HS/FS/LS SOF с:

— Импульс SOF от PAD
— Импульс SOF от внутреннего таймера 2 (TIM2)
— Изменяемый период фрейма
— Прерывание конца фрейма

- Встроенный внутренний DMA с поддержкой общих ресурсов и настраиваемым типов пакетов AHB в режиме DMA
- Включает экономию энергии вроде останова системы во время остановки USB, выключения тактового домена ядра, управления питанием PHY и DFIFO
- Выделенная 4 КБ RAM с продвинутым управлением FIFO:

— Программируемое разделение пространства RAM для разных FIFO
— Все FIFO могут содержать много пакетов
— Динамическое выделение памяти

— Размеры FIFO не кратны степени двойки ради неразрывности памяти
- Гарантия максимальной полосы пропускания USB до одного фрейма без вмешательства системы

35.2.2. Свойства хоста OTG_HS

Это:
- Внешнее питание V_{BUS}.
- До 12 каналов хоста с динамической конфигурацией для любого типа передач USB.
- Встроенный аппаратный планировщик, имеющий:

— Аппаратную очередь для 8 прерываний плюс запросов изохронных передач

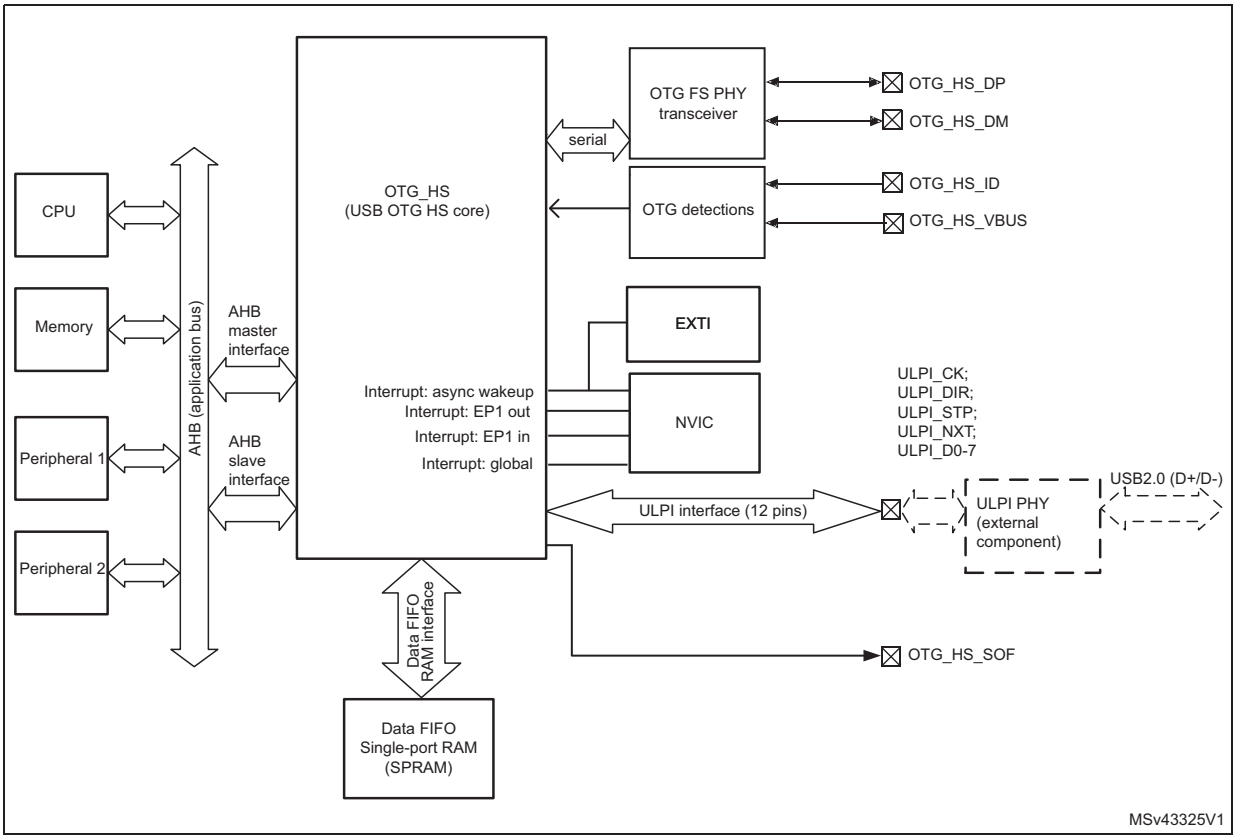
— Аппаратную очередь для 8 управляющих событий плюс запросов не-периодических групповых передач
- Управление общим RX FIFO, периодическим и не-периодическим TX FIFO.
- Динамическая подстройка времянки SOF.

35.2.3. Свойства устройства OTG_HS

Это:
- 1 двунаправленная конечная точка 0
- 5 конечных точек (EP) IN для Групповых, Прерывающих и Изохронных передач
- 5 конечных точек OUT для Групповых, Прерывающих и Изохронных передач
- Управление общим Rx FIFO и Tx-OUT FIFO
- Управление до 6 выделенными Tx-IN FIFO (по одному для каждой активной IN EP)
- Поддержка программного отключения.

35.3. Функциональное описание OTG_HS

Блок-схема OTG_HS



1. USB DMA не может напрямую адресовать внутреннюю Flash память.

35.3.1. Ножки OTG

Таблица 206. Для чего нужны ноги OTG_HS

Signal name	Signal type	Description
OTG_HS_DP	Digital input/output	USB OTG D+ line
OTG_HS_DM	Digital input/output	USB OTG D- line
OTG_HS_ID	Digital input	USB OTG ID
OTG_HS_VBUS	Analog input	USB OTG VBUS
OTG_HS_SOF	Digital output	USB OTG Start Of Frame (visibility)
OTG_HS_ULPI_CK	Digital input	USB OTG ULPI clock
OTG_HS_ULPI_DIR	Digital input	USB OTG ULPI data bus direction control
OTG_HS_ULPI_STP	Digital output	USB OTG ULPI data stream stop
OTG_HS_ULPI_NXT	Digital input	USB OTG ULPI next data stream request
OTG_HS_ULPI_D[0..7]	Digital input/output	USB OTG ULPI 8-bit bi-directional data bus

35.3.2. Высокоскоростной OTG PHY

Ядро USB OTG HS имеет встроенный интерфейс ULPI для соединения с внешним HS PHY.

35.3.3. Полноскоростной OTG PHY

OTG PHY включает компоненты:

- Модуль приёмопередатчика FS/LS хоста и устройства.
- Встроенный резистор подпорки линии ID для идентификации устройств A/B.
- Встроенные резисторы подпорки и подтяжки линий DP/DM, управляемые ядром OTG_HS. В режиме устройства к V_{BUS} подключается резистор подпорки DP (полноскоростная периферия). В режиме хоста к обеим линиям DP/DM подключаются резисторы подтяжки. Все резисторы подключаются динамически в соответствии с протоколом HNP.
- Резистор подпорки DP состоит из 2 раздельно подключаемых резисторов. Динамическая подстройка подпорки DP помогает бороться с шумами сигнала Tx/Rx.
- Компараторы уровня V_{BUS} с гистерезисом для определения V_{BUS} в Норме, A-B Сессия в Норме и порогов напряжения конца сессии. Они используются в протоколе запроса сессии (SRP), определяя начало и конец сессии и следят за уровнем V_{BUS}.
- Импульсная схема заряда/разряда V_{BUS} через во время SRP (слабый привод).

NB: Частота шины AHB должна быть выше 30 MHz.

35.4. Двухфункциональное устройство OTG

35.4.1. Определение линии ID

Режим хоста или периферии (по умолчанию) подразумевается зависящим от входной ножки ID. Состояние линии ID зависит от стороны кабеля USB, воткнутого в "разъёмаму" micro-AB.

- Если вставлена сторона B с оборванным проводом ID, то встроенный резистор подпорки создаёт высокий уровень ID и это Периферия.
- Если вставляется сторона A с заземлённым ID, то OTG_HS выдаёт прерывание изменения состояния линии ID (бит `CIDSCHG` в `OTG_HS_GINTSTS`) для инициализации программ хоста и автоматически переключается в режим хоста.

35.4.2. Двухфункциональное устройство HNP

Бит разрешения протокола HNP (бит `HNPCAP` в `OTG_HS_GUSBCFG`) позволяет ядру OTG_HS динамически переключаться из A-хоста в A-периферию и наоборот, и из B-периферии в B-хост. Текущее состояние устройства можно определить по сочетанию битов состояния ID (бит `CIDSTS` в `OTG_HS_GOTGCTL`) текущего режима (бит `CMOD` в `OTG_HS_GINTSTS`).

Программная модель HNP описана в *Секции 35.13*.

35.4.3. Двухфункциональное устройство SRP

Бит разрешения протокола SRP (бит `SRPCAP` в `OTG_FS_GUSBCFG`) разрешает ядру OTG_HS выключать питание от V_{BUS} для A-устройств, они и так всегда с ним работают.

Программная модель SRP A/B-устройств описана в *Секции 35.13*.

35.5. Периферийный USB

OTG_HS работает периферией USB как:

- OTG B-периферия
  - —OTG B-устройство по умолчанию, если кабель USB воткнут стороной B
- OTG A-периферия
  - —OTG A-устройство после того как HNP переключает OTG_HS в режим периферии
- B-устройство
  - —Если линия ID есть, работает и подключена к стороне B кабеля USB, и очищен бит HNPCAP в регистре OTG_HS_GUSBCFG.
- Только периферия:
  - —Стоит бит FDMOD в регистре OTG_HS_GUSBCFG. В этом случае линия ID игнорируется, даже если она есть в разъёме.

NB: При питании устройств в режимах B и только-периферии от V_{BUS}, V_{DD} надо подавать через внешний регулятор.

35.5.1. Устройство с SRP

Разрешение протокола SRP включается битом SRPCAP в регистре OTG_HS_GUSBCFG. Таким образом устройство A может выключать V_{BUS} при остановке сессии USB.

35.5.2. Состояния устройств

Запитанное

Обнаружение V_{BUS} при B-Сессии делает состояние Запитанным. OTG_HS автоматически подключает резистор подпорки сигнала DP полноскоростного хоста и выдаёт запрос прерывания сессии (бит SRQINT в регистре OTG_HS_GINTSTS).

Также проверяется рабочий уровень V_{BUS} от хоста. Если в B-сессии V_{BUS} падает, то OTG_HS автоматически отключается и выдаёт прерывание выхода из запитанного режима (бит SEDET в регистре OTG_HS_GOTGINT).

В запитанном режиме OTG_HS ждёт сигнала сброса от хоста, и всё. При появлении сброса (USBRST в OTG_HS_GINTSTS) выдаётся прерывание. По завершении сброса выдаётся прерывание конца переучёта (ENUMDNE в OTG_HS_GINTSTS) и OTG_HS идёт в состояние по умолчанию.

Программное отключение

Установка бита SDIS в регистре OTG_HS_DCTL отключает резистор подпорки DP и хост выдаёт прерывание отключения даже при реально воткнутом в порт хоста кабеле USB.

По умолчанию

В этом режиме OTG_HS ждёт от хоста команду SET_ADDRESS. По ней в поле DAD регистра OTG_HS_DCFG пишется адрес устройства, OTG_HS становится адресованным и может отвечать на запросы хоста.

Остановленное

Периферийный OTG_HS постоянно отслеживает активность USB. После 3 ms простоя USB выдаётся прерывание раннего останова (бит ESUSP в OTG_HS_GINTSTS) и через 3 ms подтверждается прерыванием останова (бит USBSUSP в OTG_HS_GINTSTS). Автоматически ставится бит состояния (SUSPSTS в OTG_HS_DSTS) и OTG_HS стопорится.

Выйти из останова можно самостоятельно установкой бита удалённой побудки (RWUSIG в OTG_HS_DCTL) и его снятием после задержки от 1 до 15 ms.

При появлении сигнала побудки от хоста выдаётся прерывание (бит WKUPINT в регистре OTG_HS_GINTSTS) и автоматически снимается бит останова устройства.

35.5.3. Конечные точки устройств

Ядро OTG_HS обслуживает следующие конечные точки USB:

- Управляющая точка 0:

—Двунаправленная, только для управляющих сообщений.

—Отдельные наборы регистров для входных и выходных передач.

— Собственные регистры управления (OTG_HS_DIEPCTL0/OTG_HS_DOEPCTL0), конфигурации передач (OTG_HS_DIEPTSIZ0/OTG_HS_DIEPTSIZ0) и состояния/прерываний (OTG_HS_DIEPINT0/OTG_HS_DOEPINT0) с несколько отличными от других точек битами.

• 5 конечных точек IN

— Поддерживают изохронные, групповые и прерывающие передачи.

— Собственные регистры управления (OTG_HS_DIEPCTLx), конфигурации передач (OTG_HS_DIEPTSIZx) и состояния/прерываний (OTG_HS_DIEPINTx).

— Регистр маски общих прерываний (включая EP0) Устройства IN (OTG_HS_DIEPMSK).

— Поддержка прерывания незавершённых изохронных передач IN (бит IISOIXFR в регистре OTG_HS_GINTSTS). Оно выставляется во время прерывания конца периодического фрейма (OTG_HS_GINTSTS/EOPF).

• 5 конечных точек OUT

— Поддерживают изохронные, групповые и прерывающие передачи.

— Собственные регистры управления (OTG_FS_DOEPCTLx), конфигурации передач (OTG_FS_DOEPTSIZx) и состояния/прерываний (OTG_FS_DOEPINTx).

— Регистр маски общих прерываний (включая EP0) Устройства Out (OTG_FS_DOEPMSK).

— Поддержка прерывания незавершённых изохронных передач OUT (бит INCOMPISOOUT в OTG_FS_GINTSTS). Оно выставляется во время прерывания конца периодического фрейма (OTG_FS_GINTSTS/EOPF).

Управление конечной точкой

• Через регистры управления (DIEPCTLx/DOEPCTLx) можно:

— Включить/выключить точку

— Активировать точку в текущей конфигурации

— Установить тип передачи USB (изохронная, групповая, прерывающая)

— Задать размер пакета

— Задать номер Tx-FIFO для точки IN

— Задать ожидаемый или передаваемый DATA0/DATA1 PID (только групповые/прерывающие)

— Задать чётный/нечётный фрейм изохронных передач

— Установкой бита NAK всегда выдавать хосту не-ответ независимо от состояния FIFO

— Установкой бита STALL всегда стопорить токены хоста этой точке

— Установить режим SNOOP точки OUT, чтобы не проверяла CRC принятых данных.

Передачи конечной точки

Размер и состояние передачи хранятся в регистрах DIEPTSIZx/DOEPTSIZx. Писать в него нужно до запуска конечной точки. После запуска точки регистр доступен только по чтению, пишет в него OTG_HS.

Можно установить параметры:

- Размер передачи в байтах
- Число пакетов всей передачи

Состояние/прерывания конечной точки

События со стороны USB и AHB хранятся в регистрах прерываний (DIEPINTx/DOPEPINTx). Читать их надо при стоящих битах прерываний (бит OEPINT в OTG_HS_GINTSTS для точки OUT и бит IEPINT в OTG_HS_GINTSTS для точки IN). Перед чтением этих регистров надо получить номер прервавшей конечной точки из общего регистра прерываний (OTG_HS_DAINT). Снятие бита в этих регистрах чистит соответствующие биты регистров DAINT и GINTSTS.

Ядро периферии проверяет и выдаёт прерывания для:

- Завершение передачи для сторон AHB и USB
- Фаза SETUP завершена (только управляющие OUT)
- FIFO передачи наполовину или полностью пуст (точки IN)
- Хосту передан ответ NAK (только изохронные IN)
- Принят токен IN при пустом Tx-FIFO (только групповые IN и прерывающие IN)
- Принят токен OUT для ещё не включённой точки

- Ошибка перекрёстного шума
- Программа выключила точку
- Программа включила NAK точки (только изохронные IN)
- Принята череда более 3 пакетов SETUP (только управляющая OUT)
- Таймаут (только управляющие IN)
- Отброшен изохронный пакет OUT без выдачи прерывания.

35.6. Хост USB

OTG_HS работает USB хостом как:

• OTG A-хост

– состояние A-устройства по умолчанию когда вставлена сторона A кабеля USB

• OTG B-host

– B-устройство после HNP переключения в режим хоста

• A-устройство

– Если линия ID есть, работает и подключена к A-стороне кабеля USB, и снят бит HNPCAP в регистре OTG_HS_GUSBCFG. К линиям DP/DM автоматически подключаются встроенные резисторы подтяжки.

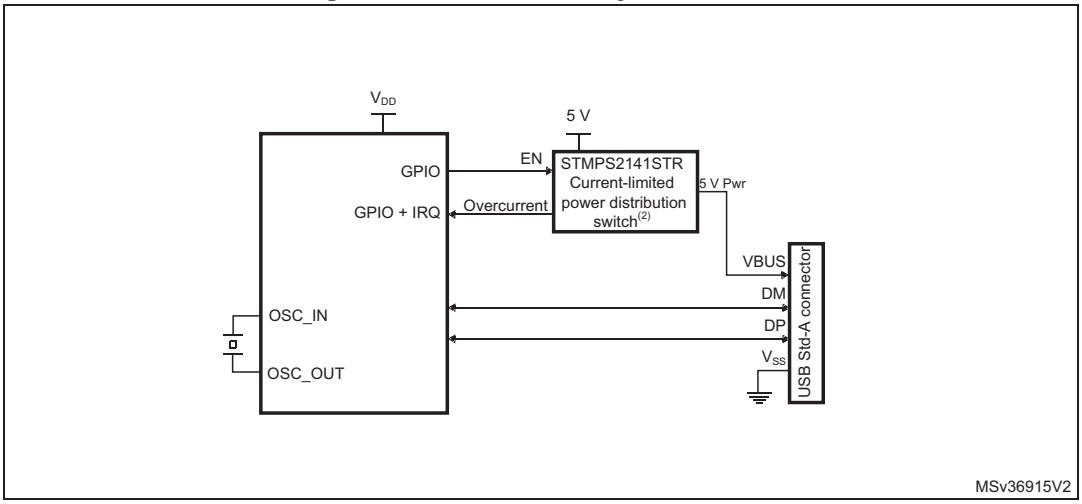
• Только хост

– Включается битом FHMOD в регистре OTG_HS_GUSBCFG. В этом случае линия ID игнорируется даже при её наличии в разъёме USB. К линиям DP/DM автоматически подключаются встроенные резисторы подтяжки.

NB: Встроенный источник 5V V_{BUS} не поддерживается, на линию надо подавать внешнее напряжение. Внешнюю подзарядку можно подавать через выход GPIO. Это требуется для конфигураций OTG A-хост, A-устройства и только-хост.

Уровень V_{BUS} при работе USB проверяется на входе V_{BUS}. Сигнал перегрузки источника подзарядки V_{BUS} можно подавать на вход GPIO для выдачи прерывания. Прерывание перегрузки должно немедленно выключать V_{BUS}.

Подключение только-хост USB



35.6.1. Хост с SRP

Поддержка SRP включается битом SRPCAP в регистре OTG_HS_GUSBCFG. Теперь можно выключать питание V_{BUS} при остановке сессии USB.

35.6.2. Состояния хоста USB

Питание порта хоста

Встроенное питание 5V V_{BUS} не поддерживается. Стало быть нужно использовать внешнее. Внешнюю подзарядку можно подавать через выходы GPIO. При питании V_{BUS} от GPIO надо ставить бит PPWR в регистре OTG_HS_HPRT.

V_{BUS} в норме

При включённом HNP или SRP ножку контроля V_{BUS} (PB13) надо подключать к V_{BUS}. Непредвиденное падение напряжения V_{BUS} ниже порога (4.25 V) запускается прерывание конца сессии (бит SEDET в OTG_HS_GOTGINT). Затем надо отключить V_{BUS} и снять бит питания порта.

При выключенных HNP SRP ножку контроля V_{BUS} (PB13) к V_{BUS} подключать не надо.

Сигнал перегрузки источника подзарядки V_{BUS} можно подавать на вход GPIO для выдачи прерывания. Прерывание перегрузки должно немедленно выключать V_{BUS} и снять бит питания порта.

Обнаружение хостом подключения периферии

При включённых SRP или HNP ядро OTG_HS обнаруживает подключение периферии USB или устройства B только при рабочем уровне питания (V_{BUS} выше 4.75 V). При подключении внешнего устройства B ядро OTG_HS выдаёт прерывание установкой бита PCDET в регистре OTG_HS_HPRT.

При выключенных HNP и SRP периферия USB и устройства B обнаруживаются сразу после подключения. Ядро OTG_HS выдаёт прерывание установкой бита PCDET в регистре OTG_HS_HPRT.

Обнаружение хостом отключения периферии

Прерывание выдаётся установкой бита DISCINT в регистре OTG_HS_GINTSTS.

Переучёт хоста

После обнаружения подключённого устройства хост должен начать переучёт, посылая новому устройству сброс USB и команды конфигурации.

Перед посылкой сброса USB надо дождаться прерывания конца антидребезга (бит DBCDNE в OTG_HS_GOTGINT), появляющегося при подключении резисторов к DP (FS) или DM (LS).

Сигнал сброса USB подаётся снятием бита PRST в регистре OTG_HS_HPRT минимум на 10 ms и максимум на 20 ms. Считать это время надо осторожно.

После сброса USB выдаётся прерывание изменения состояния включения (бит PENCHNG в OTG_HS_HPRT). Теперь можно узнать скорость подключённого устройства (бит PSPD в OTG_HS_HPRT) и начать выдавать SOF (FS) или Живи (Keep alive) (LS) и посылать команды конфигурации.

Остановка хоста

Активность USB останавливают установкой бита PSUSP в регистре OTG_HS_HPRT. OTG_HS прекращает посылку SOFs и идёт в режим останова. Выйти из него можно по инициативе устройства. При этом выдаётся прерывание побудки (бит WKUPINT в регистре OTG_HS_GINTSTS), автоматически ставится бит восстановления (бит PRES в OTG_HS_HPRT) и на USB выдаётся сигнал восстановления. Через некоторое время бит восстановления снимается и и возобновляется выдача SOF.

При побудке по инициативе хоста автоматически ставится бит восстановления через некоторое время он снимается и возобновляется выдача SOF.

35.6.3. Каналы хоста

Ядро OTG_HS обслуживает 12 каналов хоста. Хост не может одновременно поддерживать больше 8 запросов передач. Если их больше, то контроллер хоста (HCD) должен перераспределить каналы после их освобождения завершением передачи.

Канал хоста поддерживает IN/OUT и любой тип периодических/не-периодических передач. Канал имеет собственные регистры управления (HCCHAR*x*), конфигурации (HCTSIZ*x*) состояния/прерываний (HCINT*x*) со связанным регистром маски (HCINTMSK*x*).

Управление каналом хоста

- Регистр HCCHAR*x* позволяет:
  - — Включать/выключать канал.
  - — Ставить скорость HS/FS/LS целевой периферии USB
  - — Задавать адрес целевой периферии USB
  - — Задавать номер конечной точки целевой периферии USB
  - — Задавать направление передач IN/OUT
  - — Задавать тип передач USB (управляющая, групповая, прерывающая, изохронная)
  - — Задавать максимальный размер пакета (MPS)

— Задавать периодические передачи для выполнения во время чётных/нечётных фреймов

Передачи канала хоста

Длина и состояние передач содержатся в регистре HCTSIZx. Писать в него надо до запуска канала. После запуска регистр доступен только по чтению.

- Можно задавать:

— размер передачи в байтах

— число пакетов всей передачи

— начальный PID данных

Состояние/прерывания канала хоста

События со стороны USB и AHB отмечаются в регистре HCINTx. Читать его нужно при стоящем бите прерывания (бит HCINT в регистре OTG_HS_GINTSTS). Перед этим нужно прочесть номер прервавшего канала из регистра HCAINT. Снимаются биты прерывания очисткой соответствующих битов в регистрах HAINT и GINTSTS. Маска прерываний есть в регистре OTG_HS_HCINTMSKx.

- Ядро хоста проверяет следующие события прерываний:

— Завершение передачи на сторонах AHB и USB

— Остановка канала по концу передачи, ошибке передачи USB или команде выключения

— Передающий FIFO наполовину или полностью пуст (точки IN)

— Принят ответ ACK

— Принят ответ NAK

— Принят ответ STALL

— Ошибка передачи USB по ошибке CRC, таймауту, битовому заполнению, сбойному EOP

— Ошибка перекрёстных шумов

— Переполнение фрейма

— Ошибка переключения данных

35.6.4. Планировщик хоста

По началу фрейма хост сначала выполняет одновременно затребованные периодические (изохронные и прерывающие) передачи, а затем не-периодические (управляющие и групповые). Это делается с помощью очередей запросов (по одной для каждого вида передач). Каждая очередь имеет до 8 элементов. Элемент очереди содержит номер IN или OUT канала и информацию к его обработке через USB. Порядок запросов в очереди определяет последовательность пересылок по USB.

Если для текущего фрейма ещё запланирована изохронная или прерывающая передача, то в его конце выдаётся прерывание незавершённой периодической передачи (IPXFR в OTG_HS_GINTSTS). Управляет очередями запросов только ядро OTG HS. Регистры FIFO и состояния периодической (HPTXSTS) и не-периодической (HNPTXSTS) очередями доступны только по чтению. Они содержат:

- Число свободных строк (до 8)
- Доступное место в Tx-FIFO (передачи OUT)
- Токен IN/OUT, номер канала и состояние.

Перед отправкой запроса в очередь нужно убедиться в наличии там свободного места проверив биты PTXQSAV в регистре OTG_HS_HNPTXSTS или NPTQXSAV в регистре OTG_HS_HNPTXSTS.

35.7. Выдача SOF

Ядро OTG FS помогает отслеживать и конфигурировать размещение SOF в хосте и периферии, равно как и выдачу импульсов SOF. Это полезно при подключении звуковой периферии для синхронизации изохронного потока между устройством и PC.

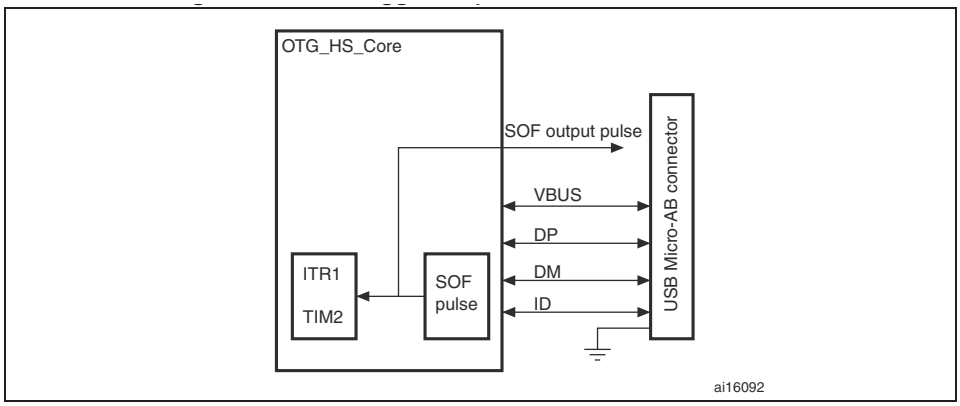
35.7.1. SOF хоста

В режиме хоста число тактов PHY между двумя последовательными токенами SOF (FS) или Keep-alive (LS) пишутся в регистр интервала фреймов (HFIR). Прерывание выдаётся по каждому началу фрейма (бит SOF в OTH_HS_GINTSTS). Номер текущего фрейма и остаток времени до следующего отслеживается в регистре HFNUM.

Выдаваемый по токену SOF импульс, длительностью 20 системных тактов можно направлять на выходную ножку SOF с помощью бита SOFOUTEN в регистре OTG_HS_GCCFG. Также импульс SOF

внутренне подключён к входу запуска TIM2. Оно разрешается битами ITR1_RMP регистра TIM2_OR.

Подключение SOF к TIM2_ITR1



35.7.2. SOF устройства

В режиме устройства при получении токена SOF (бит SOF в OTG_HS_GINTSTS) выдаётся прерывание. Номер фрейма можно почитать в регистре состояния устройства (FNSOF в регистре OTG_HS_DSTS). Выдаваемый по токену SOF импульс, длительностью 12 системных тактов можно направлять на выходную ножку SOF с помощью бита SOFOUTEN в регистре OTG_HS_GCCFG. Также импульс SOF внутренне подключён к входу запуска TIM2.

Время выдачи прерывания конца периодического фрейма (GINTSTS/EOPF) определено как 80%, 85%, 90% или 95% ожидаемого времени фрейма в регистре конфигурации устройства (PFIVL в OTG_HS_DCFG). Так можно определить конец всей изохронной передачи.

35.8. Экономные режимы OTG

Таблица 207. Совместимость экономных режимов STM32 с OTG

Режим	Описание	Совместимость с USB
Run	MCU совсем работает	Нужна при не остановленном хосте USB not in.
Sleep	Выход USB из останова выводит устройство из Sleep. Регистры периферии сохраняются.	Доступно при остановленном хосте USB.
Stop	Выход USB из останова выводит устройство из Stop. Регистры периферии сохраняются(1).	Доступно при остановленном хосте USB.
Standby	Выкл. питания. Периферию надо инициализировать.	Не совместимо с работой USB

Потребление энергии OTG PHY определяется тремя битами регистра конфигурации ядра:

- Выключение PHY (GCCFG/PWRDWN)
- Включение компараторов V_{BUS} устройства A (GCCFG/VBUSASEN).
- Включение компараторов V_{BUS} устройства B (GCCFG/VBUSASEN)

У остановленного USB экономить можно так:

- Остановка тактов PHY (бит STPPCLK в OTG_HS_PCGCCTL).
- Запрет HCLK (GATEHCLK в OTG_HS_PCGCCTL).
- Стоп системы USB.

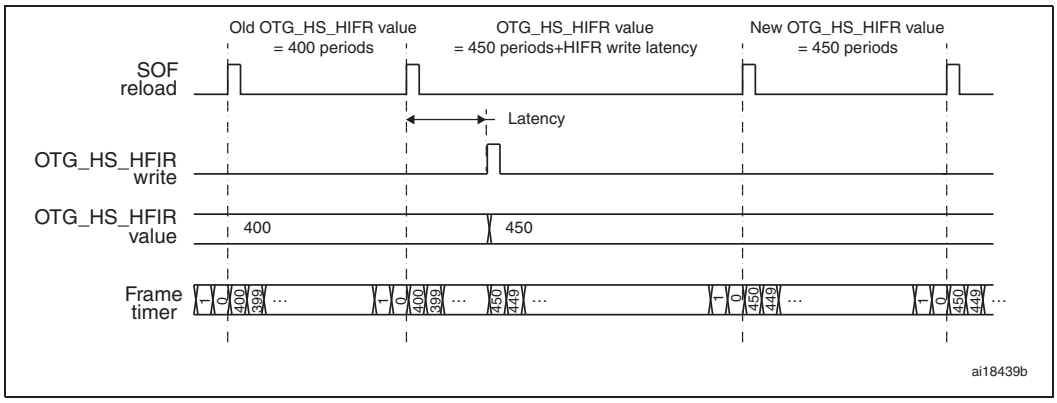
Во благо экономии, FIFO данных USB тактируются только при доступе из ядра OTG_HS.

35.9. Динамическое обновление регистра OTG_HS_HFIR

Ядро USB в режиме хоста может подстраивать время выдачи фреймов микро-SOF для синхронизации внешнего устройства.

Если регистр OTG_HS_HFIR изменился внутри текущего фрейма микро-SOF, то изменение периода SOF вступит в силу в следующем фрейме.

Динамическое обновление OTG_HS_HFIR



35.10. Размещение FIFO в RAM

35.10.1. Режим устройства

Память для приёмного FIFO: надо выделить память для 10 пакетов SETUP, ядро к этой памяти не суётся. Один участок выделяется для Глобального OUT NAK. Состояние принятого пакета пишется вместе с данными.

Поэтому для приёма пакетов надо выделять минимум (Максимальный_ Размер_Пакета/ 4) + 1. При нескольких изохронных точках надо выделить не меньше двух таких участков. Обычно выделяют два таких участка.

Вместе с последним пакетом передачи конечной точки в FIFO пихается состояние завершения передачи. Для неё нужен один участок на точку OUT.

Память для передающего FIFO: минимальный размер FIFO равен максимальному размеру пакета точки IN.

NB: Больший размер FIFO точки IN повышает производительность USB.

35.10.2. Режим хоста

Память для приёмного FIFO

Состояние принятого пакета пишется вместе с данными.

Поэтому для приёма пакетов надо выделять минимум (Максимальный_ Размер_Пакета/ 4) + 1. При нескольких изохронных точках надо выделить не меньше двух таких участков. Обычно выделяют два таких участка.

Вместе с последним пакетом передачи конечной точки в FIFO пихается состояние завершения передачи. Для неё нужен один участок на канал.

Память для передающего FIFO

Минимальный размер не-периодического передающего FIFO равен самому большому размеру пакетов не-периодических каналов OUT. Обычно выделяют два максимальных размера пакета.

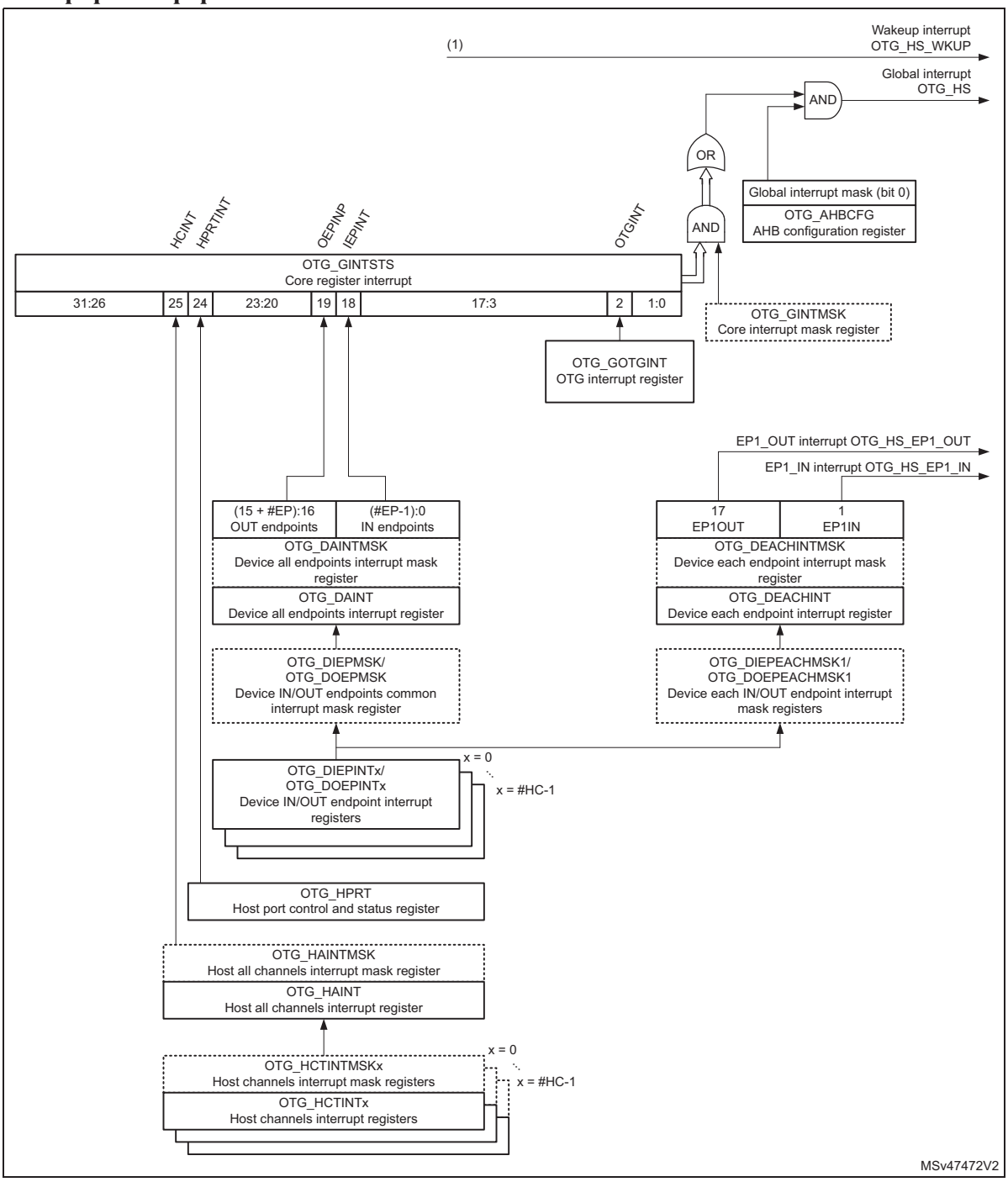
Минимальный размер периодического передающего FIFO равен самому большому размеру пакетов периодических каналов OUT. Если есть изохронные точки OUT, то максимальный размер этого канала надо удвоить.

NB: Больший размер не-периодического передающего FIFO повышает производительность USB.

В режиме DMA в FIFO надо выделить по одному участку для регистра адреса DMA (`HCDMAn`) для каждого канала.

35.11. Прерывания OTG_HS

Иерархия прерываний



1. OTG_HS_WKUP становится высоким (активным) при выходе из L1 SLEEP или L2 SUSPEND.

35.12. Регистры управления и состояния OTG_HS

Контроллер OTG_HS управляется через регистры состояния и управления (CSR) по шине AHB. Это 32-бит регистры, выравненные на границу 32-бит, доступные 32-бит словами.

CSR разделяются на:

- Общие регистры ядра
- Регистры режима хоста
- Общие регистры хоста
- CSR портов хоста
- Регистры каналов хоста
- Регистры режима устройства
- Общие регистры устройства
- Регистры конечных точек устройства
- Регистры управления питанием и тактами

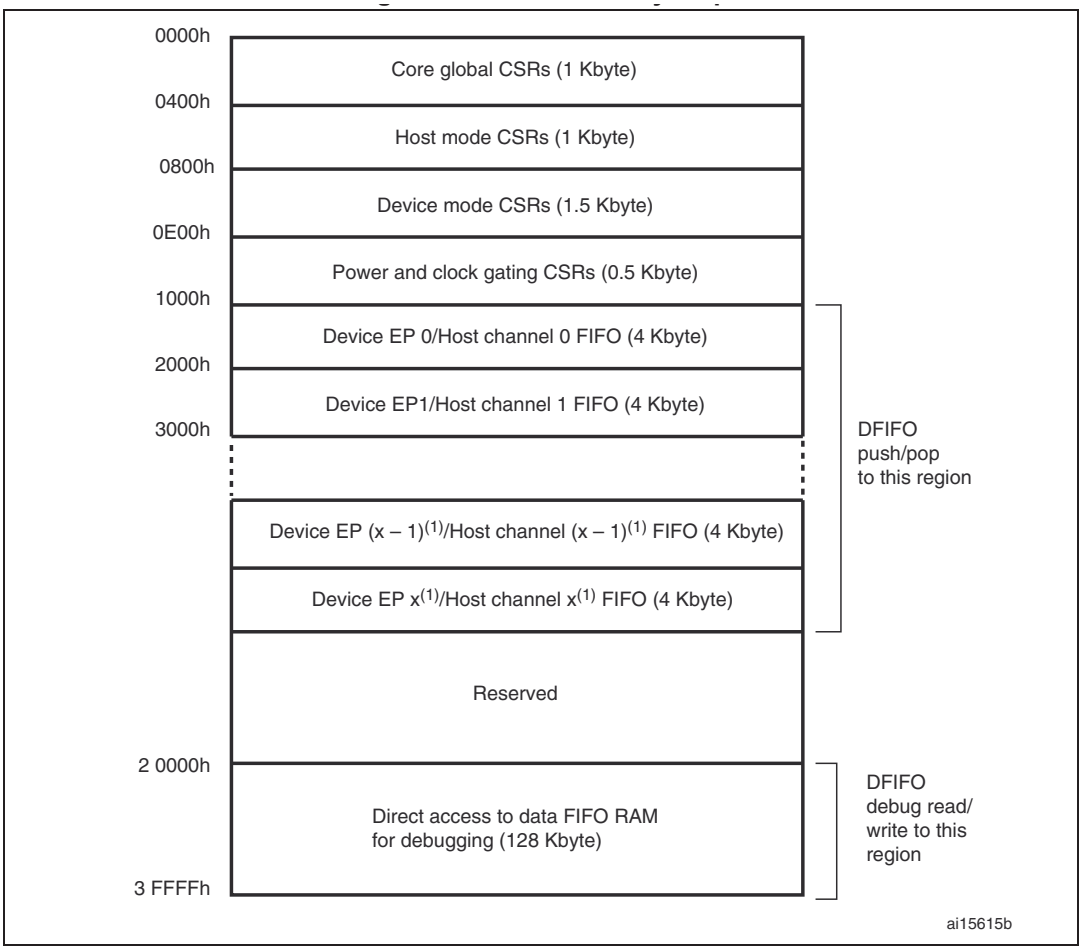
- Регистры доступа к FIFO данных (DFIFO)

Одновременно в режимах хоста и ядра доступны только Общие регистры ядра, Регистры управления питанием и тактами, Регистры доступа к FIFO данных и CSR портов хоста. Работая в одном режиме, контроллер OTG_FS не должен обращаться к регистрам другого режима. Это вызовет прерывание несовпадения режима (бит MMIS в регистре OTG_HS_GINTSTS). При переключении режимов регистры нужно переписать как при выходе из сброса.

35.12.1. Карта памяти CSR

Регистры режимов хоста и устройства занимают разные адреса. Все регистры сделаны в домене тактов AHB.

Карта памяти CSR



1. x=5 у устройства и x=11 у хоста.

Карта общих CSR

Эти регистры доступны в режимах хоста и устройства.

Таблица 208. Общие CSR

Имя	Смещение адреса	Имя регистра
OTG_HS_GOTGCTL	0x000	OTG_HS control and status register (OTG_HS_GOTGCTL)
OTG_HS_GOTGINT	0x004	OTG_HS interrupt register (OTG_HS_GOTGINT)
OTG_HS_GAHBCFG	0x008	OTG_HS AHB configuration register (OTG_HS_GAHBCFG)
OTG_HS_GUSBCFG	0x00C	OTG_HS USB configuration register (OTG_HS_GUSBCFG)
OTG_HS_GRSTCTL	0x010	OTG_HS reset register (OTG_HS_GRSTCTL)
OTG_HS_GINTSTS	0x014	OTG_HS core interrupt register (OTG_HS_GINTSTS)
OTG_HS_GINTMSK	0x018	OTG_HS interrupt mask register (OTG_HS_GINTMSK)
OTG_HS_GRXSTSR	0x01C	OTG_HS Receive status debug read/OTG status read and pop registers (OTG_HS_GRXSTSR/OTG_HS_GRXSTSP)
OTG_HS_GRXSTSP	0x020	
OTG_HS_GRXFSIZ	0x024	OTG_HS Receive FIFO size register (OTG_HS_GRXFSIZ)
OTG_HS_GNPTXFSIZ/ OTG_HS_TX0FSIZ	0x028	OTG_HS nonperiodic transmit FIFO size/Endpoint 0 transmit FIFO size register (OTG_HS_GNPTXFSIZ/OTG_HS_TX0FSIZ)

Имя	Смещение адреса	Имя регистра
OTG_HS_GNPTXSTS	0x02C	OTG_HS nonperiodic transmit FIFO/queue status register (OTG_HS_GNPTXSTS)
OTG_HS_GCCFG	0x038	OTG_HS general core configuration register (OTG_HS_GCCFG)
OTG_HS_CID	0x03C	OTG_HS core ID register (OTG_HS_CID)
OTG_HS_HPTXFSIZ	0x100	OTG_HS Host periodic transmit FIFO size register (OTG_HS_HPTXFSIZ)
OTG_HS_DIEPTXFx	0x104 0x108 ... 0x118	OTG_HS device IN endpoint transmit FIFO size register (OTG_HS_DIEPTXFx) (x = 1..7, where x is the FIFO_number)

Карта CSR хоста

Эти регистры писать при каждом входе в режим хоста.

Таблица 209. CSR режима хоста

Имя	Смещение адреса	Имя регистра
OTG_HS_HCFG	0x400	OTG_HS host configuration register (OTG_HS_HCFG)
OTG_HS_HFIR	0x404	OTG_HS Host frame interval register (OTG_HS_HFIR)
OTG_HS_HFNUM	0x408	OTG_HS host frame number/frame time remaining register (OTG_HS_HFNUM)
OTG_HS_HPTXSTS	0x410	OTG_HS_Host periodic transmit FIFO/queue status register (OTG_HS_HPTXSTS)
OTG_HS_HAINT	0x414	OTG_HS Host all channels interrupt register (OTG_HS_HAINT)
OTG_HS_HAINTMSK	0x418	OTG_HS host all channels interrupt mask register (OTG_HS_HAINTMSK)
OTG_HS_HPRT	0x440	OTG_HS host port control and status register (OTG_HS_HPRT)
OTG_HS_HCCHARx	0x500 0x520 ... 0x660	OTG_HS host channel-x characteristics register (OTG_HS_HCCHARx) (x = 0..11, where x = Channel_number)
OTG_HS_HCSPLTx	0x504	OTG_HS host channel-x split control register (OTG_HS_HCSPLTx) (x = 0..11, where x = Channel_number)
OTG_HS_HCINTx	0x508	OTG_HS host channel-x interrupt register (OTG_HS_HCINTx) (x = 0..11, where x = Channel_number)
OTG_HS_HCINTMSKx	0x50C	OTG_HS host channel-x interrupt mask register (OTG_HS_HCINTMSKx) (x = 0..11, where x = Channel_number)
OTG_HS_HCTSIZx	0x510	OTG_HS host channel-x transfer size register (OTG_HS_HCTSIZx) (x = 0..11, where x = Channel_number)
OTG_HS_HCDMAx	0x514	OTG_HS host channel-x DMA address register (OTG_HS_HCDMAx) (x = 0..11, where x = Channel_number)

Карта CSR устройства

Эти регистры писать при каждом входе в режим устройства.

Таблица 210. CSR режима устройства

Имя	Смещение адреса	Имя регистра
OTG_HS_DCFG	0x800	OTG_HS device configuration register (OTG_HS_DCFG)
OTG_HS_DCTL	0x804	OTG_HS device control register (OTG_HS_DCTL)
OTG_HS_DSTS	0x808	OTG_HS device status register (OTG_HS_DSTS)
OTG_HS_DIEPMSK	0x810	OTG_HS device IN endpoint common interrupt mask register (OTG_HS_DIEPMSK)
OTG_HS_DOEPMSK	0x814	OTG_HS device OUT endpoint common interrupt mask register (OTG_HS_DOEPMSK)
OTG_HS_DAINT	0x818	OTG_HS device all endpoints interrupt register (OTG_HS_DAINT)
OTG_HS_DAINTMSK	0x81C	OTG_HS all endpoints interrupt mask register (OTG_HS_DAINTMSK)
OTG_HS_DVBUSDIS	0x828	OTG_HS device V_{BUS} discharge time register (OTG_HS_DVBUSDIS)
OTG_HS_DVBUSPULSE	0x82C	OTG_HS device V_{BUS} pulsing time register (OTG_HS_DVBUSPULSE)
OTG_HS_DTHRCTL	0x830	OTG_HS Device threshold control register (OTG_HS_DTHRCTL)
OTG_HS_DIEPEMPMSK	0x834	OTG_HS device IN endpoint FIFO empty interrupt mask register: (OTG_HS_DIEPEMPMSK)
OTG_HS_DEACHINT	0x838	OTG_HS device each endpoint interrupt register (OTG_HS_DEACHINT)

Имя	Смещение адреса	Имя регистра
OTG_HS_DEACHINTMSK	0x83C	OTG_HS device each endpoint interrupt register mask (OTG_HS_DEACHINTMSK)
OTG_HS_DIEPEACHMSK1	0x844	OTG_HS device each in endpoint-1 interrupt register (OTG_HS_DIEPEACHMSK1)
OTG_HS_DOEPEACHMSK1	0x884	OTG_HS device each OUT endpoint-1 interrupt register (OTG_HS_DOEPEACHMSK1)
OTG_HS_DIEPCTLx	0x900 0x920 ... 0x9C0	OTG device endpoint-x control register (OTG_HS_DIEPCTLx) (x = 0..7, where x = Endpoint_number)
OTG_HS_DIEPINTx	0x908	OTG_HS device endpoint-x interrupt register (OTG_HS_DIEPINTx) (x = 0..7, where x = Endpoint_number)
OTG_HS_DIEPTSIZ0	0x910	OTG_HS device IN endpoint 0 transfer size register (OTG_HS_DIEPTSIZ0)
OTG_HS_DIEPDMAx/ OTG_HS_DOEPDMAx	0x914/0xB14	OTG_HS device endpoint-x DMA address register (OTG_HS_DIEPDMAx / OTG_HS_DOEPDMAx) (x = 1..5, where x = Endpoint_number)
OTG_HS_DTXFSTSx	0x918	OTG_HS device IN endpoint transmit FIFO status register (OTG_HS_DTXFSTSx) (x = 0..5, where x = Endpoint_number)
OTG_HS_DIEPTSIZx	0x930 0x950 ... 0x9D0	OTG_HS device endpoint-x transfer size register (OTG_HS_DOEPTSIZx) (x = 1..5, where x = Endpoint_number)
OTG_HS_DOEPCTL0	0xB00	OTG_HS device control OUT endpoint 0 control register (OTG_HS_DOEPCTL0)
OTG_HS_DOEPSIZ0	0xB10	OTG_HS device OUT endpoint 0 transfer size register (OTG_HS_DOEPTSIZ0)
OTG_HS_DOEPCTLx	0xB20 0xB40 ... 0xBC0	OTG device endpoint-x control register (OTG_HS_DIEPCTLx) (x = 0..7, where x = Endpoint_number)
OTG_HS_DOEPINTx	0xB08 0xB28 ... 0xBC8	OTG_HS device endpoint-x interrupt register (OTG_HS_DOEPINTx) (x = 0..7, where x = Endpoint_number)
OTG_HS_DOEPTSIZx	0xB30 0xB50 ... 0xBD0	OTG_HS device endpoint-x transfer size register (OTG_HS_DOEPTSIZx) (x = 1..5, where x = Endpoint_number)

Карта регистров доступа к FIFO данных (DFIFO)

Регистры доступны в обоих режимах хоста и устройства. У каналов хоста IN, FIFO доступен по чтению. У каналов хоста OUT, FIFO доступен по записи.

Таблица 211. Карта регистров доступа к FIFO данных (DFIFO)

Секция регистров доступа к FIFO	Диапазон адресов	Доступ
Device IN Endpoint 0/Host OUT Channel 0: DFIFO Write Access Device OUT Endpoint 0/Host IN Channel 0: DFIFO Read Access	0x1000–0x1FFC	w r
Device IN Endpoint 1/Host OUT Channel 1: DFIFO Write Access Device OUT Endpoint 1/Host IN Channel 1: DFIFO Read Access	0x2000–0x2FFC	w r
...	...	...
Device IN Endpoint x(1)/Host OUT Channel x(1): DFIFO Write Access Device OUT Endpoint x(1)/Host IN Channel x(1): DFIFO Read Access	0xX000–0xXFFC	w r

1. x=5 у устройства и x=11 у хоста.

Карта CSR питания и тактов

Это отдельный регистр, доступный в режимах хоста и устройства.

Таблица 212. Регистр управления питанием и тактами

Название	Имя	Смещение адреса: 0xE00–0xFFF
Power and clock gating control register	OTG_FS_PCGCCTL	0xE00-0xE04
Reserved		0xE05–0xFFF

35.12.2. Общие регистры OTG_HS

Доступны в режимах хоста и устройства, при переключении режимов переписывать не надо.

Регистр управления и состояния (OTG_HS_GOTGCTL)

Смещение адреса: 0x000

По сбросу: 0x0001 0000

31–20	19	18	17	16	15–12	11	10	9	8	7–2	1	0
Reserved	BSVLD	ASVLD	DBCT	CIDSTS	Reserved	DHNPEN	HSHNPEN	HNPRQ	HNGSCS	Reserved	SRQ	SRQSCS
	r	r	r	r		rw	rw	rw	r		rw	r

— Биты 31:20 Резерв, не трогать.

— Бит 19 **BSVLD:** B-сессия в норме

0: B-сессия сбойная.

1: B-сессия достойная.

В режиме OTG можно использовать для определения факта подключения устройства.

NB: Доступен только в режиме устройства.

— Бит 18 **ASVLD:** A-сессия в норме

0: A-сессия сбойная

1: A-сессия достойная

NB: Доступен только в режиме хоста

— Бит 17 **DBCT:** Длинное/короткое время антидребезга обнаруженного подключения

0: Длинное, физическое подключение (100 ms + 2.5 µs)

1: Короткое, программное подключение (2.5 µs)

NB: Доступен только в режиме хоста

— Бит 16 **CIDSTS**: Состояние ID подключения

0: Контроллер OTG_FS в режиме A-устройства

1: Контроллер OTG_FS в режиме B-устройства

NB: Доступен в обоих режимах.

— Биты 15:12 Резерв, не трогать.

— Бит 11 **DHNPEN**: Разрешение HNP устройства

Успешно принята команда **SetFeature.SetHNPEnable** от хоста USB.

0: HNP не разрешён

1: HNP разрешён

NB: Доступен только в режиме устройства.

— Бит 10 **HSHNPEN**: Хост установил HNP устройству

Программно ставится после разрешения HNP на устройстве (командой **SetFeature.SetHNPEnable**).

0: Неудача

1: Успех

NB: Доступен только в режиме хоста

— Бит 9 **HNPRQ**: Выдача запроса HNP

Ставится программно. Программно можно снять записью 0 после установки бита HNSSCHG в регистре OTG_FS_GOTGINT. Ядро снимает этот бит после очистки бита HNSSCHG.

0: Запроса HNP нет

1: Запрос HNP

NB: Доступен только в режиме устройства.

— Бит 8 **HNGSCS**: Успешная беседа с Хостом

Ставится ядром при успешной беседе с хостом. Снимается ядром при стоящем запросе HNP (HNPRQ).

0: Не договорились

1: Успешно побеседовали

NB: Доступен только в режиме устройства.

— Биты 7:2 Резерв, не трогать.

— Бит 1 **SRQ**: Программный запрос сессии на USB

Ставится программой. Программно может сниматься записью 0 при изменении состояния беседы с хостом (стоит бит HNSSCHG в регистре OTG_FS_GOTGINT). Ядро снимает этот бит при чистом бите HNSSCHG. При использовании полноскоростного USB 1.1 перед запросом сессии надо дождаться разрядки V_{BUS} до 0.2 V после снятия бита BSVLD в OTG_FS_GOTGCTL. это время приведено в документации производителя.

0: Ничего не просили

1: Просим сессию

NB: Доступен только в режиме устройства.

— Бит 0 **SRQSCS**: Успешный запрос сессии

Ставится ядром.

0: Сбойный

1: Успех

NB: Доступен только в режиме устройства.

Регистр прерываний (OTG_HS_GOTGINT)

Смещение адреса: 0x004

По сбросу: 0x0000 0000

Флаги ставятся при прерываниях OTG, снимаются записью 1 в нужный бит.

31 … 20	19	18	17	16 … 10	9	8	7 … 3	2	1 … 0
Reserved	DBCDNE	ADTOCHG	HNGDET	Reserved	HNSSCHG	SRSSCHG	Reserved	SEDET	Res.
	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1		rc_w1	

— Биты 31:20 Резерв, не трогать.

— Бит 19 **DBCDNE**: Антидребезг завершён

Ставится ядром после подключения устройства. Теперь можно выдавать сброс USB. Бит работает только при стоящем бите HNPCAP или SRPCAP в регистре OTG_FS_GUSBCFG.

NB: Доступен только в режиме хоста

— Бит 18 **ADTOCHG**: Таймаут A-устройства

Ставится ядром после таймаута ожидания подключения B-устройства.

NB: Доступен в обоих режимах.

— Бит 17 **HNGDET**: Начало беседы хоста

Ставится ядром после обнаружения запроса беседы с хостом на USB.

NB: Доступен в обоих режимах.

— Биты 16:10 Резерв, не трогать.

— Бит 9 **HNSSCHG**: Состояние беседы хоста изменилось

Ставится ядром после изменения состояния бита HNGSCS в регистре OTG_FS_GOTGCTL.

NB: Доступен в обоих режимах.

— Бит 8 **SRSSCHG**: Состояние запроса сессии изменилось

Ставится ядром после изменения состояния бита SRQSCS в регистре OTG_FS_GOTGCTL.

NB: Доступен в обоих режимах.

— Биты 7:3 Резерв, не трогать.

— Бит 2 **SEDET**: Конец сессии

Ставится ядром после падения $V_{BUS} < 0.8$ V. Конец сессии B-устройства.

— Биты 1:0 Резерв, не трогать.

Регистр конфигурации AHB (OTG_HS_GAHBCFG)

Смещение адреса: 0x008

По сбросу: 0x0000 0000

Пишется после включения питания или смены режима до запуска любой передачи по AHB или USB. Параметры относятся в основном к AHB. После начальной записи изменять его нельзя.

31 … 9	8	7	6	5	4	3	2	1	0
Reserved	PTXFELVL	TXFELVL	Reserved	DMAEN	HBSTLEN				GINT
	rw	rw		rw	rw	rw	rw	rw	rw

— Биты 31:9 Резерв, не трогать.

— Бит 8 **PTXFELVL**: Уровень пустоты Периодического TxFIFO

Выдача прерывания пустого периодического TxFIFO (бит PTXFE в OTG_FS_GINTSTS) при:

0: Полупустом периодическом TxFIFO
1: Совсем пустом периодическом TxFIFO
NB: Доступен только в режиме хоста

— Бит 7 **TXFELVL**: Уровень пустоты TxFIFO
В режиме устройства: выдача прерывания пустого TxFIFO точек IN (TXFE в OTG_HS_DIEPINTx) при:
0: Полупустом TxFIFO
1: Совсем пустом TxFIFO
NB: Доступен только в режиме устройства

— Бит 6 Резерв, не трогать.

— Бит 5 **DMAEN**: Включение DMA
0: Ядро в режиме ведомого
1: Ядро в режиме DMA

— Биты 4:1 **HBSTLEN**: Длина/тип пакета
0000 Одинарный
0001 INCR
0011 INCR4
0101 INCR8
0111 INCR16
Иное: Резерв

— Бит 0 **GINT**: Маска Общего прерывания
Маска собранных воедино источников прерываний. Регистры состояния прерываний всё равно изменяются независимо от этого бита.
0: Прерывать нельзя.
1: Прерывать можно.
NB: Доступен в обоих режимах.

Регистр конфигурации USB (OTG_HS_GUSBCFG)

Смещение адреса: 0x00C

По сбросу: 0x0000 1440

Пишется после включения питания или смены режима до запуска любой передачи по AHB или USB. Параметры относятся к USB и USB-PHY. После начальной записи изменять его нельзя.

31	30	29	28 27 26	25	24	23	22	21	20	19	18	17	16	15	14	13 12 11 10	9	8	7	6	5 4 3	2 1 0
CTXPKT	FDMOD	FHMOD	Reserved	ULPIIPD	PTCI	PCCI	TSDPS	ULPIEVBUSI	ULPIEVBUSD	ULPICSM	ULPIAR	ULPIFSLS	Reserved	PHYLPCS	Reserved	TRDT	HNPCAP	SRPCAP	Reserved	PHYSEL	Reserved	TOCAL
rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw		rw		rw	rw	rw		rw		rw

— Бит 31 **CTXPKT**: Нарушенный пакет Tx
Только для отладки, не ставьте этот бит ни за что и никогда.
NB: Доступен в обоих режимах.

— Бит 30 **FDMOD**: Принудительный режим устройства
Запись 1 переключает в режим устройства независимо от входной ножки ID.
0: Нормальный режим
1: Принудительный режим устройства
Изменения наступают через 25 ms после установки этого бита.
NB: Доступен в обоих режимах.

— Бит 29 **FHMOD**: Принудительный режим хоста
Запись 1 переключает в режим хоста независимо от входной ножки ID.
0: Нормальный режим
1: Принудительный режим хоста
Изменения наступают через 25 ms после установки этого бита.
NB: Доступен в обоих режимах.

— Биты 28:26 Резерв, не трогать.

— Бит 25 **ULPIIPD**: Выключение защиты ULPI
Это про встроенную защиту PHY для ULPI. Резисторы можно отключить. См. в спецификацию ULPI.

0: Защита включена

1: Защита выключена

— Бит 24 **PTCI**: Индикатор прохождения

Управляет прохождением комплементарного сигнала через внутренний компаратор V_{BUS} до использования в RX CMD. См. в спецификацию ULPI.

0: Проходит

1: Не проходит

— Бит 23 **PCCI**: Индикатор комплемента

Управляет инверсией PHY входного сигнала External Vbus Indicator для создания комплементарного выхода. См. в спецификацию ULPI.

0: PHY не инвертирует External Vbus Indicator

1: PHY инвертирует External Vbus Indicator

— Бит 22 **TSDPS**: Выбор импульса TermSel D-Line при запросе SRP.

0: Импульс utmi_txvalid (default)

1: Импульс utmi_termsel

— Бит 21 **ULPIEVBUSI**: Индикатор внешней V_{BUS} ULPI

Использование ULPI PHY внешнего компаратора перегрузки V_{BUS}.

0: PHY использует внутренний компаратор V_{BUS}

1: PHY использует внешний компаратор V_{BUS}

— Бит 20 **ULPIEVBUSD**: Выбор источника V_{BUS} ULPI PHY

0: PHY кормит V_{BUS} изнутри (default)

1: PHY кормит V_{BUS} снаружи.

— Бит 19 **ULPICSM**: ULPI ClockSuspendM

Ставит бит ClockSuspendM в регистре управления ULPI PHY. Только для serial и car kit режимов.

0: PHY выключает внутренние такты на время останова

1: PHY не выключает внутренние такты на время останова

— Бит 18 **ULPIAR**: Само восстановление ULPI

Ставит бит AutoResume в регистре управления ULPI PHY.

0: PHY не использует фичу AutoResume

1: PHY использует AutoResume

— Бит 17 **ULPIFSLS**: Выбор интерфейса ULPI FS/LS

Только для передатчика FS в ULPI PHY.

0: ULPI

1: ULPI FS/LS

— Бит 16 Резерв, не трогать.

— Бит 15 **PHYLPCS**: Выбор экономных тактов PHY

Это 480 MHz или 48 MHz. В режимах FS и LS это обычно 48 MHz.

0: 480 MHz внутренний PLL

1: 48 MHz внешние

В режиме 480 MHz UTMI работает на 60 или 30 MHz в зависимости от 8- или 16-бит данных. В режиме 48 MHz UTMI работает на 48 MHz в FS и LS.

— Бит 16 Резерв, не трогать.

— Биты 13:10 **TRDT**: Время обратной связи USB в тактах PHY

Они ставятся в соответствии с Таблицей 213, в зависимости от частоты AHB.

NB: Доступен только в режиме устройства.

— Бит 9 **HNPCAP**: Разрешение HNP

0: HNP запрещён.

1: HNP Разрешён.

NB: Доступен в обоих режимах.

— Бит 8 **SRPCAP**: Разрешение SRP

0: SRP запрещён.

1: SRP Разрешён.

NB: Доступен в обоих режимах.

— Бит 7 Резерв, не трогать.

— Бит 6 **PHYSEL**: Выбор полноскоростного трансивера

Всегда 1, только запись.

— Биты 5:3 Резерв, не трогать.

— Биты 2:0 **TOCAL**: Калибровка таймаута FS

Число тактов PHY, добавляемых к стандартной задержке (от 16 до 18 времён бита) между передачами пакетов для компенсации работы PHY от разных производителей. Определяется на основе скорости переучёта. На каждый такт PHY добавляется 0.25 времени бита.

Таблица 213. Значения TRDT

Диапазон частот AHB (MHz)		Мин. значение TRDT
Min.	Max	
30	—	0x9

Регистр сброса (OTG_HS_GRSTCTL)

Смещение адреса: 0x010

По сбросу: 0x8000 0000

31	30	29–11	10–6	5	4	3	2	1	0
AHBIDL	DMAREQ	Reserved	TXFNUM	TXFFLSH	RXFFLSH	Reserved	FCRST	HSRST	CSRST
r	r		rw	rs	rs		rs	rs	rs

— Бит 31 **AHBIDL**: Простой ведущего AHB

Машина состояний ведущего AHB в Простое.

NB: Доступен в обоих режимах.

— Бит 30 **DMAREQ**: Запрос DMA активен (для отладки)

— Биты 29:11 Резерв, не трогать.

— Биты 10:6 **TXFNUM**: Номер сливаемого TxFIFO

FIFO сливается установкой TxFIFO Flush. Пока бит TxFIFO Flush стоит, изменять это поле нельзя.

00000:

– Не-периодический TxFIFO в режиме хоста

– TxFIFO 0 в режиме устройства

00001:

– Периодический TxFIFO в режиме хоста

– TxFIFO 1 в режиме устройства

00010: TxFIFO 2 в режиме устройства

...

00101: TxFIFO 15 в режиме устройства

10000: Все передающие FIFO в обоих режимах.

NB: Доступен в обоих режимах.

— Бит 5 **TXFFLSH**: Слив TxFIFO

Команда на слив одного или всех FIFO, но не в середине передач с FIFO.

Отсутствие передач проверяется так:

Для чтения из TxFIFO—Было прерывание "NAK в действии"

Для записи в TxFIFO—Стоит бит AHBIDL в регистре OTG_FS_GRSTCTL.

NB: Доступен в обоих режимах.

— Бит 4 **RXFFLSH**: Слив RxFIFO

Команда на слив всего RxFIFO, но не в середине передач с RxFIFO.

Перед записью в этот бит надо убедиться, что RxFIFO ни читается ни пишется.

Перед запуском других операций надо дождаться снятия этого бита. Это требует 8 тактов (самый медленный из PHY и AHB).

NB: Доступен в обоих режимах.

— Бит 3 Резерв, не трогать.

— Бит 2 **FCRST**: Сброс счётчика фреймов хоста

Следующий посылаемый SOF будет иметь номер фрейма 0.

NB: Доступен только в режиме хоста.

— Бит 1 **HSRST**: Программный сброс HCLK

Слив управляющей логики конвейеров домена тактов AHB после завершения передач AHB.

Управляющие биты CSR машин состояния домена AHB снимаются.

Маски прерываний машин состояния домена AHB снимаются.

Биты состояния прерываний машин состояния домена AHB не снимаются.

Очистка может занять несколько тактов.

NB: Доступен в обоих режимах.

— Бит 0 **CSRST**: Программный сброс ядра

Сбрасывает домены HCLK и PCLK:

Чистит прерывания и все биты регистров CSR за исключением битов:

- RSTPDMODL в OTG_FS_PCGCCTL
- GAYEHCLK в OTG_FS_PCGCCTL
- PWRCLMP в OTG_FS_PCGCCTL
- STPPCLK в OTG_FS_PCGCCTL
- FSLSPCS в OTG_FS_HCFG
- DSPD в OTG_FS_DCFG

Все машины состояния модуля (кроме блока ведомого AHB) сбрасываются в Простой, передающие и приёмный FIFO сливаются.

Передачи ведущего AHB завершаются после конца последней фазы данных. Передачи по USB завершаются немедленно.

Бит можно писать когда угодно. Снимается сам через несколько тактов. После снятия бита перед доступом к PHY надо подождать не меньше 3 тактов PHY (задержка синхронизации) и дождаться установки бита 31 в этом регистре (Простой ведущего AHB).

Обычно программный сброс используется при разработке программ и динамическом изменении битов выбора PHY, после которого надо сбросить и домен PHY.

NB: Доступен в обоих режимах.

Регистр прерываний ядра (OTG_HS_GINTSTS)

Смещение адреса: 0x014

По сбросу: 0x0400 0020

Некоторые биты работают только в режиме хоста, иные только в режиме устройства. В регистре отмечен текущий режим. Изменяемые биты состояния прерванный снимаются записью 1.

Биты состояния прерываний FIFO только читаются, они изменяются автоматически при обращении к FIFO.

Во избежание излишних прерываний чистить регистр `OTG_FS_GINTSTS` при инициализации надо до снятия маски запрета прерываний.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WKUINT	SRQINT	DISCINT	CIDSCHG	Reserved	PTXFE	HCINT	HPRTINT	Reserved		IPXFR/INCOMPISOOUT	ISOIXFR	OEPINT	IEPINT	Reserved		EOPF	ISOODRP	ENUMDNE	USBRST	USBSUSP	ESUSP	Reserved		GONAKEFF	GINAKEFF	NPTXFE	RXFLVL	SOF	OTGINT	MMIS	CMOD
rc_w1					r	r	r	Res.		rc_w1		r	r			rc_w1								r	r	r	r	rc_w1	r	rc_w1	r

— Бит 31 **WKUPINT**: Прерывание удалённой побудки

В режиме устройства ставится при появлении на шине USB события восстановления.

В режиме хоста ставится при появлении на шине USB события удалённой побудки.

NB: Доступен в обоих режимах.

— Бит 30 **SRQINT**: Прерывание запроса/новой сессии

В режиме хоста ставится при появлении на шине USB запроса сессии от устройства.

В режиме устройства ставится при появлении рабочего уровня V_{BUS} для B-устройства.

NB: Доступен в обоих режимах.

— Бит 29 **DISCINT**: Прерывание отключения (вытыкания) внешнего устройства

NB: Доступен в обоих режимах.

— Бит 28 **CIDSCHG**: Состояние линии ID разъёма изменилось

NB: Доступен в обоих режимах.

— Бит 27 Резерв, не трогать.

— Бит 26 **PTXFE**: Периодический TxFIFO пуст

Ставится если в периодическом передающем FIFO есть куда писать. Полу- или полное опустение задаётся битом PTXFELVL в регистре OTG_FS_GAHBCFG.

NB: Доступен только в режиме хоста.

— Бит 25 **HCINT**: Прерывание каналов хоста

В режиме хоста ставится ядром при наличии запроса какого-либо канала хоста. Номер запросившего канала читается из регистра OTG_FS_HAINT, причина запроса читается из регистра OTG_FS_HCINTx. Снимается очисткой бита запроса в регистре OTG_FS_HCINTx.

NB: Доступен только в режиме хоста.

— Бит 24 **HPRTINT**: Прерывание изменения состояния порта Хоста

Событие запроса читается из регистра OTG_FS_HPRT. Снимается очисткой бита запроса в регистре OTG_FS_HPRT.

NB: Доступен только в режиме хоста.

— Биты 23:22 Резерв, не трогать.

— Бит 21 **IPXFR**: Незавершённая периодическая передача

В режиме хоста ядро извещает о незавершённой периодической передаче для текущего фрейма.

INCOMPISOOUT: Незавершённая изохронная передача OUT

В режиме устройства ядро извещает о незавершённой изохронной передаче OUT для текущего фрейма. Выставляется вместе с битом конца периодического фрейма (EOPF) в этом регистре.

— Бит 20 **IISOIXFR**: Незавершённая изохронная передача IN

В режиме устройства ядро извещает о незавершённой изохронной передаче IN для текущего фрейма. Выставляется вместе с битом конца периодического фрейма (EOPF) в этом регистре.

NB: Доступен только в режиме устройства.

— Бит 19 **OEPINT**: Прерывание конечной точки OUT

В режиме устройства ставится ядром при наличии запроса какой-либо конечной точки OUT. Номер запросившей точки читается из регистра OTG_FS_DAINT, причина запроса читается из регистра OTG_FS_DOEPINTx. Снимается очисткой бита запроса в регистре OTG_FS_DOEPINTx.

NB: Доступен только в режиме устройства.

— Бит 18 **IEPINT**: Прерывание конечной точки IN

В режиме устройства ставится ядром при наличии запроса какой-либо конечной точки IN. Номер запросившей точки читается из регистра OTG_FS_DAINT, причина запроса читается из регистра OTG_FS_DIEPINTx. Снимается очисткой бита запроса в регистре OTG_FS_DIEPINTx.

NB: Доступен только в режиме устройства.

— Биты 17:16 Резерв, не трогать.

— Бит 15 **EOPF**: Прерывание конца периодического фрейма

Возникает при истечении интервала периодического фрейма (поле PFIVL в OTG_FS_DCFG) в текущем фрейме.

NB: Доступен только в режиме устройства.

— Бит 14 **ISOODRP**: Прерывание выброса изохронного пакета OUT

Ставится ядром при нехватке места в RxFIFO для изохронного пакета OUT.

NB: Доступен только в режиме устройства.

— Бит 13 **ENUMDNE**: Переучёт завершён

После этого в регистре OTG_FS_DSTS лежит скорость переучёта.

NB: Доступен только в режиме устройства.

— Бит 12 **USBRST**: Сигнал сброса на шине USB

NB: Доступен только в режиме устройства.

— Бит 11 **USBSUSP**: Сигнал Приостановки на шине USB

NB: Доступен только в режиме устройства.

— Бит 10 **ESUSP**: Ранняя приостановка

На шине USB не было активности в течении 3 ms.

NB: Доступен только в режиме устройства.

— Биты 9:8 Резерв, не трогать.

— Бит 7 **GONAKEFF**: Глобальный OUT NAK работает

Бит SGONAK в регистре OTG_FS_DCTL сработал. Очищается установкой бита CGONAK в регистре OTG_FS_DCTL.

NB: Доступен только в режиме устройства.

— Бит 6 **GINAKEFF**: Глобальный NAK не-периодических IN работает

Бит SGINAK в регистре OTG_FS_DCTL сработал. Очищается установкой бита CGINAK в регистре OTG_FS_DCTL.

Это прерывание не означает, что по шине USB будет посылаться NAK, бит STALL старше.

NB: Доступен только в режиме устройства.

— Бит 5 **NPTXFE**: Не-периодический TxFIFO пуст

Ставится если в не-периодическом передающем FIFO есть куда писать. Полу- или полное опустение задаётся битом TXFELVL в регистре OTG_FS_GAHBCFG.

NB: Доступен только в режиме устройства.

— Бит 4 **RXFLVL**: RxFIFO не пуст

Из RxFIFO можно прочесть хоть один пакет.

NB: Доступен в обоих режимах.

— Бит 3 **SOF**: Старт фрейма

В режиме хоста ядро извещает что на шину USB послан SOF (FS) или Keep-Alive (LS). Снимается записью 1 в этот бит.

В режиме устройства ядро извещает, что на USB появился токен. Номер текущего фрейма читают в регистре Состояния устройства. Прерывание проявляется только при работе с FS.

NB: Доступен в обоих режимах.

— Бит 2 **OTGINT**: Прерывание OTG

Ядро извещает о событии протокола OTG. Само событие выявляют по регистру OTG_FS_GOTGINT. Бит снимается очисткой бита события в регистре OTG_FS_GOTGIN.

NB: Доступен в обоих режимах.

— Бит 1 **MMIS**: Прерывание несовпадения режима

Ядро ставит этот бит при попытке доступа:

- К регистрам режима хоста из режима устройства
- К регистрам режима устройства из режима хоста

NB: Доступен в обоих режимах.

— Бит 0 **CMOD**: Текущий режим

0: Режим устройства

1: Режим хоста

NB: Доступен в обоих режимах.

Регистр маски прерываний (OTG_HS_GINTMSK)

Смещение адреса: 0x018

По сбросу: 0x0000 0000

Изменение регистра маски на регистр запросов прерывание (OTG_FS_GINTSTS) не влияет.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WUIM	SRQIM	DISCINT	CIDSCHGM	Reserved	PTXFEM	HCIM	PRTIM	Reserved	FSUSPM	IPXFRM/IISOOXFRM	IISOIXFRM	OEPINT	IEPINT	Reserved		EOPFM	ISOODRPM	ENUMDNEM	USBRST	USBSUSPM	ESUSPM	Reserved		GONAKEFFM	GINAKEFFM	NPTXFEM	RXFLVLM	SOFM	OTGINT	MMISM	Reserved
rw	rw	rw	rw		rw	rw	rw		rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	

Состояние битов:

0: Нельзя

1: Можно

— Бит 31 **WUIM**: Маска прерывания побудки

NB: Доступен в обоих режимах.

— Бит 30 **SRQIM**: Маска прерывания запроса/новой сессии

NB: Доступен в обоих режимах.

— Бит 29 **DISCINT**: Маска прерывания отключения устройства

NB: Доступен только в режиме устройства.

— Бит 28 **CIDSCHGM**: Маска прерывания изменения ID
NB: Доступен в обоих режимах.
— Бит 27 Резерв, не трогать.
— Бит 26 **PTXFEM**: Маска прерывания пустого периодического TxFIFO
NB: Доступен только в режиме хоста.
— Бит 25 **HCIM**: Маска прерывания каналов хоста
NB: Доступен только в режиме хоста.
— Бит 24 **PRTIM**: Маска прерывания порта хоста
NB: Доступен только в режиме хоста.
— Бит 23 Резерв, не трогать.
— Бит 22 **FSUSPM**: Маска задержанной выборки данных
NB: Доступен только в режиме устройства.
— Бит 21 **IPXFRM**: Маска прерывания незавершённой периодической передачи
NB: Доступен только в режиме хоста.
IISOOXFRM: Маска прерывания незавершённой изохронной передачи OUT
NB: Доступен только в режиме устройства.
— Бит 20 **IISOIXFRM**: Маска прерывания незавершённой изохронной передачи IN
NB: Доступен только в режиме устройства.
— Бит 19 **OEPINT:** Маска прерывания конечной точки OUT
NB: Доступен только в режиме устройства.
— Бит 18 **IEPINT**: Маска прерывания конечной точки IN
NB: Доступен только в режиме устройства.
— Биты 17:16 Резерв, не трогать.
— Бит 15 **EOPFM**: Маска прерывания конца периодического фрейма
NB: Доступен только в режиме устройства.
— Бит 14 **ISOODRPM**: Маска прерывания выброса изохронного пакета OUT
NB: Доступен только в режиме устройства.
— Бит 13 **ENUMDNEM**: Маска прерывания конца переучёта
NB: Доступен только в режиме устройства.
— Бит 12 **USBRST**: Маска прерывания сброса USB
NB: Доступен только в режиме устройства.
— Бит 11 **USBSUSPM**: Маска прерывания приостановки USB
NB: Доступен только в режиме устройства.
— Бит 10 **ESUSPM**: Маска прерывания ранней приостановки
NB: Доступен только в режиме устройства.
— Биты 9:8 Резерв, не трогать.
— Бит 7 **GONAKEFFM**: Маска прерывания Global OUT NAK effective
NB: Доступен только в режиме устройства.
— Бит 6 **GINAKEFFM**: Маска прерывания Global non-periodic IN NAK effective
NB: Доступен только в режиме устройства.
— Бит 5 **NPTXFEM**: Маска прерывания пустого не-периодического TxFIFO
NB: Доступен только в режиме хоста.
— Бит 4 **RXFLVLM**: Маска прерывания непустого RxFIFO
NB: Доступен в обоих режимах.
— Бит 3 **SOFM**: Маска прерывания старта фрейма
NB: Доступен в обоих режимах.
— Бит 2 **OTGINT**: Маска прерывания OTG
NB: Доступен в обоих режимах.
— Бит 1 **MMISM**: Маска прерывания несовпадения режима
NB: Доступен в обоих режимах.
— Бит 0 Резерв, не трогать.

Регистры Состояния приёма отладки/Состояния приёма и извлечения (OTG_HS_GRXSTSR/OTG_HS_GRXSTSP)

Смещение адреса регистра состояния приёма: 0x01C
Смещение адреса регистра извлечения: 0x020
По сбросу: 0x0000 0000

Чтение Состояния приёма отладки возвращает верхушку приёмного FIFO. Чтение Состояния приёма и извлечения дополнительно извлекает данные верхней строки RxFIFO.

The receive status contents must be interpreted differently in host and device modes. The core ignores the receive status pop/read when the receive FIFO is empty and returns a value of 0x0000 0000. The application must only pop the Receive Status FIFO when the Receive FIFO non-empty bit of the Core interrupt register (RXFLVL bit in OTG_HS_GINTSTS) is asserted.

Режим хоста

31–21	20–17	16–15	14–4	3–0
Reserved	PKTSTS	DPID	BCNT	CHNUM
	r	r	r	r

— Биты 31:21 Резерв, не трогать.

— Биты 20:17 **PKTSTS**: Статус принятого пакета

0010: Принят пакет данных IN

0011: Передача IN завершена (с прерыванием)

0101: Ошибка переключения данных (с прерыванием)

0111: Канал остановлен (с прерыванием)

Иные: Резерв

— Биты 16:15 **DPID**: PID данных принятого пакета

00: DATA0

10: DATA1

01: DATA2

11: MDATA

— Биты 14:4 **BCNT**: Счётчик байтов принятого пакета IN

— Биты 3:0 **CHNUM**: Номер канала текущего принятого пакета

Режим устройства

31–25	24–21	20–17	16–15	14–4	3–0
Reserved	FRMNUM	PKTSTS	DPID	BCNT	EPNUM
	r	r	r	r	r

— Биты 31:25 Резерв, не трогать.

— Биты 24:21 **FRMNUM**: Номер фрейма

Младшие 4 бита номера фрейма принятого пакета. Только при поддержке изохронных точек OUT.

— Биты 20:17 **PKTSTS**: Статус принятого пакета

0001: Глобальный OUT NAK (с прерыванием)

0010: Принят пакет данных OUT

0011: Передача OUT завершена (с прерыванием)

0100: Передача SETUP завершена (с прерыванием)

0110: Принят пакет данных SETUP

Others: Reserved

— Биты 16:15 **DPID**: PID данных принятого пакета OUT

00: DATA0

10: DATA1

01: DATA2

11: MDATA

— Биты 14:4 **BCNT**: Счётчик байтов принятого пакета

— Биты 3:0 **EPNUM**: Номер конечной точки текущего принятого пакета

Регистр размера приёмного FIFO (OTG_HS_GRXFSIZ)

Смещение адреса: 0x024

По сбросу: 0x0000 0400

31–16	15–0
Reserved	RXFD
	r/rw

— Биты 31:16 Резерв, не трогать.

— Биты 15:0 **RXFD**: Глубина RxFIFO в 32-бит словах

Минимальное значение 16

Максимальное значение 1024

По сбросу питания пишется значение наибольшей глубины Rx FIFO данных.

Регистр размера передающего не-периодического FIFO хоста (OTG_HS_HNPTXFSIZ)
Размер передающего FIFO точки 0 (OTG_HS_DIEPTXF0)

Смещение адреса: 0x028

По сбросу: 0x0000 0200

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NPTXFD/TX0FD																NPTXFSA/TX0FSA															
r/rw																r/rw															

Режим хоста

— Биты 31:16 **NPTXFD**: Глубина не-периодического TxFIFO в 32-бит словах.

Минимальное значение 16

Максимальное значение 1024

— Биты 15:0 **NPTXFSA**: Адрес начала RAM не-периодических передач

Режим устройства

— Биты 31:16 **TX0FD**: Глубина TxFIFO конечной точки 0 в 32-бит словах.

Минимальное значение 16

Максимальное значение 256

— Биты 15:0 **TX0FSA**: Адрес начала RAM передач конечной точки 0

Регистр не-периодического передающего FIFO/состояния очереди (OTG_HS_HNPTXSTS)

Смещение адреса: 0x02C

По сбросу: 0x0008 0400

Содержит информацию о свободном месте в TxFIFO.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	NPTXQTOP							NPTQXSAV								NPTXFSAV															
	r							r								r															

— Бит 31 Резерв, не трогать.

— Биты 30:24 **NPTXQTOP**: Верх очереди не-периодических запросов передачи

Биты 30:27: Номер Канала/Конечной точки

Биты 26:25: Статус передачи

– 00: Токен IN/OUT
– 01: Пакет нулевой длины (устройство IN/хост OUT)
– 11: Команда остановки канала

Бит 24: Окончание (последняя строка этого Канала/Конечной точки)

— Биты 23:16 **NPTQXSAV**: Доступное место в очереди не-периодических запросов передачи

В режиме хоста очередь хранит запросы и IN и OUT, в режиме устройства только запросы. IN.

00: Очередь запросов не-периодических передач заполнена

01: 1 свободная строка

10: 2 свободных строки

bxn: *n* свободных строк ($0 \le n \le 8$)

Иные: Резерв

— Биты 15:0 **NPTXFSAV**: Доступное место очереди TxFIFO в 32-бит словах

00: Непериодический TxFIFO полон

01: 1 свободное слово

10: 2 свободных слова

0xn: *n* свободных слов ($0 \le n \le 1024$)

Иные: Резерв

Общий регистр конфигурации ядра (OTG_HS_GCCFG)

Смещение адреса: 0x038

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										NOVBUSSENS	SOFOUTEN	VBUSBSEN	VBUSASEN	I2CPADEN	.PWRDWN	Reserved															
										rw	rw	rw	rw	rw	rw																

— Биты 31:22 Резерв, не трогать.

— Бит 21 **NOVBUSSENS**: Выключение контроля V_{BUS}

В режимах только хост и только устройства.

0: Можно.

1: Не надо.

— Бит 20 **SOFOUTEN**: Разрешение выдачи SOF на PAD

0: Нельзя

1: Можно

— Бит 19 **VBUSBSEN:** Разрешение проверки V_{BUS} устройства “B”

0: Нельзя

1: Можно

— Бит 18 **VBUSASEN:** Разрешение проверки V_{BUS} устройства “A”

0: Нельзя

1: Можно

— Бит 17 **I2CPADEN**: Enable I^2C bus connection for the external I^2C PHY interface.

0: Нельзя.

1: Можно.

— Бит 16 **PWRDWN**: Выключение питания трансивера

0: Питание выключено

1: Трансивер включён

— Биты 15:0 Резерв, не трогать.

Регистр ID ядра (OTG_HS_CID)

Смещение адреса: 0x03C

По сбросу: 0x0000 1100

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRODUCT_ID																															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:21 **PRODUCT_ID**: Программируемый ID продукта

Регистр размера периодического передающего FIFO хоста (OTG_HS_HPTXFSIZ)

Смещение адреса: 0x100

По сбросу: 0x0200 0800

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PTXFSIZ																PTXSA															
r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw

— Биты 31:16 **PTXFD**: Глубина периодического TxFIFO хоста в 32-бит словах

Минимальное значение 16

— Биты 15:0 **PTXSA**: Адрес начала периодического TxFIFO хоста

По сбросу питания содержит сумму наибольших глубин Rx FIFO данных и не-периодического Tx FIFO данных.

Регистр размера передающего FIFO конечной точки (OTG_HS_DIEPTXFx) (x = 1..7, где x это номер FIFO)

Смещение адреса: 0x104 + (FIFO_number – 1) × 0x04 × (x-1)

По сбросу: 0x0200 0400

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INEPTXFD																INEPTXSA															
r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw	r/rw

— Биты 31:16 **INEPTXFD**: Глубина TxFIFO конечной точки N в 32-бит словах
Минимальное значение 16, максимальное 512.
По сбросу питания это значение определено как наибольшая глубина FIFO точки IN.

— Биты 15:0 **INEPTXSA**: Адрес начала FIFOx передачи конечной точки IN
Должен быть выравнен на границу 32-бит слова.

35.12.3. Регистры режима хоста

В режиме устройства их содержимое не определено.

Регистр конфигурации хоста (OTG_HS_HCFG)

Смещение адреса: 0x400

По сбросу: 0x0000 0000

Конфигурация ядра по сбросу питания. После инициализации хоста изменять нельзя.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																													FSLSS	FSLSPCS	
																													r	rw	rw

— Биты 31:16 Резерв, не трогать.

— Бит 2 **FSLSS**: Поддержка только FS и LS
Определяет скорость переучёта. Пишется единожды.
1: Только FS/LS, даже для устройств с поддержкой HS (только чтение)

— Биты 1:0 **FSLSPCS**: Выбор тактов PHY FS/LS

В режиме FS

01: PHY на частоте 48 MHz
Иные: Резерв

В режиме LS

00: Резерв
01: PHY на частоте 48 MHz
10: PHY на частоте 6 MHz
11: Резерв

NB: Ставить надо после события подключения устройства в соответствии с его частотой (после изменения бита надо делать программный сброс).

Регистр интервала фреймов хоста (OTG_HS_HFIR)

Смещение адреса: 0x404

По сбросу: 0x0000 EA60

Значение интервала фреймов для текущей скорости переучёта.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																FRIVL															
																rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:16 Резерв, не трогать.

— Биты 15:0 **FRIVL**: Интервал фреймов
Программное определение интервала между двумя последовательными токенами SOF (FS) или Keep-Alive (LS) в тактах PHY. Писать можно только после установки бита PENA в регистре OTG_FS_HPRT. Если не задано, то ядро вычисляет его по значению поля FSLSPCS в регистре OTG_FS_HCFG. Писать надо единожды при начальной конфигурации.
1 ms × (частоту тактов PHY)

Регистр номера/остатка времени фрейма хоста (OTG_HS_HFNUM)

Смещение адреса: 0x408

По сбросу: 0x0000 3FFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FTREM																FRNUM															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

— Биты 31:16 **FTREM**: Остаток времени фрейма в тактах PHY
Декрементируется по каждому такту PHY. Перегружается из регистра интервала при достижении 0 и передаче нового SOF.

— Биты 15:0 **FRNUM**: Номер фрейма
Инкрементируется по новому SOF, обнуляется при достижении 0x3FFF.

Регистр состояния периодических передающего FIFO/очереди (OTG_HS_HPTXSTS)

Смещение адреса: 0x410

По сбросу: 0x0008 0100

Только чтение. Информация о свободном месте в TxFIFO и очереди периодических передач.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PTXQTOP								PTXQSAV								PTXFSAVL															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:24 **PTXQTOP**: Верх очереди запросов периодических передач
Используется при отладке.
Бит 31: Чётный/Нечётный фрейм
0: Чётный
1: Нечётный
Биты 30:27: Номер Канала/Конечной точки
Биты 26:25: Тип
00: IN/OUT
01: Пакет нулевой длины
11: Команда выключения канала
Бит 24: Последняя строка этого Канала/Конечной точки

— Биты 23:16 **PTXQSAV**: Свободное место в очереди периодических передач
Очередь содержит и IN и OUT запросы.
00: Очередь полна
01: 1 свободная строка
10: 2 свободных строки
bxn: n свободных строк (0 ≤ n ≤ 8)
Others: Резерв

— Биты 15:0 **PTXFSAVL**: Свободное место в TxFIFO периодических передач в 32-би словах
0000: TxFIFO полон
0001: 1 свободное слово
0010: 2 свободных слова
bxn: n свободных слов (0 ≤ n ≤ PTXFD)
Others: Резерв

Регистр всех прерываний хоста (OTG_HS_HAINT)

Смещение адреса: 0x414

По сбросу: 0x0000 0000

Отражает запросы прерываний каналов. Прерывание вызывается по общему биту `HCINT` в регистре `OTG_FS_GINTSTS` при разрешении в регистре маски всех каналов. Биты ставятся и снимаются в соответствии с битами регистров прерывания соответствующего канала `x`.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																HAINT															
																r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

— Биты 31:24 Резерв, не трогать.

— Биты 31:24 **HAINT**: Прерывания каналов
По одному биту на канал: бит 0 для канала 0, бит 15 для канала 15

Регистр маски всех прерываний хоста (OTG_HS_HAINTMSK)

Смещение адреса: 0x418

По сбросу: 0x0000 0000

31–16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	HAINTM															
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:24 Резерв, не трогать.

— Биты 31:24 **HAINTM**: Маска прерывания каналов

0: Нельзя

1: Можно

По одному биту на канал: бит 0 для канала 0, бит 15 для канала 15

Регистр управления и состояния портов хоста (OTG_HS_HPRT)

Смещение адреса: 0x440

По сбросу: 0x0000 0000

Сейчас хост OTG поддерживает только один порт. Регистр содержит информацию о сбросе USB, включении, остановке, восстановлении, подключении и поверке. Прерывания вызываются по биту `HPRTINT` в регистре `OTG_HS_GINTSTS`. Чтение этого регистра снимает бит запроса прерывания. Биты rc_w1 снимаются записью 1.

31–19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	PSPD		PTCTL				PPWR	PLSTS		Reserved	PRST	PSUSP	PRES	POCCHNG	POCA	PENCHNG	PENA	PCDET	PCSTS
	r	r	rw	rw	rw	rw	rw	r	r		rw	rs	rw	rc_w1	r	rc_w1	rc_w0	rc_w1	r

— Биты 31:19 Резерв, не трогать.

— Биты 18:17 **PSPD**: Скорость подключённого к порту устройства

01: FS

10: LS

11: Резерв

— Биты 16:13 **PTCTL**: Тест порта

0000: Режим теста выключен

0001: Режим Test_J

0010: Режим Test_K

0011: Режим Test_SE0_NAK

0100: Режим Test_Packet

0101: Режим Test_Force_Enable

Others: Резерв

— Бит 12 **PPWR**: Питание порта

0: Выключено

1: Включено

— Биты 11:10 **PLSTS**: Текущее состояние линий порта

Бит 10: Логический уровень OTG_FS_FS_DP

Бит 11: Логический уровень f OTG_FS_FS_DM

— Бит 9 Резерв, не трогать.

— Бит 8 **PRST**: Сброс порта

Ставится программно, снимается программно после завершения последовательности сброса.

0: Сбросу нету

1: Сброс есть

Ставится на период не менее 10 ms перед запуском сброса порта и ещё ждём 10 ms перед снятием этого бита. В стандарте USB этого нет.

— Бит 7 **PSUSP**: Приостановка порта

Ставится программно. Ядро просто перестаёт посылать SOF. Для остановки тактов PHY надо ставить свой бит остановки. При чтении выдаёт текущее состояние порта.

Снимается ядром после установки битов WKUINT или DISCINT в регистре OTG_FS_GINTSTS.

0: Порт работает

1: Порт стоит

— Бит 6 **PRES**: Восстановление порта

Программно подаёт на порт сигнал восстановления. При программном снятии бита снимается и сигнал. При обнаружении побудки (бит WKUINT в OTG_FS_GINTSTS) ядро выдаёт сигнал и снимает его после обнаружении отключения. Чтение выдаёт текущее состояние сигнала.

0: Сигнала восстановления нет

1: Сигнал есть

— Бит 5 **POCCHNG**: Сигнал перегрузки порта по току изменился

— Бит 4 **POCA**: Перегрузка порта по току

0: Нету

1: Есть

— Бит 3 **PENCHNG**: Включение порта изменилось

— Бит 2 **PENA**: Включение порта

Порт включается ядром в конце последовательности сброса, выключается при перегрузке, отключении или программном снятии этого бита. Программно бит не ставится. Прерывания не вызывает.

0: Выключен

1: Включён

— Бит 1 **PCDET**: Порт подключён

Ставится ядром при подключении устройства, прерывание вызывается битом HPRTINT в регистре OTG_FS_GINTSTS). Снимается записью 1.

— Бит 0 **PCSTS**: Состояние подключения порта

0: Устройства нет

1: Устройство есть

Регистр характеристик канала-x (OTG_HS_HCCHARx) (x = 0..11, где x = Номер_канала)

Смещение адреса: 0x500 + (Номер_канала × 0x20)

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
rs	rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Бит 31 **CHENA**: Включение канала

Ставится программно, снимается хостом OTG.

0: Выключен

1: Включён

— Бит 30 **CHDIS**: Выключение канала

Программная установка бита останавливает приём/передачу данных не дожидаясь их завершения. Далее надо дождаться подтверждения выключения прерыванием.

— Бит 29 **ODDFRM**: Нечётный фрейм

Программно задаёт передачу чётного/нечётного фрейма периодической (изохронной или прерывающей) передачи.

0: Чётный фрейм

1: Нечётный фрейм

— Биты 28:22 **DAD**: Адрес устройства

— Биты 21:20 **MCNT**: Многосчётчик

Число передач для этого фрейма периодической конечной точки. Для не-периодических точек не используется.

00: Резерв. Результат непредсказуем

01: 1 передача

10: 2 передачи на фрейм

11: 3 передачи на фрейм

— Биты 19:18 **EPTYP**: Тип конечной точки

00: Управляющая

01: Изохронная

10: Групповая

11: Прерывающая

— Бит 17 **LSDEV**: Низкоскоростное устройство

— Бит 16 Резерв. не трогать.

— Бит 15 **EPDIR**: Направление конечной точки

0: OUT

1: IN

— Биты 14:11 **EPNUM**: Номер конечной точки

— Биты 10:0 **MPSIZ**: Максимальный размер пакета

Регистр разбиения канала-x (OTG_HS_HCSPLTx) (x = 0..11, где x = Номер_канала)

Смещение адреса: 0x504 + (Номер_канала × 0x20)

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPLITEN	Reserved														COMPLSPLT	XACTPOS		HUBADDR							PRTADDR							
rw															rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Бит 31 **SPLITEN**: Разрешение разбиения

— Биты 30:17 Резерв. не трогать.

— Бит 16 **COMPLSPLT**: Полное разбиение

— Биты 15:14 **XACTPOS**: Позиция передачи

В каждой передаче OUT идут все, первые, средние или последние полезные данные (>=188 байтам).

11: Все. (<= 188)

10: Начало. (> 188)

00: Серёдка. (> 188)

01: Конец. (< 188).

— Биты 13:7 **HUBADDR**: Адрес хаба

— Биты 6:0 **PRTADDR**: Адрес порта.

Регистр прерываний канала-x (OTG_HS_HCINTx) (x = 0..11, где x = Номер_канала)

Смещение адреса: 0x508 + (Номер_канала × 0x20)

По сбросу: 0x0000 0000

Читать регистр надо при стоящем бите `HCINT` в регистре `OTG_HS_GINTSTS`, но сначала надо узнать номер прервавшего канала из регистра `OTG_HS_HAINT`. Снятие бита в этом регистре чистит соответствующие биты в регистрах `OTG_HS_HAINT` и `OTG_HS_GINTSTS`.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																					DTERR	FRMOR	BBERR	TXERR	Reserved	ACK	NAK	STALL	Reserved	CHH	XFRC
																					rc_w1	rc_w1	rc_w1	rc_w1		rc_w1	rc_w1	rc_w1		rc_w1	rc_w1

— Биты 31:11 Резерв. не трогать.

— Бит 10 **DTERR**: Ошибка переключения данных

— Бит 9 **FRMOR**: Переполнение фрейма

— Бит 8 **BBERR**: Ошибка перекрёстных шумов

— Бит 7 **TXERR**: Ошибка передачи

Сбой CRC

Таймаут

Ошибка бита заполнения

Дрянной EOP

— Бит 6 Резерв. не трогать.

— Бит 5 **ACK**: Прерывание принятого/переданного ACK

— Бит 4 **NAK**: Прерывание принятого ответа NAK

— Бит 3 **STALL**: Прерывание принятого ответа STALL

— Бит 2 Резерв. не трогать.
— Бит 1 **CHH**: Канал остановлен
Indicates the transfer completed abnormally either because of any USB transaction error or in response to disable request by the application.
— Бит 0 **XFRC**: Передача завершена без ошибок

Регистр маски прерываний канала-x (OTG_HS_HCINTMSKx) (x = 0..11, где x = Номер_канала)

Смещение адреса: 0x508 + (Номер_канала × 0x20)

По сбросу: 0x0000 0000

31–11	10	9	8	7	6	5	4	3	2	1	0
Reserved	DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	Reserved	CHHM	XFRCM
	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

Маска прерывания события: 0- нельзя, 1 - можно.
— Биты 31:11 Резерв. не трогать.
— Бит 10 **DTERRM**: Ошибка переключения данных
— Бит 9 **FRMORM**: Переполнение фрейма
— Бит 8 **BBERRM**: Ошибка перекрытых шумов
— Бит 7 **TXERRM**: Ошибка передачи
— Бит 6 **NYET**: Ответ принят
— Бит 5 **ACKM**: Прерывание принятого/переданного ACK
— Бит 4 **NAKM**: Прерывание принятого ответа NAK
— Бит 3 **STALLM**: Прерывание принятого ответа STALL
— Бит 2 Резерв. не трогать.
— Бит 1 **CHHM**: Канал остановлен
— Бит 0 **XFRCM**: Передача завершена

Регистр размера передачи канала-x (OTG_HS_HCTSIZx) (x = 0..11, где x = Номер_канала)

Смещение адреса: 0x510 + (Номер_канала × 0x20)

По сбросу: 0x0000 0000

31	30–29	28–19	18–0
Reserved	DPID	PKTCNT	XFRSIZ
rw	rw	rw	rw

— Бит 31 Резерв. не трогать.
— Биты 30:29 **DPID:** PID данных начальной передачи
Программно задаёт PID данных начальной передачи. Для остальных передач управляется хостом.
00: DATA0
01: DATA2
10: DATA1
11: MDATA (non-control)/SETUP (control)
— Биты 28:19 **PKTCNT**: Счётчик пакетов
Программно пишется ожидаемое число передаваемых (OUT) или принимаемых (IN) пакетов передачи.
Декрементируется хостом после каждой успешной передачи пакета OUT/IN. При достижении 0 выдаётся прерывание завершения.
— Биты 18:0 **XFRSIZ**: Размер передачи
Для OUT это число байтов данных передачи хоста.
Для IN это размер приёмного буфера. Должен быть кратен максимальному размеру пакета IN (периодического и не-периодического).

Регистр адреса DMA канала-x (OTG_HS_HCDMAx) (x = 0..11, где x = Номер_канала)

Смещение адреса: 0x514 + (Номер_канала × 0x20)

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAADDR																															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 30:29 **DMAADDR:** Адрес

35.12.4. Регистры режима устройства

Регистр конфигурации устройства (OTG_HS_DCFG)

Смещение адреса: 0x800

По сбросу: 0x0220 0000

Конфигурация ядра по сбросу питания, некоторых команд или переучёта. После инициализации регистр изменять нельзя.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						PERSCHIVL		Reserved	Reserved										PFIVL		DAD							Reserved	NZLSOHSK	DSPD	
						rw	rw												rw		rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

— Биты 31:26 Резерв. не трогать.

— Биты 25:24 **PERSCHIVL**: Интервал периодического планирования

Это интервал времени, отводимого внутреннему DMA на выборку данных периодической точки IN. Задают как 25, 50 или 75% (микро)фрейма.

– Если периодические точки есть, то так и отводится
– Если их нет, то обслуживаются не-периодические, поле игнорируется
– После затраты указанного времени (микро)фрейма DMA идёт на не-периодические точки

00: 25% (микро)фрейма
01: 50% (микро)фрейма
10: 75% (микро)фрейма
11: Резерв

— Биты 22:13 Резерв. не трогать.

— Биты 12:11 **PFIVL**: Интервал периодического фрейма

Время выдачи прерывания конца периодического фрейма до полного завершения его передач.

00: 80% интервала фрейма
01: 85% интервала фрейма
10: 90% интервала фрейма
11: 95% интервала фрейма

— Биты 10:4 **DAD**: Адрес устройства

Пишется после каждой команды **SetAddress**.

— Бит 3 Резерв. не трогать.

— Бит 2 **NZLSOHSK**: Ответ на посылку OUT не-нулевой длины фазы статуса

0: Отдать принятый пакет OUT программе (нулевой или не-нулевой длины) и послать ответ на основе битов NAK и STALL конечной точки в регистре управления устройства.

1: На передачу статуса OUT не-нулевой длины ответить STALL, пакет программе не отдавать.

— Биты 1:0 **DSPD**: Скорость устройства

Скорость, показываемая устройством во время переучёта, но реальная скорость определяется хостом USB после завершения "чик-чирик" последовательности.

00: Резерв
01: Резерв
10: Резерв
11: FS (USB 1.1 Частота 48 MHz)

Регистр управления устройства (OTG_HS_DCTL)

Смещение адреса: 0x804

По сбросу: 0x0000 0000

31 … 12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	POPRGDNE	CGONAK	SGONAK	CGINAK	SGINAK	TCTL			GONSTS	GINSTS	SDIS	RWUSIG
	rw	w	w	w	w	rw	rw	rw	r	r	rw	rw

— Биты 31:12 Резерв. не трогать.

— Бит 11 **POPRGDNE**: Побудка включения завершена

Ставится программно после конфигурации регистра.

— Бит 10 **CGONAK**: Очистка глобального OUT NAK записью 1 в этот бит.

— Бит 9 **SGONAK**: Установка глобального OUT NAK

Выдаёт ответ NAK по всем конечным точкам OUT.

Ставится программно только при снятом бите GONAKEFF в регистре OTG_FS_GINTSTS.

— Бит 8 **CGINAK**: Очистка глобального IN NAK записью 1 в этот бит.

— Бит 7 **SGINAK**: Установка глобального IN NAK

Выдаёт ответ NAK по всем не-периодическим конечным точкам IN.

Ставится программно только при снятом бите GINAKEFF в регистре OTG_FS_GINTSTS.

— Биты 6:4 **TCTL**: Управление тестом

000: Выключен

001: Режим Test_J

010: Режим Test_K

011: Режим Test_SE0_NAK

100: Режим Test_Packet

101: Режим Test_Force_Enable

Иные: Резерв

— Бит 3 **GONSTS**: Состояние глобального OUT NAK

0: Ответ посылается на основе заполнения RxFIFO и состояния битов NAK и STALL.

1: Данные в RxFIFO не пишутся, независимо от заполнения. На все пакеты, кроме SETUP, отсылается NAK. Изохронные пакеты OUT пропадают втуне.

— Бит 2 **GINSTS**: Состояние глобального IN NAK

0: Ответ отсылается на основе наличия данных в передающем FIFO.

1: По всем не-периодическим точкам IN отсылается ответ NAK, независимо от данных в FIFO.

— Бит 1 **SDIS**: Программное отключение

При стоящем бите хост в упор не видит подключённого устройства и устройство не принимает сигналы по USB. При очистке этого бита после программного отключения хосту посылается событие подключения. После переподключения устройства хост начинает его переучёт.

0: Нормальная работа.

1: Ядро выдаёт хосту событие подключения.

— Бит 0 **RWUSIG**: Удалённое пробуждение

Установка бита посылает хосту сигнал удалённой побудки. Очищать бит надо в интервале от 1 ms до 15 ms после установки.

Таблица 214. Минимальная длительность программного отключения

Рабочая скорость	Состояние устройства	Минимальная длительность
HS	Работает	125 µs
FS	Останов	1 ms + 2.5 µs
FS	Простой	2.5 µs
FS	Работает	2.5 µs

Регистр состояния устройства (OTG_HS_DSTS)

Смещение адреса: 0x808

По сбросу: 0x0000 0010

Состояние событий ядра со стороны USB. Читать надо по прерываниям от регистра OTG_HS_DAINT).

31–22	21–8	7–4	3	2–1	0
Reserved	FNSOF	Reserved	EERR	ENUMSPD	SUSPSTS
	r		r	r	r

— Биты 31:22 Резерв. не трогать.
— Биты 21:8 **FNSOF**: Номер фрейма принятого SOF
— Биты 7:4 Резерв. не трогать.
— Бит 3 **EERR**: Непредсказуемая ошибка
Ставится ядром. Контроллер OTG_HS уходит в Останов и выдаётся прерывание раннего останова (бит ESUSP в OTG_HS_GINTSTS). Восстановить работу можно через программное отключение.
— Биты 2:1 **ENUMSPD**: Скорость после переучёта
00: HS
01: Резерв
10: Резерв
11: FS (частота PHY равна 48 MHz)
Иные: Резерв
— Бит 0 **SUSPSTS**: Состояние приостановки
В режиме устройства отражает сигнал Останова на шине USB. Ядро выходит из Останова при:
– Активности на линиях данных USB
– Установке бита удалённой побудки RWUSIG в регистре OTG_HS_DCTL.

Общий регистр маски прерываний конечных точек IN устройства (OTG_HS_DIEPMSK)

Смещение адреса: 0x810

По сбросу: 0x0000 0000

Относится к выдаче прерываний всех регистров OTG_HS_DIEPINTx. По умолчанию всё замаскировано.

31–14	13	12–9	8	7	6	5	4	3	2	1	0
Reserved	NAKM	Reserved	TXFURM	Reserved	INEPNEM	INEPNMM	ITTXFEMSK	TOM	AHBERRM	EPDM	XFRCM
	rw		rw		rw	rw	rw	rw	rw	rw	rw

Маска прерывания: 0 - Нельзя, 1 - Можно.
— Биты 31:7 Резерв. не трогать.
— Бит 6 **NAKM**: Прерывание NAK
— Биты 12:9 Резерв. не трогать.
— Бит 8 **TXFURM**: Исчерпание FIFO
— Бит 7 Резерв. не трогать
— Бит 6 **INEPNEM**: У конечной точки IN действует NAK
— Бит 5 **INEPNMM**: Принят токен IN с несовпадающей EP
— Бит 4 **ITTXFEMSK**: Принят токен IN при пустом TxFIFO
— Бит 3 **TOM**: Таймаут (Не-изохронные конечные точки)
— Бит 2 **AHBERRM**: Ошибка шины AHB
— Бит 1 **EPDM**: Конечная точка выключена
— Бит 0 **XFRCM**: Передача завершена

Общий регистр маски прерываний конечных точек OUT устройства (OTG_HS_DOEPMSK)

Смещение адреса: 0x814

По сбросу: 0x0000 0000

Относится к выдаче прерываний всех регистров OTG_HS_DOEPINTx По умолчанию всё замаскировано.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																	NYETMSK	NAKMSK	BERRM	Reserved			OPEM	Reserved	B2BSTUP	STSPHSRXM	OTEPDM	STUPM	AHBERRM	EPDM	XFRCM
																	rw	rw	rw				rw		rw	rw	rw	rw	rw	rw	rw

Маска прерывания: 0 - Нельзя, 1 - Можно.

— Биты 31:5 Резерв. не трогать.
— Бит 14 **NYETMSK**: Прерывание NYET
— Бит 13 **NAKMSK**: Прерывание NAK
— Бит 12 **BERRM**: Прерывание ошибки болтовни
— Биты 11:9 Резерв. не трогать.
— Бит 8 **OPEM**: Ошибка пакета OUT
— Бит 7 Резерв. не трогать.
— Бит 6 **B2BSTUP**: Ошибка череды пакетов SETUP
— Бит 5 **STSPHSRXM**: Ошибка фазы статуса в управляющей записи
— Бит 4 **OTEPDM**: Принят токен OUT выключенной управляющей точки OUT.
— Бит 3 **STUPM**: Фаза SETUP управляющей точки завершена.
— Бит 2 **AHBERRM**: Ошибка шины AHB
— Бит 1 **EPDM**: Конечная точка выключена
— Бит 0 **XFRCM**: Передача завершена

Регистр прерываний всех конечных точек устройства (OTG_HS_DAINT)

Смещение адреса: 0x818

По сбросу: 0x0000 0000

Регистр OTG_HS_DAINT содержит прерывания всех конечных OUT и IN точек устройства. Прерывания разрешаются битами OEPINT и IEPINT в регистре OTG_HS_GINTSTS. Для двунаправленных точек используются два бита. Биты в этом регистре ставятся и снимаются соответствующими битами в регистрах OTG_HS_DIEPINTx/OTG_HS_DOEPINTx.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OEPINT																IEPINT															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

— Биты 31:16 **OEPINT**: Прерывания конечных точек OUT
Для точек OUT:
Бит 16 для OUT конечной точки 0, бит 31 для OUT конечной точки 15
— Биты 15:0 **IEPINT**: Прерывания конечных точек IN
Для точек IN:
Бит 0 для IN конечной точки 0, бит 15 для IN конечной точки 15.

Регистр маски прерываний всех конечных точек устройства (OTG_HS_DAINTMSK)

Смещение адреса: 0x81C

По сбросу: 0x0000 0000

Маскирует прерывания по регистру OTG_HS_DAINT.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OEPM																IEPM															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Маска прерывания: 0 - Нельзя, 1 - Можно.

— Биты 31:16 **OEPINTM**: Прерывания конечных точек OUT
Для точек OUT:
Бит 16 для OUT конечной точки 0, бит 18 для OUT конечной точки 3
— Биты 15:0 **IEPINTM**: Прерывания конечных точек IN
Для точек IN:
Бит 0 для IN конечной точки 0, бит 3 для IN конечной точки 3.

Регистр времени разряда V_{BUS} устройства (OTG_HS_DVBUSDIS)

Смещение адреса: 0x828

По сбросу: 0x0000 17D7

31–16	15–0
Reserved	VBUSDT
	rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw

— Биты 31:16 Резерв, не трогать.

— Биты 15:0 **VBUSDT**: Время разряда V_{BUS} устройства после импульса во время SRP.

Вычисляется в тактах PHY / 1 024. Зависит от нагрузки V_{BUS}, можно подстраивать.

Регистр времени импульса V_{BUS} устройства (OTG_HS_DVBUSPULSE)

Смещение адреса: 0x82C

По сбросу: 0x0000 05B8

31–12	11–0
Reserved	DVBUSP
	rw rw rw rw rw rw rw rw rw rw rw rw

— Биты 31:12 Резерв, не трогать.

— Биты 11:0 **DVBUSP:** Время импульса V_{BUS} устройства во время SRP.

Вычисляется в тактах PHY / 1 024.

Регистр порога устройства (OTG_HS_DTHRCTL)

Смещение адреса: 0x830

По сбросу: 0x0000 0000

31–28	27	26	25–17	16	15–11	10–2	1	0
Reserved	ARPEN	Reserved	RXTHRLEN	RXTHREN	Reserved	TXTHRLEN	ISOTHREN	NONISOTHREN
	rw		rw rw rw rw rw rw rw rw rw	rw		rw rw rw rw rw rw rw rw rw	rw	rw

— Биты 31:28 Резерв, не трогать.

— Бит 27 **ARPEN**: Разрешение паркинга арбитром точек IN

При включённом пороге и стоящем бите арбитр паркует точку IN, для которой по USB принят токен. Так избегают исчерпания. По умолчанию паркинг включён.

— Бит 26 Резерв, не трогать.

— Биты 25:17 **RXTHRLEN:** Длина порога приёма в словах

Также это число принятых по USB данных до начала передачи по AHB. Должна быть не менее 8 слов. Рекомендуется RXTHRLEN равная длине пакета AHB (HBSTLEN в OTG_HS_GAHBCFG).

— Бит 16 **RXTHREN**: Разрешение порога приёма

— Биты 15:11 Резерв, не трогать.

— Биты 10:2 **TXTHRLEN:** Длина порога передачи в словах

Задаёт число байтов соответствующей точки в передающем FIFO для начала передачи по USB. Должна быть не менее 8 слов. Определяет порог изохронных и не-изохронных точек IN. Рекомендуется TXTHRLEN равная длине пакета AHB (HBSTLEN в OTG_HS_GAHBCFG).

— Бит 1 **ISOTHREN**: Разрешение порога точек ISO IN

— Бит 0 **NOISOTHREN**: Разрешение порога не изохронных точек IN

Регистр маски прерываний пустого FIFO конечных точек IN (OTG_HS_DIEPEMPMSK)

Смещение адреса: 0x834

По сбросу: 0x0000 0000

31–16	15–0
Reserved	INEPTXFEM
	rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw

Маска прерывания: 0 - Нельзя, 1 - Можно.

— Биты 31:16 Резерв, не трогать.

— Биты 15:0 **INEPTXFEM**: TxFIFO конечной точки IN пуст

Маскируют биты прерываний по TXFE регистров OTG_FS_DIEPINTx.

Бит 0 для IN конечной точки 0, бит 15 для IN конечной точки 15.

Регистр прерываний каждой конечной точки (OTG_HS_DEACHINT)

Смещение адреса: 0x838

По сбросу: 0x0000 0000

Один бит для точки 1 IN и один бит для точки 1 OUT

31–18	17	16–2	1	0
Reserved	OEP1INT	Reserved	IEP1INT	Reserved
	r		r	

— Биты 31:18 Резерв, не трогать.

— Бит 17 **OEP1INT**: Прерывание точки 1 OUT

— Биты 16:2 Резерв, не трогать.

— Бит 1 **IEP1INT**: Прерывание точки 1 IN

— Бит 0 Резерв, не трогать.

Регистр маски прерываний каждой конечной точки (OTG_HS_DEACHINTMSK)

Смещение адреса: 0x83C

По сбросу: 0x0000 0000

Один бит для точки 1 IN и один бит для точки 1 OUT

31–18	17	16–2	1	0
Reserved	OEP1INTM	Reserved	IEP1INTM	Reserved
	rw		rw	

— Биты 31:18 Резерв, не трогать.

— Бит 17 **OEP1INTM**: Прерывание точки 1 OUT

— Биты 16:2 Резерв, не трогать.

— Бит 1 **IEP1INTM**: Прерывание точки 1 IN

— Бит 0 Резерв, не трогать.

Регистр маски прерываний каждой IN конечной точки-1 (OTG_HS_DIEPEACHMSK1)

Смещение адреса: 0x844

По сбросу: 0x0000 0000

31–14	13	12–11	10	9	8	7	6	5	4	3	2	1	0
Reserved	NAKM	Reserved		BIM	TXFURM	Reserved	INEPNEM	INEPNMM	ITTXFEMSK	TOM	AHBERRM	EPDM	XFRCM
	rw			rw	rw		rw	rw	rw	rw	rw	rw	rw

— Биты 31:14 Резерв, не трогать.

— Бит 13 **NAKM**: Маска прерывания NAK

— Биты 12:10 Резерв, не трогать.

— Бит 9 **BIM**: Маска прерывания BNA

— Бит 8 **TXFURM**: Маска прерывания исчерпания FIFO

— Бит 7 Резерв, не трогать.
— Бит 6 **INEPNEM**: Маска прерывания NAK точки IN работает
— Бит 5 **INEPNMM**: Маска прерывания несовпадения токена IN с EP
— Бит 4 **ITTXFEMSK**: Маска прерывания токен IN принят при пустом TxFIFO
— Бит 3 **TOM**: Маска прерывания таймаута (не-изохронные точки)
— Бит 2 **AHBERRM**: Маска прерывания ошибки шины AHB
— Бит 1 **EPDM**: Маска прерывания выключения точки
— Бит 0 **XFRCM**: Маска прерывания конца передачи

Регистр маски прерываний каждой OUT конечной точки-1 (OTG_HS_DOEPEACHMSK1)

Смещение адреса: 0x884

По сбросу: 0x0000 0000

31:15	14	13	12	11:10	9	8	7	6	5	4	3	2	1	0
Reserved	NYETM	NAKM	BERRM	Reserved	BIM	TXFURM	Reserved	INEPNEM	INEPNMM	ITTXFEMSK	TOM	AHBERRM	EPDM	XFRCM
	rw	rw	rw		rw	rw		rw	rw	rw	rw	rw	rw	rw

— Биты 31:15 Резерв, не трогать.
— Бит 14 **NYETM**: Маска прерывания NYET
— Бит 13 **NAKM**: Маска прерывания NAK
— Бит 12 **BEKRR**: Маска прерывания ошибки болтовни
— Биты 11:10 Резерв, не трогать.
— Бит 9 **BIM**: Маска прерывания BNA
— Бит 8 **OPEM**: Маска прерывания ошибки пакета OUT
— Биты 7:3 Резерв, не трогать.
— Бит 2 **AHBERRM**: Маска прерывания ошибки шины AHB
— Бит 1 **EPDM**: Маска прерывания выключения точки
— Бит 0 **XFRCM**: Маска прерывания конца передачи

Регистр управления конечной точки-x (OTG_HS_DIEPCTLx) (x = 0..7, где x = номер точки)

Смещение адреса: 0x900 + 0x20 * x

По сбросу: 0x0000 0000

Применяется для логических точек не 0.

31	30	29	28	27	26	25:22	21	20	19:18	17	16	15	14:11	10:0
EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM	Stall	Reserved	EPTYP	NAKSTS	EONUM/DPID	USBAEP	Reserved	MPSIZ
rs	rs	w	w	w	w	rw	rw/ rs		rw	r	r	rw		rw

— Бит 31 **EPENA**: Включение конечной точки
Снимается ядром перед выдачей прерываний:
– Фаза SETUP завершена
– Точка выключена
– Передача завершена
— Бит 30 **EPDIS**: Выключение конечной точки
Установка бита останавливает даже незавершённую передачу данных конечной точки. Далее надо дождаться прерывания выключения точки. Снимается ядром перед выдачей прерывания. Останавливать можно только включённую конечную точку.
— Бит 29 **SODDFRM**: Установить нечётный фрейм изохронных IN и OUT
— Бит 28 **SD0PID/SEVNFRM**: Установить DATA0 PID прерывающих/групповых IN
— Бит 27 **SNAK**: Установка ответа NAK

Программно задаёт выдачу NAK этой точки. Ядро может ставить его по концу передачи или после приёма пакета SETUP для этой точки.

— Бит 26 **CNAK**: Очистка NAK записью в этот бит

— Биты 25:22 **TXFNUM**: Номер TxFIFO точки IN.

— Бит 21 **STALL**: Ответ STALL

Для не-управляющих, не-изохронных точек IN:

Ставится программно, снимается ядром после приёма пакета SETUP. Имеет приоритет перед битами NAK, Глобального IN NAK и Глобального OUT NAK.

Для управляющих точек:

Ставится программно, снимается ядром после приёма пакета SETUP. Имеет приоритет перед битами NAK, Глобального IN NAK и Глобального OUT NAK. Безотносительно этого бита, ядро на пакеты SETUP отвечает ACK.

— Бит 20 Резерв, не трогать.

— Биты 19:18 **EPTYP**: Тип конечной точки

00: Управляющая

01: Изохронная

10: Групповая

11: Прерывающая

— Бит 17 **NAKSTS**: Состояние NAK

0: Не-NAK ответы на основании состояния FIFO

1: Ответы NAK.

При стоящем бите ядро останавливает передачи данных даже при их наличии в TxFIFO. На пакет SETUP ядро посылает ACK независимо от состояния этого бита.

— Бит 16 **EONUM/DPID**: Чётный/нечётный фрейм

Только изохронные точки IN.

По нему ядро использует поля SEVNFRM и SODDFRM этого регистра.

0: Чётный

1: Нечётный

DPID: PID данных

Только прерывающие/групповые точки IN.

Это PID первого пакета для передачи или приёма после активации точки. Для задания DATA0 или DATA1 PID используется поле SD0PID.

0: DATA0

1: DATA1

— Биты 29:28 Резерв, не трогать.

— Бит 15 **USBAEP**: Активная конечная точка USB

Ядро снимает бит для всех точек (кроме EP 0) после обнаружения сброса USB. После приёма команд SetConfiguration и SetInterface надо установить регистры точки и этот бит.

— Биты 14:11 Резерв, не трогать.

— Биты 10:0 **MPSIZ**: Максимальный размер пакета в байтах

Регистр управления конечной точкой OUT 0 (OTG_HS_DOEPCTL0)

Смещение адреса: 0xB00

По сбросу: 0x0000 8000

31	30	29 28	27	26	25 24 23 22	21	20	19 18	17	16	15	14 13 12 11 10 9 8 7 6 5 4 3 2	1 0
EPENA	EPDIS	Reserved	SNAK	CNAK	Reserved	Stall	SNPM	EPTYP	NAKSTS	Reserved	USBAEP	Reserved	MPSIZ
w	r		w	w		rs	rw	r r	r		r		r r

— Бит 31 **EPENA**: Включить конечную точку

Установка бита запускает передачу данных конечной точки 0.

Снимается ядром перед выдачей прерываний:

– Фаза SETUP завершена

– Точка выключена

– Передача завершена

— Бит 30 **EPDIS**: Выключение конечной точки

Программно не выключается.

— Биты 29:28 Резерв, не трогать.

— Бит 27 **SNAK**: Установка NAK

Программно задаёт выдачу NAK этой точки. Ядро может ставить его после завершения передачи точки OUT или приёма пакета SETUP.

— Бит 26 **CNAK**: Очистка NAK

— Биты 25:22 Резерв, не трогать.

— Бит 21 **STALL**: Ответ STALL

Ставится программно, снимается ядром после приёма пакета SETUP. Имеет приоритет перед битами NAK и Глобального OUT NAK. На пакет SETUP ядро посылает ACK независимо от состояния этого бита.

— Бит 20 **SNPM**: Режим Snoop (Подглядывания)

В этом режиме ядро не проверяет правильность пакетов OUT перед записью в память.

— Биты 19:18 **PTYP**: Тип конечной точки, ноль для управляющей.

— Бит 17 **NAKSTS**: Состояние NAK

0: Не-NAK ответы на основании состояния FIFO

1: Ответы NAK.

При стоящем бите ядро останавливает приём данных даже при наличии места в RxFIFO. На пакет SETUP ядро посылает ACK независимо от состояния этого бита.

— Бит 16 Резерв, не трогать.

— Бит 15 **USBAEP**: Активная конечная точка, всегда равен 1.

— Биты 14:2 Резерв, не трогать.

— Бит 1:0 **MPSIZ**: Максимальный размер пакета точки OUT

00: 64 байта

01: 32 байта

10: 16 байт

11: 8 байт

Регистр управления конечной OUT точки-x (OTG_HS_DOEPCTLx) (x = 1..3, где x = Номер_конечной_точки)

Смещение адреса точки OUT: 0xB00 + (Номер_точки × 0x20)

По сбросу: 0x0000 0000

31	30	29	28	27	26	25 24 23 22	21	20	19 18	17	16	15	14 13 12 11	10 9 8 7 6 5 4 3 2 1 0
EPENA	EPDIS	SODDFRM/SD1PID	SD0PID/SEVNFRM	SNAK	CNAK	Reserved	Stall	SNPM	EPTYP	NAKSTS	EONUM/DPID	USBAEP	Reserved	MPSIZ
rs	rs	w	w	w	w		rw/ rs	rw	rw	r	r	rw		rw

— Бит 31 **EPENA**: Включение конечной точки IN и OUT.

Установка бита запускает передачу данных конечной точки.

Снимается ядром перед выдачей прерываний:

– Фаза SETUP завершена

– Точка выключена

– Передача завершена

— Бит 30 **EPDIS**: Выключение конечной точки

Установка бита останавливает даже незавершённую передачу/приём данных конечной точки. Далее надо дождаться прерывания выключения точки. Снимается ядром перед выдачей прерывания. Останавливать можно только включённую конечную точку.

— Бит 29 **SD1PID**: Установить PID DATA1 (DPID) прерывающих/групповых точек IN и OUT.

SODDFRM: Установить нечётный фрейм изохронных точек IN и OUT (поле EONUM)

— Бит 28 **SD0PID**: Установить PID DATA0 (DPID) прерывающих/групповых точек OUT

SEVNFRM: Установить чётный фрейм изохронных точек OUT (поле EONUM)

— Бит 27 **SNAK**: Установка NAK

Программно задаёт выдачу NAK этой точки. Ядро может ставить его после завершения передачи точки OUT или приёма пакета SETUP.

— Бит 26 **CNAK**: Очистка NAK

— Биты 25:22 Резерв, не трогать.

— Бит 21 **STALL**: Ответ STALL

Для не-управляющих, не-изохронных точек IN (доступ rw).

Программно останавливает все передачи от хоста точке. Имеет приоритет перед битами NAK, Глобального IN NAK и Глобального OUT NAK. Снимается только программно.

Для управляющих точек (доступ rs).

Ставится программно, снимается ядром после приёма пакета SETUP. Имеет приоритет перед битами NAK, Глобального IN NAK и Глобального OUT NAK. На пакет SETUP ядро посылает ACK независимо от состояния этого бита.

— Бит 20 **SNPM**: Режим Snoop (Подглядывания)

В этом режиме ядро не проверяет правильность пакетов OUT перед записью в память.

— Биты 19:18 **EPTYP**: Тип конечной точки

00: Управляющая

01: Изохронная

10: Групповая

11: Прерывающая

— Бит 17 **NAKSTS**: Состояние NAK

0: Не-NAK ответы на основании состояния FIFO

1: Ответы NAK.

При стоящем бите ядро останавливает приём данных даже при наличии места в RxFIFO. На пакет SETUP ядро посылает ACK независимо от состояния этого бита.

— Бит 16 **EONUM**: Чётный/Нечётный фрейм изохронных точек IN.

Устанавливается записью в биты SEVNFRM и SODDFRM этого регистра.

0: Чётный

1: Нечётный

DPID: PID первого пакета данных прерывающей/групповой конечной точки

Устанавливается записью в бит SD0PID:

0: DATA0

1: DATA1

— Бит 15 **USBAEP**: Активная конечная точка USB

По сбросу USB ядро снимает этот бит для всех точек, кроме EP 0. Ставить надо после завершения исполнения команд **SetConfiguration** и **SetInterface**.

— Биты 14:11 Резерв, не трогать.

— Биты 10:0 **MPSIZ**: Максимальный размер пакета в байтах

Регистр прерываний конечной точки-x (OTG_HS_DIEPINTx) (x = 0..7, где x = Номер_конечной_точки)

Смещение адреса: 0x908 + (Номер_точки × 0x20)

По сбросу: 0x0000 0080

Читать можно только при стоящем бите `IEPINT` в `OTG_HS_GINTSTS`, но сначала надо узнать номер точки по регистру `OTG_HS_DAINT`. Снятие бита в этом регистре чистит соответствующие биты в регистрах `OTG_HS_DAINT` и `OTG_HS_GINTSTS`.

31 … 14	13	12	11	10 … 9	8	7	6	5	4	3	2	1	0
Reserved	NAK	Reserved	PKTDRPSTS	Reserved	TXFIFOUDRN	TXFE	INEPNE	INEPNM	ITTXFE	TOC	AHBERR	EPDISD	XFRC
	rc_w1		rc_w1		rc_w1	r	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

— Биты 31:14 Резерв, не трогать.

— Бит 13 **NAK**: Прерывание NAK

Ставится ядром при передаче или приёме NAK. Для изохронных точек IN при передаче пакета нулевой длины из-за недоступности данных в Tx FIFO.

— Бит 12 Резерв, не трогать.

— Бит 11 **PKTDRPSTS**: Пакет ISOC OUT отброшен. Без прерывания.

— Биты 10:9 Резерв, не трогать.
— Бит 8 **TXFIFOUDRN**: Исчерпание Tx FIFO. Действует при включённом пороге.
— Бит 7 **TXFE**: Передающий FIFO полу- или полностью пуст
Уровень пустоты TxFIFO задаётся битом TXFELVL в регистре OTG_FS_GAHBCFG.
— Бит 6 **INEPNE**: NAK конечной точки IN в действии
Снимается записью бита CNAK в регистре OTG_FS_DIEPCTLx. Установка бита вызывает прерывание. По шине USB не обязательно пойдёт NAK, бит STALL приоритетнее.
— Бит 5 Резерв, не трогать.
— Бит 4 **ITTXFE**: Принят токен IN при пустом не-периодическом TxFIFO
Выставляется прерывание этой точки.
— Бит 3 **TOC**: Таймаут управляющей точки IN.
— Бит 13 **AHBERR**: Ошибка шины AHB.
— Бит 1 **EPDISD**: Прерывание выключенной конечной точки
— Бит 0 **XFRC**: Прерывание конца передачи по AHB или USB.

Регистр прерываний конечной точки-x (OTG_HS_DOEPINTx) (x = 0..7, где x = Номер_конечной_точки)

Смещение адреса: 0xB08 + (Endpoint_number × 0x20)

По сбросу: 0x0000 0080

Читать можно только при стоящем бите `OEPINT` в регистре `OTG_HS_GINTSTS`, но сначала надо узнать номер точки по регистру `OTG_HS_DAINT`. Снятие бита в этом регистре чистит соответствующие биты в регистрах `OTG_HS_DAINT` и `OTG_HS_GINTSTS`.

31 … 15	14	13	12	11 … 9	8	7	6	5	4	3	2	1	0
Reserved	NYET	NAK	BERR	Reserved	OUTPKTERR	Reserved	B2BSTUP	Reserved	OTEPDIS	STUP	AHBERR	EPDISD	XFRC
	rc_w1	rc_w1	rc_w1		rc_w1		rc_w1		rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

— Биты 31:15 Резерв, не трогать.
— Бит 14 **NYET**: По не-изохронной точке OUT принят ответ NYET.
— Бит 13 **NAK**: Прерывание NAK
Ставится ядром при передаче или приёме NAK. Для изохронных точек IN при передаче пакета нулевой длины из-за недоступности данных в Tx FIFO.
— Бит 12 **BERR**: Ошибка болтовни.
— Биты 11:9 Резерв, не трогать.
— Бит 8 **OUTPKTERR**: Ошибка пакета OUT.
— Бит 7 Резерв, не трогать.
— Бит 6 **B2BSTUP**: Принята череда (больше 3) пакетов SETUP управляющей точки OUT.
— Бит 5 Резерв, не трогать.
— Бит 4 **OTEPDIS**: Принят пакет OUT выключенной управляющей точки (с прерыванием)
— Бит 3 **STUP**: Фаза SETUP управляющей точки OUT завершена (с прерыванием)
— Бит 2 **AHBERR**: Ошибка шины AHB.
— Бит 1 **EPDISD**: Прерывание выключения конечной точки
— Бит 0 **XFRC**: Прерывание конца передачи по AHB или USB.

Регистр размера передачи конечной точки IN 0 (OTG_HS_DIEPTSIZ0)

Смещение адреса: 0x910

По сбросу: 0x0000 0000

Писать надо до включения точки 0. После этого можно только читать.

31 … 21	20	19	18 … 7	6	5	4	3	2	1	0
Reserved	PKTCNT		Reserved	XFRSIZ						
	rw	rw		rw	rw	rw	rw	rw	rw	rw

— Биты 31:21 Резерв, не трогать.
— Биты 20:19 **PKTCNT**: Счётчик общего числа пакетов передачи

Декрементируется при каждом чтении из TxFIFO.

— Биты 18:7 Резерв, не трогать.

— Биты 6:0 **XFRSIZ**: Размер передачи в байтах (с прерыванием после исчерпания)

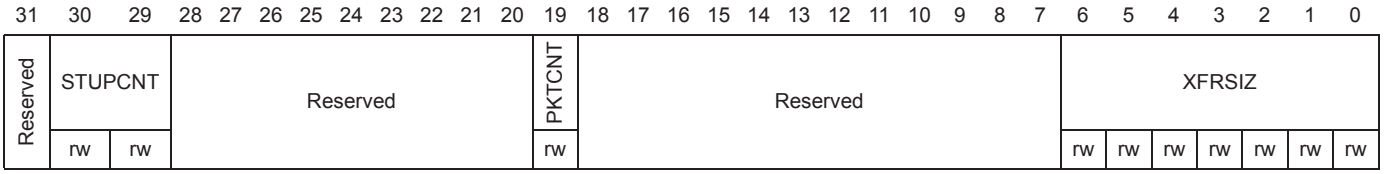
Уменьшается при каждой записи в TxFIFO.

Регистр размера передачи конечной точки OUT 0 (OTG_HS_DOEPTSIZ0)

Смещение адреса: 0xB10

По сбросу: 0x0000 0000

Писать надо до включения точки 0. После этого можно только читать.



— Бит 31 Резерв, не трогать.

— Биты 30:29 **STUPCNT**: Счётчик череды пакетов SETUP

01: 1 пакет

10: 2 пакета

11: 3 пакета

— Биты 28:20 Резерв, не трогать.

— Бит 19 **PKTCNT**: Счётчик пакетов

Обнуляется при записи в RxFIFO.

— Биты 18:7 Резерв, не трогать.

— Биты 6:0 **XFRSIZ**: Размер передачи в байтах (с прерыванием после исчерпания)

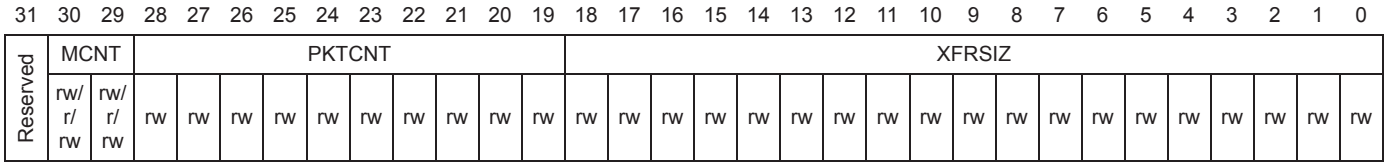
Уменьшается при каждом чтении из RxFIFO.

Регистр размера передачи конечной точки-x (OTG_HS_DIEPTSIZx) (x = 1..3, где x = Номер_конечной_точки)

Смещение адреса: 0x910 + (Endpoint_number × 0x20)

По сбросу: 0x0000 0000

Писать надо до включения точки. После этого можно только читать.



— Бит 31 Резерв, не трогать.

— Биты 30:29 **MCNT**: Многосчётчик пакетов в периодическом фрейме IN.

01: 1 пакет

10: 2 пакета

11: 3 пакета

— Биты 28:19 **PKTCNT**: Счётчик общего числа пакетов передачи

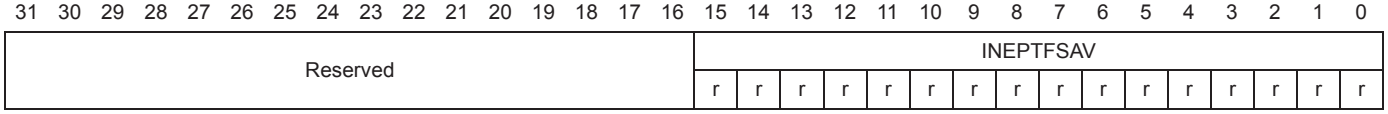
Декрементируется при каждом чтении из TxFIFO.

— Биты 18:0 **XFRSIZ**: Размер передачи в байтах

Уменьшается при каждой записи в TxFIFO.

Регистр состояния FIFO конечной точки IN (OTG_HS_DTXFSTSx) (x = 0..3, где x = Номер_конечной_точки)

Смещение адреса точки IN: 0x918 + (Endpoint_number × 0x20)



— Биты 31:16 Резерв, не трогать.

— Биты 15:0 **INEPTFSAV**: Доступное место в TxFIFO точки IN в 32-бит словах

0x0: TxFIFO полон
0x1: 1 слово
0x2: 2 слова
0xn: n слов
Иное: Резерв

Регистр размера передачи конечной точки-x OUT (OTG_HS_DOEPTSIZx) (x = 1..5, где x = Номер_конечной_точки)

Смещение адреса: 0xB10 + (Endpoint_number × 0x20)

По сбросу: 0x0000 0000

Писать надо до включения точки. После этого можно только читать.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	RXDPID/STUPCNT		PKTCNT										XFRSIZ																		
	rw/r/rw	rw/r/rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Бит 31 Резерв, не трогать.

— Биты 30:29 **RXDPID**: PID принятых данных изохронной точки OUT.

00: DATA0
01: DATA2
10: DATA1
11: MDATA

STUPCNT: Максимальное число пакетов SETUP управляющей точки OUT.

01: 1 пакет
10: 2 пакета
11: 3 пакета

— Биты 28:19 **PKTCNT**: Счётчик общего числа пакетов передачи

Декрементируется при каждой записи в RxFIFO.

— Биты 18:0 **XFRSIZ**: Размер передачи в байтах (с прерыванием после исчерпания)

Уменьшается при каждом чтении из RxFIFO.

Регистр адреса DMA передачи конечной точки-x (OTG_HS_DIEPDMAx/ OTG_HS_DOEPDMAx) (x = 1..5, где x = Номер_конечной_точки)

Смещение адреса точек IN: 0x914 + (Endpoint_number × 0x20)

Смещение адреса точек OUT: 0xB14 + (Endpoint_number × 0x20)

По сбросу: Чёрти-что

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAADDR																															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:0 **DMAADDR**: Адрес памяти для DMA.

NB: Для управляющих точек там хранятся данные пакетов OUT и SETUP. При череде пакетов SETUP более 3, данные SETUP в памяти перекрываются. Регистр инкрементируется при каждой передаче AHB. Словно выравнивание.

35.12.5. Регистр питания и тактов (OTG_HS_PCGCCTL)

Смещение адреса: 0xE00

По сбросу: 0x0000 0000

Доступен в обоих режимах.

31:5	4	3:2	1	0
Reserved	PHYSUSP	Reserved	GATEHCLK	STPPCLK
	rw		rw	rw

— Биты 31:5 Резерв, не трогать.

— Бит 4 **PHYSUSP**: PHY остановлен

— Биты 3:2 Резерв, не трогать.

— Бит 1 **GATEHCLK**: Разрешение подачи HCLK

Ставится и снимается программно.

— Бит 0 **STPPCLK**: Остановка тактов PHY

Ставится и снимается программно.

35.12.6. Карта регистров OTG_HS

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	OTG_HS_GOTGCTL	Reserved												BSVLD	ASVLD	DBCT	CIDSTS	Reserved				DHNPEN	HSHNPEN	HNPRQ	HNGSCS	Reserved						SRQ	SRQSCS
	Reset value													0	0	0	1					0	0	0	0							0	0
0x004	OTG_HS_GOTGINT	Reserved												DBCDNE	ADTOCHG	HNGDET	Reserved							HNSSCHG	SRSSCHG	Reserved					SEDET	Res.	
	Reset value													0	0	0								0	0						0		
0x008	OTG_HS_GAHBCFG	Reserved																							PTXFELVL	TXFELVL	Reserved						GINT
	Reset value																								0	0							0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00C	OTG_HS_GUSBCFG	CTXPKT	FDMOD	FHMOD	Reserved			ULPIIPD	PTCI	PCCI	TSDPS	ULPIEVBUSI	ULPIEVBUSD	ULPICSM	ULPIAR	ULPIFSLS	Reserved	PHYLPCS	Reserved	TRDT				HNPCAP	SRPCAP	Reserved	PHYSEL	Reserved			TOCAL		
	Reset value	0	0	0				0	0	0	0	0	0	0	0	0		0		0	1	0	1	0	0		1				0	0	0
0x010	OTG_HS_GRSTCTL	AHBIDL	DMAREQ	Reserved																			TXFNUM					TXFFLSH	RXFFLSH	Reserved	FCRST	HSRST	CSRST
	Reset value	1	0																				0	0	0	0	0	0	0	0	0	0	0
0x014	OTG_HS_GINTSTS	WKUINT	SRQINT	DISCINT	CIDSCHG	Reserved	PTXFE	HCINT	HPRTINT	Reserved	DATAFSUSP	IPXFR/INCOMPISOOUT	IISOIXFR	OEPINT	IEPINT	Reserved		EOPF	ISOODRP	ENUMDNE	USBRST	USBSUSP	ESUSP	Reserved		GONAKEFF	GINAKEFF	NPTXFE	RXFLVL	SOF	OTGINT	MMIS	CMOD
	Reset value	0	0	0	0		1	0	0		0	0	0	0	0			0	0	0	0	0	0			0	0	1	0	0	0	0	0
0x018	OTG_HS_GINTMSK	WUIM	SRQIM	DISCINT	CIDSCHGM	Reserved	PTXFEM	HCIM	PRTIM	Reserved	FSUSPM	IPXFRM/IISOOXFRM	IISOIXFRM	OEPINT	IEPINT	Reserved		EOPFM	ISOODRPM	ENUMDNEM	USBRST	USBSUSPM	ESUSPM	Reserved		GONAKEFFM	GINAKEFFM	NPTXFEM	RXFLVLM	SOFM	OTGINT	MMISM	Reserved
	Reset value	0	0	0	0		0	0	0		0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0	0	
0x01C	OTG_HS_GRXSTSR (Host mode)	Reserved											PKTSTS				DPID		BCNT											CHNUM		
	Reset value												0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	OTG_HS_GRXSTSR (peripheral mode)	Reserved							FRMNUM				PKTSTS				DPID		BCNT											EPNUM		
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	OTG_HS_GRXSTSP (Host mode)	Reserved											PKTSTS				DPID		BCNT											CHNUM		
	Reset value												0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	OTG_HS_GRXSTSP (peripheral mode)	Reserved							FRMNUM				PKTSTS				DPID		BCNT											EPNUM		
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	OTG_HS_GRXFSIZ	Reserved																RXFD															
	Reset value																	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x028	OTG_HS_GNPTXFSIZ (Host mode)	NPTXFD																NPTXFSA															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
	OTG_HS_GNPTXFSIZ (peripheral mode)	TX0FD																TX0FSA															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x02C	OTG_HS_GNPTXSTS	Res.	NPTXQTOP							NPTQXSAV								NPTXFSAV															
	Reset value		0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0x030	OTG_HS_GI2CCTL	BSYDNE	RW	Reserved	I2CDATSE0	I2CDEVADR		Reserved	ACK	I2CEN	ADDR							REGADDR						RWDATA									
0x038	OTG_HS_GCCFG	Reserved										NOVBUSSENS	SOFOUTEN	VBUSBSEN	VBUSASEN	.I2CPADEN	.PWRDWN	Reserved															
	Reset value											0	0	0	0	0	0																
0x03C	OTG_HS_CID	PRODUCT_ID																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0
0x100	OTG_HS_HPTXFSIZ	PTXFD																PTXSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0x104	OTG_HS_DIEPTXF1	INEPTXFD																INEPTXSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0x108	OTG_HS_DIEPTXF2	INEPTXFD																INEPTXSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0x10C	OTG_HS_DIEPTXF3	INEPTXFD																INEPTXSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0x110	OTG_HS_DIEPTXF4	INEPTXFD																INEPTXSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0x400	OTG_HS_HCFG	Reserved																													FSLSS	FSLSPCS	
	Reset value																														0	0	0
0x404	OTG_HS_HFIR	Reserved																FRIVL															
	Reset value																	1	1	1	0	1	0	1	0	0	1	1	0	0	0	0	0
0x408	OTG_HS_HFNUM	FTREM																FRNUM															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x410	OTG_HS_HPTXSTS	PTXQTOP								PTXQSAV								PTXFSAVL															
	Reset value	0	0	0	0	0	0	0	0	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x414	OTG_HS_HAINT	Reserved																HAINT															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x418	OTG_HS_HAINTMSK	Reserved																HAINTM															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x440	OTG_HS_HPRT	Reserved													PSPD	PTCTL			PPWR	PLSTS		Reserved	PRST	PSUSP	PRES	POCCHNG	POCA	PENCHNG	PENA	PCDET	PCSTS	
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x500	OTG_HS_HCCHAR0	CHENA	CHDIS	ODDFRM	DAD							MC		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x520	OTG_HS_HCCHAR1	CHENA	CHDIS	ODDFRM	DAD							MC		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x540	OTG_HS_HCCHAR2	CHENA	CHDIS	ODDFRM	DAD							MC		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x560	OTG_HS_HCCHAR3	CHENA	CHDIS	ODDFRM	DAD							MC		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x580	OTG_HS_HCCHAR4	CHENA	CHDIS	ODDFRM	DAD							MC		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5A0	OTG_HS_HCCHAR5	CHENA	CHDIS	ODDFRM	DAD							MC		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C0	OTG_HS_HCCHAR6	CHENA	CHDIS	ODDFRM	DAD							MC		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5E0	OTG_HS_HCCHAR7	CHENA	CHDIS	ODDFRM	DAD							MC		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x600	OTG_HS_HCCHAR8	CHENA	CHDIS	ODDFRM	DAD							MC		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x620	OTG_HS_HCCHAR9	CHENA	CHDIS	ODDFRM	DAD							MC		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x640	OTG_HS_HCCHAR10	CHENA	CHDIS	ODDFRM	DAD							MC		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x660	OTG_HS_HCCHAR11	CHENA	CHDIS	ODDFRM	DAD							MC		EPTYP		LSDEV	Reserved	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x504	OTG_HS_HCSPLT0	SPLITEN	Reserved														COMPLSPLT	XACTPOS		HUBADDR							PRTADDR						
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x508	OTG_HS_HCINT0	Reserved																					DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x524	OTG_HS_HCSPL1	SPLITEN	Reserved														COMPLSPLT	XACTPOS		HUBADDR							PRTADDR						
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x528	OTG_HS_HCINT1	Reserved																					DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x544	OTG_HS_HCSPLT2	SPLITEN	Reserved														COMPLSPLT	XACTPOS		HUBADDR							PRTADDR						
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x548	OTG_HS_HCINT2	Reserved																					DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x564	OTG_HS_HCSPLT3	SPLITEN	Reserved														COMPLSPLT	XACTPOS		HUBADDR							PRTADDR						
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x568	OTG_HS_HCINT3	Reserved																					DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x584	OTG_HS_HCSPLT4	SPLITEN	Reserved														COMPLSPLT	XACTPOS		HUBADDR							PRTADDR							
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x588	OTG_HS_HCINT4	Reserved																					DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x5A4	OTG_HS_HCSPLT5	SPLITEN	Reserved														COMPLSPLT	XACTPOS		HUBADDR							PRTADDR							
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5A8	OTG_HS_HCINT5	Reserved																					DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x5C4	OTG_HS_HCSPLT6	SPLITEN	Reserved														COMPLSPLT	XACTPOS		HUBADDR							PRTADDR							
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C8	OTG_HS_HCINT6	Reserved																					DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x5E4	OTG_HS_HCSPLT7	SPLITEN	Reserved														COMPLSPLT	XACTPOS		HUBADDR							PRTADDR							
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5E8	OTG_HS_HCINT7	Reserved																					DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x604	OTG_HS_HCSPLT8	SPLITEN	Reserved														COMPLSPLT	XACTPOS		HUBADDR							PRTADDR							
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x608	OTG_HS_HCINT8	Reserved																					DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x624	OTG_HS_HCSPLT9	SPLITEN	Reserved														COMPLSPLT	XACTPOS		HUBADDR			PRTADDR										
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x628	OTG_HS_HCINT9	Reserved																					DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x644	OTG_HS_HCSPLT10	SPLITEN	Reserved														COMPLSPLT	XACTPOS		HUBADDR			PRTADDR										
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x648	OTG_HS_HCINT10	Reserved																					DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x664	OTG_HS_HCSPLT11	SPLITEN	Reserved														COMPLSPLT	XACTPOS		HUBADDR			PRTADDR										
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x668	OTG_HS_HCINT11	Reserved																					DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x50C	OTG_HS_HCINTMSK0	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	AHBERRM	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x52C	OTG_HS_HCINTMSK1	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	AHBERRM	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x54C	OTG_HS_HCINTMSK2	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	AHBERRM	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x56C	OTG_HS_HCINTMSK3	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	AHBERRM	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x58C	OTG_HS_HCINTMSK4	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	AHBERRM	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x5AC	OTG_HS_HCINTMSK5	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	AHBERRM	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x5CC	OTG_HS_HCINTMSK6	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	AHBERRM	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x5EC	OTG_HS_HCINTMSK7	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	AHBERRM	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x60C	OTG_HS_HCINTMSK8	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	AHBERRM	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x62C	OTG_HS_HCINTMSK9	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	AHBERRM	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x64C	OTG_HS_HCINTMSK10	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	AHBERRM	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x66C	OTG_HS_HCINTMSK11	Reserved																					DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	AHBERRM	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x510	OTG_HS_HCTSIZ0	DOPING	DPID		PKTCNT										XFRSIZ																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x530	OTG_HS_HCTSIZ1	DOPING	DPID		PKTCNT										XFRSIZ																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x550	OTG_HS_HCTSIZ2	DOPING	DPID		PKTCNT										XFRSIZ																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x570	OTG_HS_HCTSIZ3	DOPING	DPID		PKTCNT										XFRSIZ																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x590	OTG_HS_HCTSIZ4	DOPING	DPID		PKTCNT										XFRSIZ																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5B0	OTG_HS_HCTSIZ5	DOPING	DPID		PKTCNT										XFRSIZ																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5D0	OTG_HS_HCTSIZ6	DOPING	DPID		PKTCNT										XFRSIZ																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5F0	OTG_HS_HCTSIZ7	DOPING	DPID		PKTCNT										XFRSIZ																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x610	OTG_HS_HCTSIZ8	DOPING	DPID		PKTCNT										XFRSIZ																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x630	OTG_HS_HCTSIZ9	DOPING	DPID		PKTCNT										XFRSIZ																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x650	OTG_HS_HCTSIZ10	DOPING	DPID		PKTCNT										XFRSIZ																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x670	OTG_HS_HCTSIZ11	DOPING	DPID		PKTCNT										XFRSIZ																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x514	OTG_HS_HCDMA0	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x524	OTG_HS_HCDMA1	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x544	OTG_HS_HCDMA2	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x564	OTG_HS_HCDMA3	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x584	OTG_HS_HCDMA4	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5A4	OTG_HS_HCDMA5	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x5C4	OTG_HS_HCDMA6	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5E4	OTG_HS_HCDMA7	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x604	OTG_HS_HCDMA8	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x624	OTG_HS_HCDMA9	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x644	OTG_HS_HCDMA10	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x664	OTG_HS_HCDMA11	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x800	OTG_HS_DCFG	Reserved						PERSCHIVL		Reserved	Reserved										PFIVL		DAD							Reserved	NZLSOHSK	DSPD	
	Reset value							1	0												0	0	0	0	0	0	0	0	0		0	0	0
0x804	OTG_HS_DCTL	Reserved																				POPRGDNE	CGONAK	SGONAK	CGINAK	SGINAK	TCTL			GONSTS	GINSTS	SDIS	RWUSIG
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x808	OTG_HS_DSTS	Reserved										FNSOF														Reserved				EERR	ENUMSPD		SUSPSTS
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0
0x810	OTG_HS_DIEPMSK	Reserved																		NAKM	Reserved					TXFURM	Reserved	INEPNEM	INEPNMM	ITTXFEMSK	TOM	AHBERRM	EPDM	XFRCM
	Reset value																			0						0	0	0	0	0	0	0	0	0
0x814	OTG_HS_DOEPMSK	Reserved																	NYETMSK	NAKMSK	BERRM	Reserved				OPEM	Reserved	B2BSTUP	STSPHSRXM	OTEPDM	STUPM	AHBERRM	EPDM	XFRCM
	Reset value																		0	0	0					0		0	0	0	0	0	0	0
0x818	OTG_HS_DAINT	OEPINT																IEPINT															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x81C	OTG_HS_DAINTMSK	OEPM																IEPM															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x828	OTG_HS_DVBUSDIS	Reserved																VBUSDT															
	Reset value																	0	0	0	1	0	1	1	1	1	1	0	1	0	1	1	1

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x82C	OTG_HS_DVBUSPULSE	Reserved																				DVBUSP											
	Reset value																					0	1	0	1	1	0	1	1	1	0	0	0
0x830	OTG_HS_DTHRCTL	Reserved				ARPEN	Reserved	RXTHRLEN									RXTHREN	Reserved				TXTHRLEN										ISOTHREN	NONISOTHREN
	Reset value					0		0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x834	OTG_HS_DIEPEMPMSK	Reserved																INEPTXFEM															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x838	OTG_HS_DEACHINT	Reserved																Reserved															Reserved
	Reset value															0																0	
0x83C	OTG_HS_DEACHINTMSK	Reserved																Reserved															Reserved
	Reset value															0																0	
0x844	OTG_HS_DIEPEACHMSK1	Reserved																	NAKM	Reserved	BIM	TXFURM	Reserved	INEPNEM	INEPNMM	ITTXFEMSK	TOM	AHBERRM	EPDM	XFRCM		
	Reset value																		0		0	0		0	0	0	0	0	0	0		
0x884	OTG_HS_DOEPEACHMSK1	Reserved																	NYETM	NAKM	BERRM	Reserved	BIM	TXFURM	Reserved	INEPNEM	INEPNMM	ITTXFEMSK	TOM	AHBERRM	EPDM	XFRCM		
	Reset value																		0	0	0		0	0		0	0	0	0	0	0	0		
0x900	OTG_HS_DIEPCTL0	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				STALL	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0x918	TG_FS_DTXFSTS0	Reserved																INEPTFSAV															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x920	OTG_HS_DIEPCTL1	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				Stall	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0x938	TG_FS_DTXFSTS1	Reserved																INEPTFSAV															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x940	OTG_HS_DIEPCTL2	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				Stall	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0x958	TG_FS_DTXFSTS2	Reserved																INEPTFSAV															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x960	OTG_HS_DIEPCTL3	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				Stall	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0x978	TG_FS_DTXFSTS3	Reserved																INEPTFSAV															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x980	OTG_HS_DIEPCTL4	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				Stall	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0x9A0	OTG_HS_DIEPCTL5	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				STALL	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0x9C0	OTG_HS_DIEPCTL6	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				STALL	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0x9E0	OTG_HS_DIEPCTL7	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	TXFNUM				STALL	Reserved	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0xB00	OTG_HS_DOEPCTL0	EPENA	EPDIS	Reserved		SNAK	CNAK	Reserved				STALL	SNPM	EPTYP		NAKSTS	Reserved	USBAEP	Reserved													MPSIZ	
	Reset value	0	0			0	0					0	0	0	0	0		1														0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xB20	OTG_HS_DOEPCTL1	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	Reserved				STALL	SNPM	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0xB40	OTG_HS_DOEPCTL2	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	Reserved				Stall	SNPM	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0xB60	OTG_HS_DOEPCTL3	EPENA	EPDIS	SODDFRM	SD0PID/SEVNFRM	SNAK	CNAK	Reserved				Stall	SNPM	EPTYP		NAKSTS	EONUM/DPID	USBAEP	Reserved				MPSIZ										
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0x908	OTG_HS_DIEPINT0	Reserved																		NAK	Reserved	PKTDRPSTS	Reserved		TXFIFOUDRN	TXFE	INEPNE	INEPNM	ITTXFE	TOC	AHBERR	EPDISD	XFRC
	Reset value																			0		0			0	1	0	0	0	0	0	0	0
0x928	OTG_HS_DIEPINT1	Reserved																		NAK	Reserved	PKTDRPSTS	Reserved		TXFIFOUDRN	TXFE	INEPNE	INEPNM	ITTXFE	TOC	AHBERR	EPDISD	XFRC
	Reset value																			0		0			0	1	0	0	0	0	0	0	0
0x948	OTG_HS_DIEPINT2	Reserved																		NAK	BERR	PKTDRPSTS	Reserved		TXFIFOUDRN	TXFE	INEPNE	INEPNM	ITTXFE	TOC	AHBERR	EPDISD	XFRC
	Reset value																			0		0			0	1	0	0	0	0	0	0	0
0x968	OTG_HS_DIEPINT3	Reserved																		NAK	Reserved	PKTDRPSTS	Reserved		TXFIFOUDRN	TXFE	INEPNE	INEPNM	ITTXFE	TOC	AHBERR	EPDISD	XFRC
	Reset value																			0	0	0			0	1	0	0	0	0	0	0	0
0x988	OTG_HS_DIEPINT4	Reserved																		NAK	Reserved	PKTDRPSTS	Reserved		TXFIFOUDRN	TXFE	INEPNE	INEPNM	ITTXFE	TOC	AHBERR	EPDISD	XFRC
	Reset value																			0		0			0	1	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x9A8	OTG_HS_DIEPINT5	Reserved																		NAK	Reserved	PKTDRPSTS	Reserved		TXFIFOUDRN	TXFE	INEPNE	INEPNM	ITTXFE	TOC	AHBERR	EPDISD	XFRC
	Reset value																			0		0			0	1	0	0	0	0	0	0	0
0x9C8	OTG_HS_DIEPINT6	Reserved																		NAK	Reserved	PKTDRPSTS	Reserved		TXFIFOUDRN	TXFE	INEPNE	INEPNM	ITTXFE	TOC	AHBERR	EPDISD	XFRC
	Reset value																			0		0			0	1	0	0	0	0	0	0	0
0x9E8	OTG_HS_DIEPINT7	Reserved																		NAK	Reserved	PKTDRPSTS	Reserved		TXFIFOUDRN	TXFE	INEPNE	INEPNM	ITTXFE	TOC	AHBERR	EPDISD	XFRC
	Reset value																			0		0			0	1	0	0	0	0	0	0	0
0xB08	OTG_HS_DOEPINT0	Reserved																	NYET	NAK	BERR	Reserved			OUTPKTERR	Reserved	B2BSTUP	Reserved	OTEPDIS	STUP	AHBERR	EPDISD	XFRC
	Reset value																		0	0	0				0		0		0	1	0	0	0
0xB28	OTG_HS_DOEPINT1	Reserved																	NYET	NAK	BERR	Reserved			OUTPKTERR	Reserved	B2BSTUP	Reserved	OTEPDIS	STUP	AHBERR	EPDISD	XFRC
	Reset value																		0	0	0				0		0		0	0	0	0	0
0xB48	OTG_HS_DOEPINT2	Reserved																	NYET	NAK	BERR	Reserved			OUTPKTERR	Reserved	B2BSTUP	Reserved	OTEPDIS	STUP	AHBERR	EPDISD	XFRC
	Reset value																		0	0	0				0		0		0	0	0	0	0
0xB68	OTG_HS_DOEPINT3	Reserved																	NYET	NAK	BERR	Reserved			OUTPKTERR	Reserved	B2BSTUP	Reserved	OTEPDIS	STUP	AHBERR	EPDISD	XFRC
	Reset value																		0	0	0				0		0		0	0	0	0	0
0xB88	OTG_HS_DOEPINT4	Reserved																	NYET	NAK	BERR	Reserved			OUTPKTERR	Reserved	B2BSTUP	Reserved	OTEPDIS	STUP	AHBERR	EPDISD	XFRC
	Reset value																		0	0	0				0		0		0	0	0	0	0
0xBA8	OTG_HS_DOEPINT5	Reserved																	NYET	NAK	BERR	Reserved			OUTPKTERR	Reserved	B2BSTUP	Reserved	OTEPDIS	STUP	AHBERR	EPDISD	XFRC
	Reset value																		0	0	0				0		0		0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xBC8	OTG_HS_DOEPINT6	Reserved																	NYET	NAK	BERR	Reserved			OUTPKTERR	Reserved	B2BSTUP	Reserved	OTEPDIS	STUP	AHBERR	EPDISD	XFRC
	Reset value																		0	0	0				0		0		0	0	0	0	0
0xBE8	OTG_HS_DOEPINT7	Reserved																	NYET	NAK	BERR	Reserved			OUTPKTERR	Reserved	B2BSTUP	Reserved	OTEPDIS	STUP	AHBERR	EPDISD	XFRC
	Reset value																		0	0	0				0		0		0	0	0	0	0
0x910	OTG_HS_DIEPTSIZ0	Reserved												PKTCNT	Reserved												XFRSIZ						
	Reset value													0	0												0	0	0	0	0	0	0
0x930	OTG_HS_DIEPTSIZ1	Reserved	MCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x934	OTG_HS_DIEPDMA1	DMAADDR																															
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x93C	OTG_HS_DIEPDMAB1	DMABADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x950	OTG_HS_DIEPTSIZ2	Reserved	MCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x954	OTG_HS_DIEPDMA2	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x95C	OTG_HS_DIEPDMAB2	DMABADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x970	OTG_HS_DIEPTSIZ3	Reserved	MCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x974	OTG_HS_DIEPDMA3	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x97C	OTG_HS_DIEPDMAB3	DMABADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB10	OTG_HS_DOEPTSIZ0	Reserved	STUPCNT		Reserved									PKTCNT	Reserved												XFRSIZ						
	Reset value		0	0										0													0	0	0	0	0	0	0
0xB30	OTG_HS_DOEPTSIZ1	Reserved	RXDPID/STUPCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB34	OTG_HS_DOEPDMA1	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xB3C	OTG_HS_DOEPDMAB1	DMABADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB50	OTG_HS_DOEPTSIZ2	Reserved	RXDPID/STUPCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB54	OTG_HS_DOEPDMA2	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB5C	OTG_HS_DOEPDMAB2	DMABADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB70	OTG_HS_DOEPTSIZ3	Reserved	RXDPID/STUPCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB74	OTG_HS_DOEPDMA3	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB7C	OTG_HS_DOEPDMAB3	DMABADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xE00	OTG_HS_PCGCCTL	Reserved																											PHYSUSP	Reserved		GATEHCLK	STPPCLK
	Reset value																																

35.13. Программная модель OTG_HS

35.13.1. Инициализация ядра

Сторона подключения кабеля (A или B) и режим контроллера показывается битом CMOD в регистре OTG_HS_GINTSTS).

Независимо от режимов хоста или устройства сначала пишем глобальные регистры:

1. В регистре OTG_HS_GAHBCFG:
   - Бит режима DMA
   - Длину пакета AHB
   - Бит маски глобального прерывания GINTMSK = 1
   - RxFIFO не пуст (бит RXFLVL в OTG_HS_GINTSTS)
   - Уровень пустоты периодического TxFIFO
2. В регистре OTG_HS_GUSBCFG:
   - Бит разрешения HNP
   - Бит разрешения SRP
   - Поле калибровки таймаута FS
   - Поле времени обратной связи USB
3. Разрешаем маски в регистре OTG_HS_GINTMSK:
   - Маска прерывания OTG
   - Маска прерывания несовпадения режимов
4. По биту CMOD в OTG_HS_GINTSTS определяем режим контроллера OTG_HS (хост или устройство).

35.13.2. Инициализация хоста

Исполняем следующие шаги:

1. Демаскируем HPRTINT в регистре OTG_FS_GINTMSK
2. В регистре OTG_FS_HCFG ставим полную скорость
3. Битом PPWR в OTG_FS_HPRT включаем V_{BUS} на USB.
4. Ждём подключения устройства к хосту по прерыванию PCDET в OTG_FS_HPRT0.
5. Запускаем сброс битом PRST в регистре OTG_HS_HPRT.
6. Не меньше 10 ms ждём его завершения.
7. Снимаем бит PRST в регистре OTG_HS_HPRT.
8. Ждём прерывания PENCHNG в регистре OTG_HS_HPRT.
9. Узнаём скорость переучёта по биту PSPD в регистре OTG_HS_HPRT.
10. В регистр HFIR пишем нужное число для 1 такта PHY.
11. В соответствии с полученной скоростью устройства пишем поле FSLSPCS регистра OTG_HS_HCFG. Если оно изменилось, то надо сбросить порт.
12. В регистр OTG_HS_GRXFSIZ пишем размер приёмного FIFO.
13. В регистр OTG_HS_HNPTXFSIZ пишем размер и адрес начала не-периодического передающего FIFO.
14. В регистр OTG_HS_HNPTXFSIZ пишем размер и адрес начала периодического передающего FIFO.

Для работы с устройствами нужно инициализировать хоть один канал.

35.13.3. Инициализация устройства

После сброса по включению питания или изменении режима на устройство надо:

1. В регистре OTG_HS_DCFG определить:
   - Скорость устройства
   - Состояние ответа OUT ненулевой длины.
2. В регистре OTG_HS_GINTMSK демаскировать прерывания:
   - Сброс USB
   - Переучёт завершён
   - Ранний останов
   - Останов USB
   - SOF
3. Битом VBUSBSEN в OTG_HS_GCCFG разрешить контроль V_{BUS} устройства “B” и резистор подпорки линии DP.
4. Дождаться прерывания USBRST в регистре OTG_HS_GINTSTS. После получения этого прерывания сброс длится ещё 10 ms.

Ждём прерывания ENUMDNE в регистре OTG_HS_GINTSTS. Это конец сброса USB. Теперь по регистру OTG_HS_DSTS определяем скорость переучёта и инициализируем конечные точки.

Всё, устройство готово принимать пакеты SOF и выдавать управляющие передачи по точке 0.

35.13.4. Режим DMA

Для передач AHB <=> USB хост OTG всегда использует ведущий интерфейс AHB, который пользуется адресом DMA (регистр HCDMAx для хоста и DIEPDMAx/DOEPDMAx для устройства).

35.13.5. Программная модель хоста

Инициализация канала

Выполняем шаги:

1. В регистре OTG_HS_GINTMSK демаскируем:
2. Прерывания канала

- Пустого не-периодического передающего FIFO передач OUT или
- Полупустого не-периодического передающего FIFO передач OUT.

3. В регистре OTG_HS_GINTMSK демаскируем прерывания выбранных каналов.
4. В регистре OTG_HS_GINTMSK демаскируем прерывания нужных условий передач.
5. В регистры OTG_HS_HCTSIZx пишем общий размер передач и ожидаемое число пакетов, включая короткие. В поле PID пишем начальный PID (первой для OUT или ожидаемой от IN).
6. В регистры OTG_HS_HCSPLTx пишем адреса хаба и порта (только для разбитых передач).
7. В регистры HCDMAx пишем начальные адреса буферов
8. В регистре OTG_HS_HCCHARx выбранного канала пишем характеристики устройства, вроде типа, скорости, направления и т.д.

Канал можно включать только при готовности к передаче или приёму данных.

Остановка канала

Выключить канал можно установкой битов CHDIS и CHENA регистра OTG_HS_HCCHARx. Хост сливает стоящие запросы, не прерывая запущенные передачи, и выдаёт прерывание остановки канала. Перед выделением канала для другой работы надо дождаться прерывания CHH в OTG_HS_HCINTx.

При выключении канала в режиме DMA очередь запросов проверять не надо. Хост OTG_HS ищет место для выключенных запросов на стороне выключенного канала во время арбитража. Но при стоящем бите CHDIS и HCCHARx все выданные запросы из очереди изгоняются.

Перед выключением канала нужно убедиться в наличии хоть одной свободной строки в соответствующей очереди запросов (периодической или не-периодической). Запросы в очереди можно слить установкой бита CHDIS и очисткой бита CHENA регистра OTG_HS_HCCHARx.

Канал выключают при:

1. Прерывании XFRC OTG_HS_HCINTx во время не-периодической передачи IN или широкополосной прерывающей передачи IN (только ведомый)
2. Прерываниях STALL, TXERR, BBERR или DTERR регистра OTG_HS_HCINTx. При этом такой же канал должен уметь принимать прерывания DTERR, NAK, DATA, TXERR.
3. Прерывании DISCINT регистра OTG_HS_GINTSTS. (выключаются все каналы).
4. При прекращении передач до их нормального завершения.

Ping-протокол

При работе OTG_HS на высокой скорости с групповыми или управляющими (фаза данных и статуса) точками OUT должно инициировать Пинг-протокол.

Это надо при получении прерываний NAK/NYET/TXERR. Здесь HS_OTG прекращает передачи по этой точке, теряет все полученные и выбранные запросы OUT (из очереди) и сливает соответствующий FIFO.

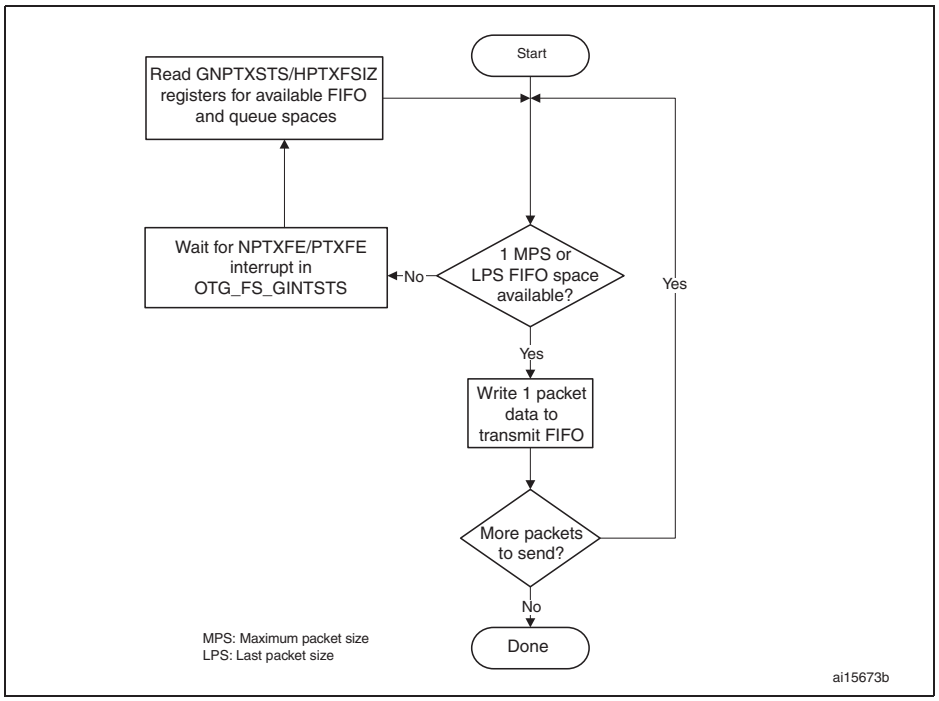
Так только в режиме ведомого. Тут можно послать токен Пинг установкой бита DOPING в HCTSIZx до включения канала или при его работе. Хост HS_OTG пишет запрос Пинг в очередь. И ждём прерывания с ответом (NAK, ACK или TXERR). Можно слать новый Пинг или продолжать передачи после получения ACK от точки OUT в ответ на пинг. В режиме DMA ставить бит DOPING в HCTSIZx при ответе NAK/NYET для групповых/управляющих OUT. Хост OTG_HS делает это автоматически вплоть до получения ответа ACK и переключается на передачу данных.

Модель работы

Всё делается с уже инициализированным каналом.

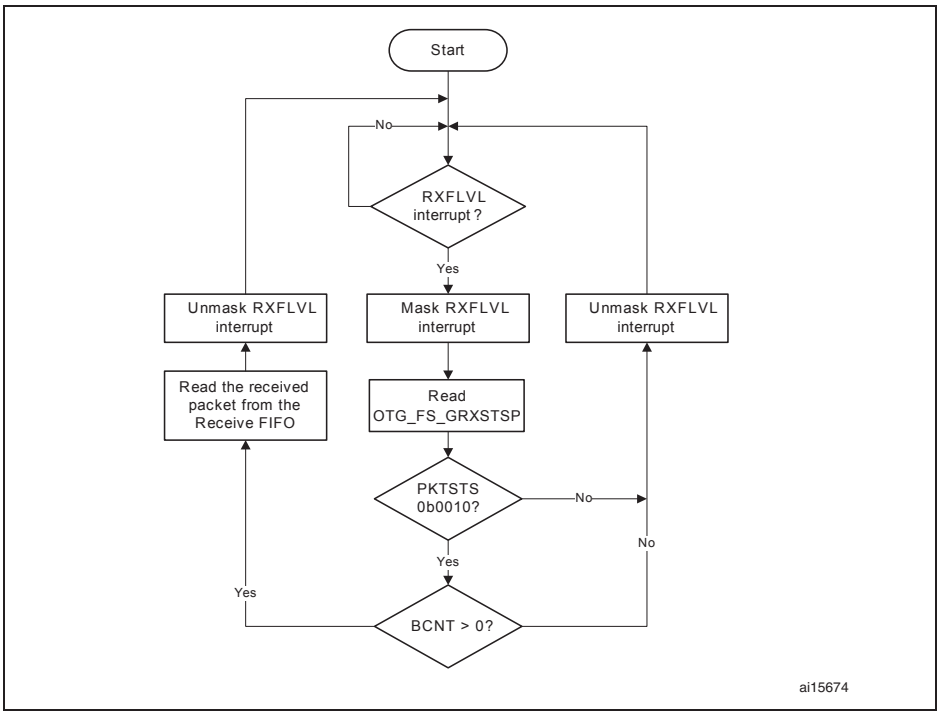
- **Запись в передающий FIFO**

Одновременно с записью последнего слова пакета, хост OTG_HS автоматически пишет запрос OUT в периодическую/не-периодическую очередь. Писать надо только при наличии свободного места в передающем FIFO. В FIFO пишут только словами. При некратной длине пакета он дополняется. Хост OTG_HS определяет конкретную длину пакета на основе установленных максимальной длины пакета и размера передачи.



• Чтение приёмного FIFO

Пакеты, отличные от пакетов данных IN (bx0010) должны игнорироваться.



• Групповые и управляющие передачи OUT/SETUP

На рисунке ниже передаются два пакета OUT и один пакет SETUP. Полагаем, что:

- Передаются два максимальных пакета (размер = 1024 байта).
- Передающий не-периодический FIFO может хранить два пакета (128 байт для FS).
- Глубина очереди не-периодических запросов = 4.

• Нормальные групповые и управляющие операции OUT/SETUP

Последовательность действий для канала 1 такова:

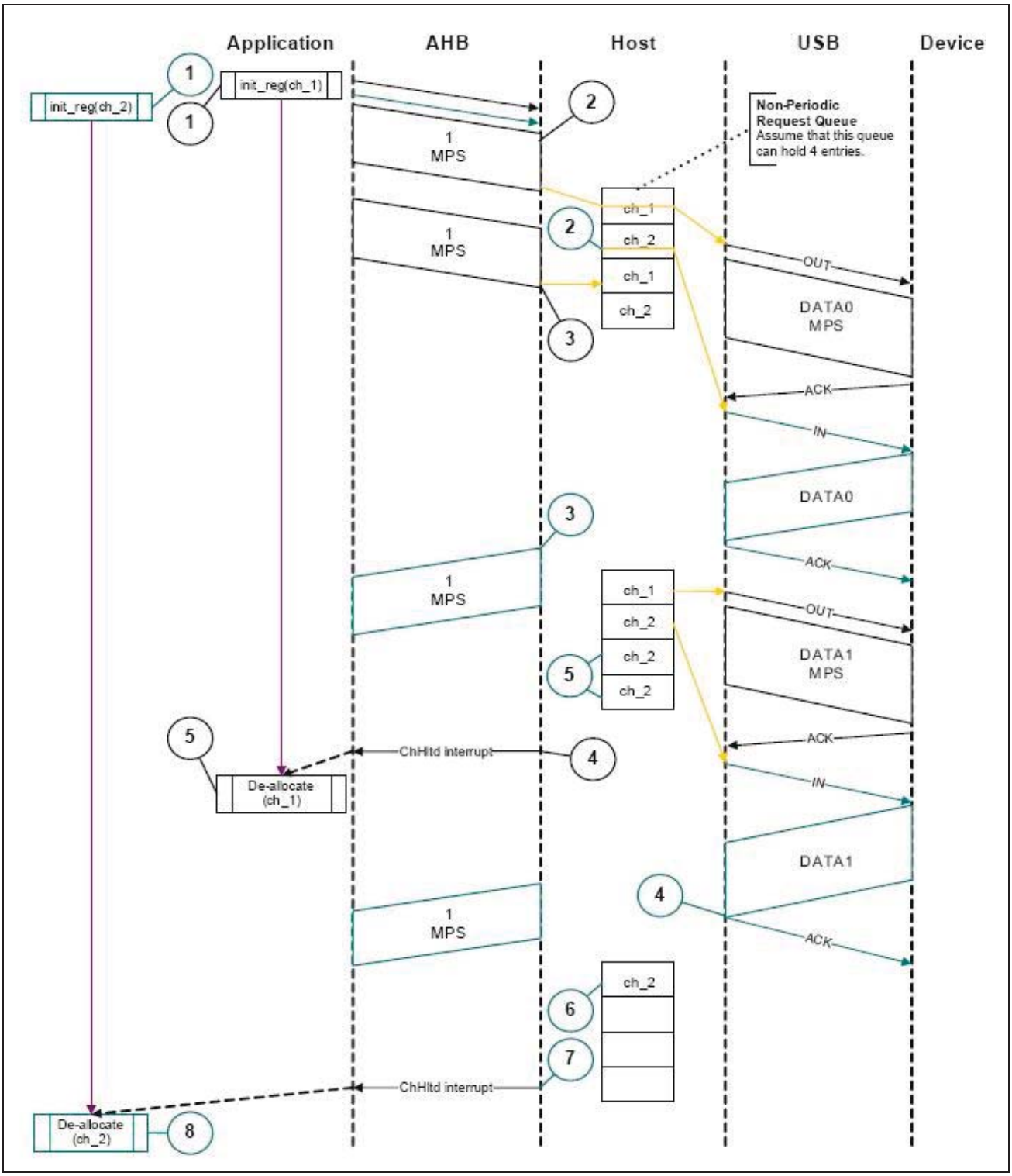
a) Инициализируем канал 1
b) Пишем первый пакет канала 1
c) Вместе с записью последнего слова хост пишет не-периодический запрос в очередь
d) После появления запроса в очереди ядро пытается послать токен OUT в текущем фрейме
e) Пишем второй (последний) пакет канала 1

f) После успешного завершения последней передачи выдаётся прерывание XFRC

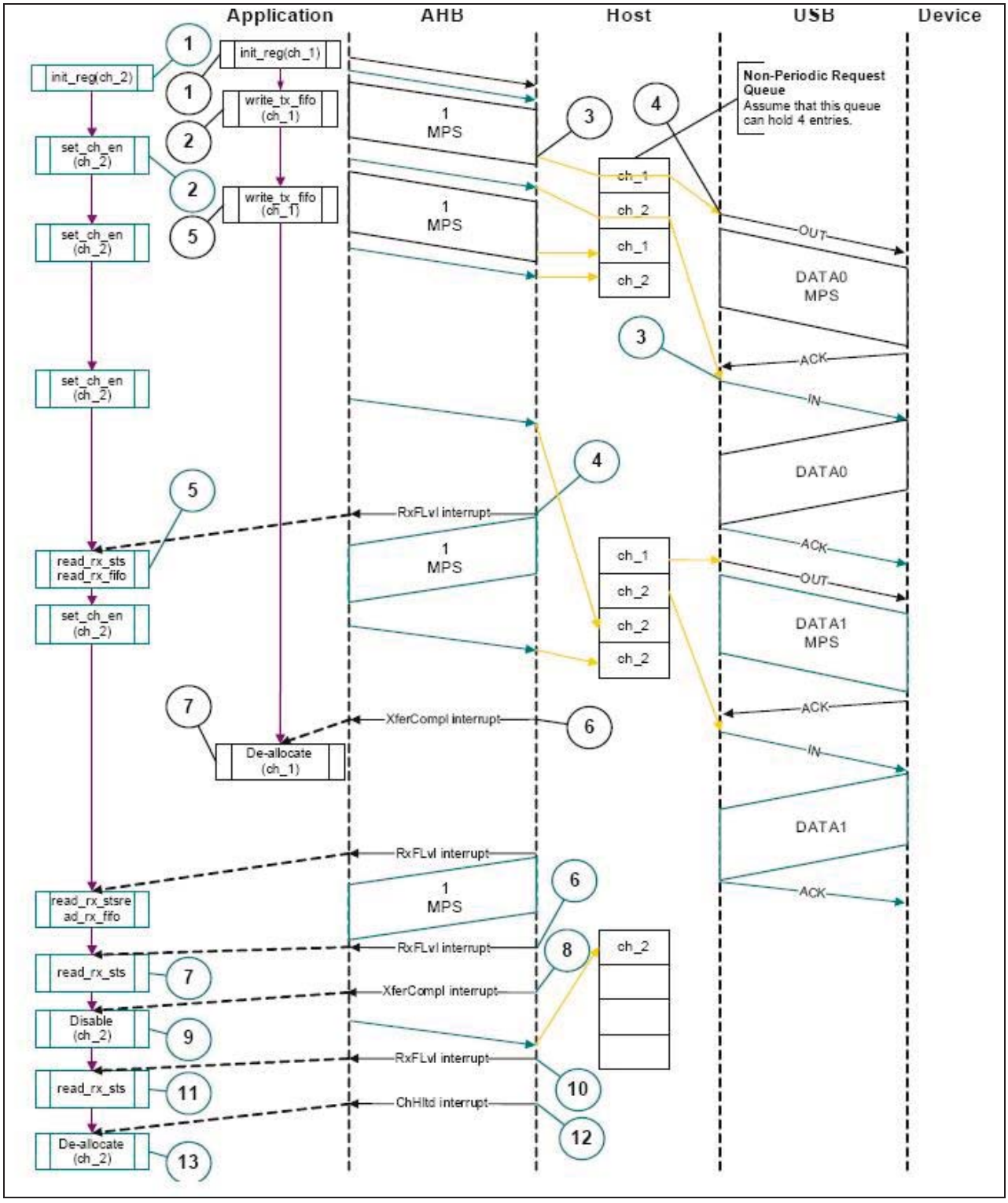
g) По прерыванию XFRC освобождает канал для других передач

h) Обрабатываем не-ACK ответы

Нормальные групповые/управляющие OUT/SETUP и групповые/управляющие IN - режим DMA



Нормальные групповые/управляющие OUT/SETUP и групповые/управляющие IN - режим Ведомого



• **Обработчик прерываний групповых/управляющих передач OUT/SETUP и IN**

а) Групповые/Управляющие OUT/SETUP

Наличие свободного места в FIFO и в очереди запросов определяют по прерыванию `NPTXFE` регистра `OTG_HS_GINTSTS`.

```
Unmask (NAK/TXERR/STALL/XFRC)
if (XFRC) {
    Reset Error Count
    Mask ACK
    De-allocate Channel
    }
else if (STALL)    {
```

```
        Transfer Done = 1
        Unmask CHH
        Disable Channel
}
    else if (NAK or TXERR ) {
        Rewind Buffer Pointers
        Unmask CHH
        Disable Channel
if (TXERR)   {
            Increment Error Count
            Unmask ACK
    }
        else {
            Reset Error Count
            }
        }
    else if (CHH){
        Mask CHH
        if (Transfer Done or (Error_count == 3)) {
            De-allocate Channel
            }
        else  {
            Re-initialize Channel
            }
        }
    else if (ACK) {
        Reset Error Count
        Mask ACK
        }
```

b) Групповые/Управляющие IN

Запросы выдаются при наличии свободного места в очереди.

```
    Unmask (TXERR/XFRC/BBERR/STALL/DTERR)
    if (XFRC){
        Reset Error Count
        Unmask CHH
        Disable Channel
        Reset Error Count
        Mask ACK
}
    else if (TXERR or BBERR or STALL) {
        Unmask CHH
        Disable Channel
        if (TXERR) {
            Increment Error Count
            Unmask ACK
            }
        }
    else if (CHH) {
        Mask CHH
        if (Transfer Done or (Error_count == 3)) {
            De-allocate Channel
            }
        else {
            Re-initialize Channel
            }
        }
    else if (ACK) {
        Reset Error Count
        Mask ACK
}
    else if (DTERR) {
        Reset Error Count
        }
```

• Групповые и управляющие передачи IN

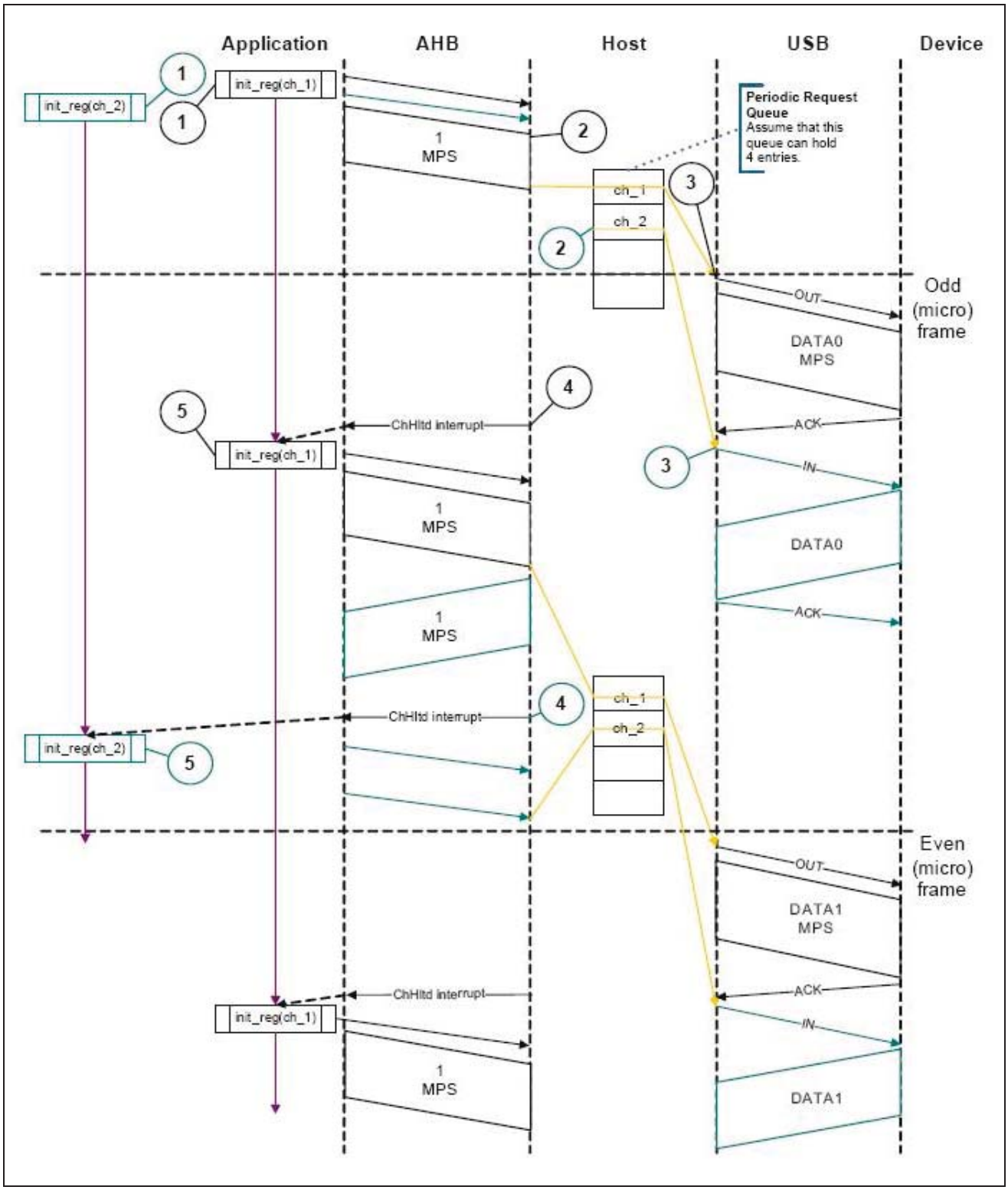
На рисунке ниже принимаем групповые или управляющие передачи IN по каналу 2. Полагаем, что:

– Принимаются два максимальных пакета (размер = 1024 байта).

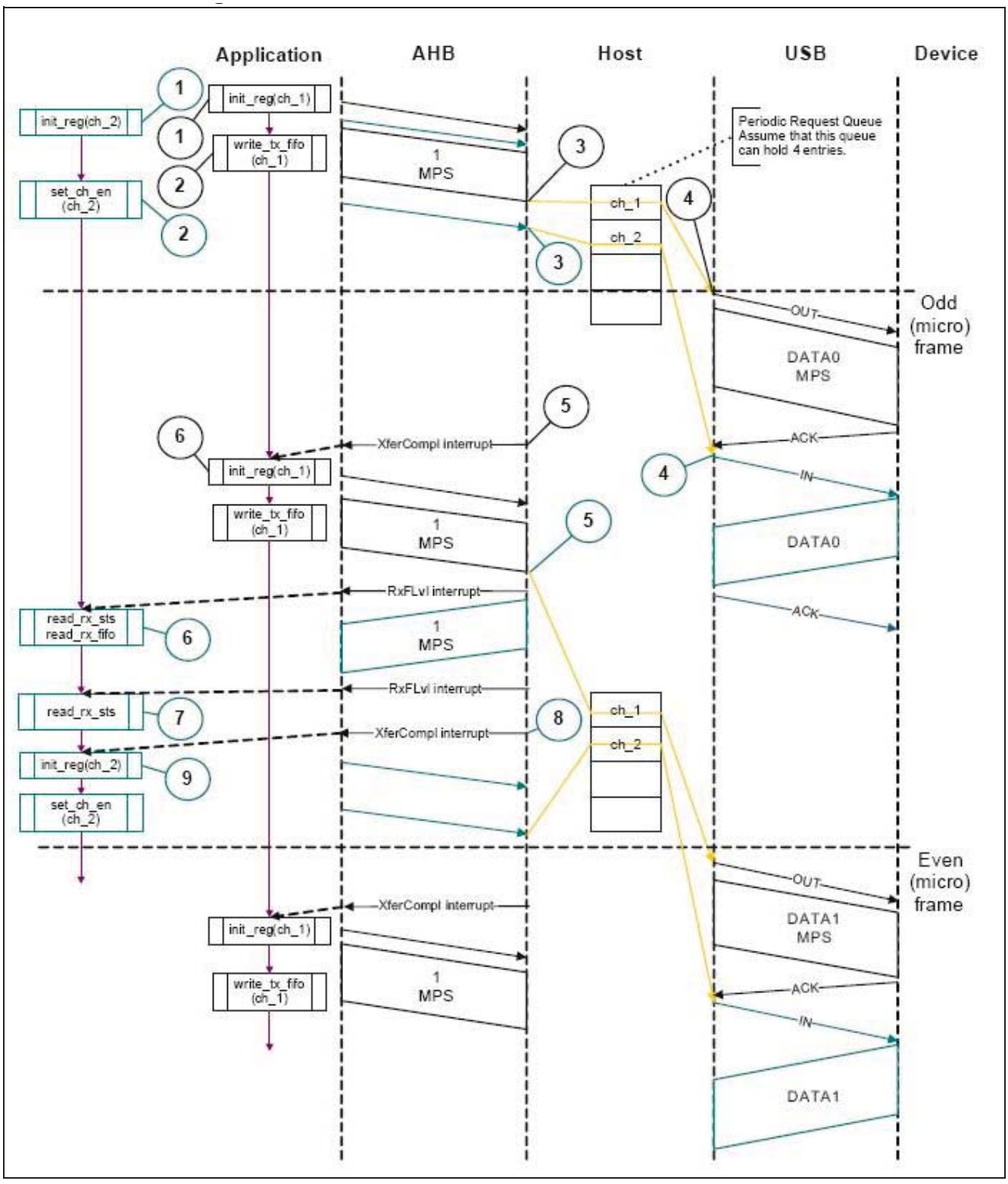
– Приёмный FIFO может хранить хоть один максимальный пакет и два слова состояния на пакет (72 байта для FS).

– Глубина не-периодической очереди = 4.

Групповые и управляющие передачи IN - режим DMA



Групповые и управляющие передачи IN - режим ведомого



Последовательность операций такова:

a) Инициализируем канал 2.

b) Ставим бит `CHENA` в `HCCHAR2` для записи запроса IN в не-периодическую очередь.

c) Ядро пробует послать токен IN после завершения текущей передачи OUT.

d) После приёма пакета в FIFO ядро выдаёт прерывание `RXFLVL`.

e) Приняв прерывание `RXFLVL` маскируем его, читаем число принятых байтов из состояния принятого пакета и читаем FIFO. Затем демаскируем прерывание `RXFLVL`.

f) Ядро выдаёт прерывание `RXFLVL` по каждой строке состояния конца приёма в FIFO.

g) Принятые не-IN пакеты данных (PKTSTS в GRXSTSR ≠ 0b0010) игнорируются.

h) Ядро выдаёт прерывание XFRC по каждому чтению состояния принятого пакета.

i) По прерыванию XFRC выключаем канал и стопорим запись дальнейших запросов в OTG_HS_HCCHAR2. Ядро пишет запрос выключения канала в не-периодическую очередь сразу после записи в регистр OTG_HS_HCCHAR2.

j) Ядро выдаёт прерывание RXFLVL сразу записи состояния останова в FIFO.

k) Читаем и игнорируем состояния приёма пакета.

l) Ядро выдаёт прерывание CHH сразу после извлечения состояния останова из FIFO.

m) По прерыванию CHH отдаём канал для новых операций.

n) Обрабатываем не-ACK ответы

Управляющие передачи в режиме ведомого

Управляющие передачи Setup, Data и Status передаются раздельно. Передача OUT Setup, Data или Status подобна групповой передаче OUT. Передача IN Data или Status подобна групповой передаче IN. Для всех трёх передач в поле EPTYP регистра OTG_FS_HCCHAR1 пишут "Управляющая". Для передач Setup в поле PID регистра OTG_FS_HCTSIZ1 пишут SETUP.

Прерывающие передачи OUT

Они показаны на рисунке ниже. Полагаем, что:

– Передаётся 1 пакет в каждом фрейме (до 1 максимального размера), начиная с нечётного фрейма (размер = 1024 байта)

– Периодический передающий FIFO может хранить один пакет (1 KB)

– Глубина периодической очереди равна = 4

Последовательность операций такова:

a) Инициализируем и включаем канал 1. Бит ODDFRM в OTG_FS_HCCHAR1 должен стоять.

b) Пишем первый пакет канала 1.

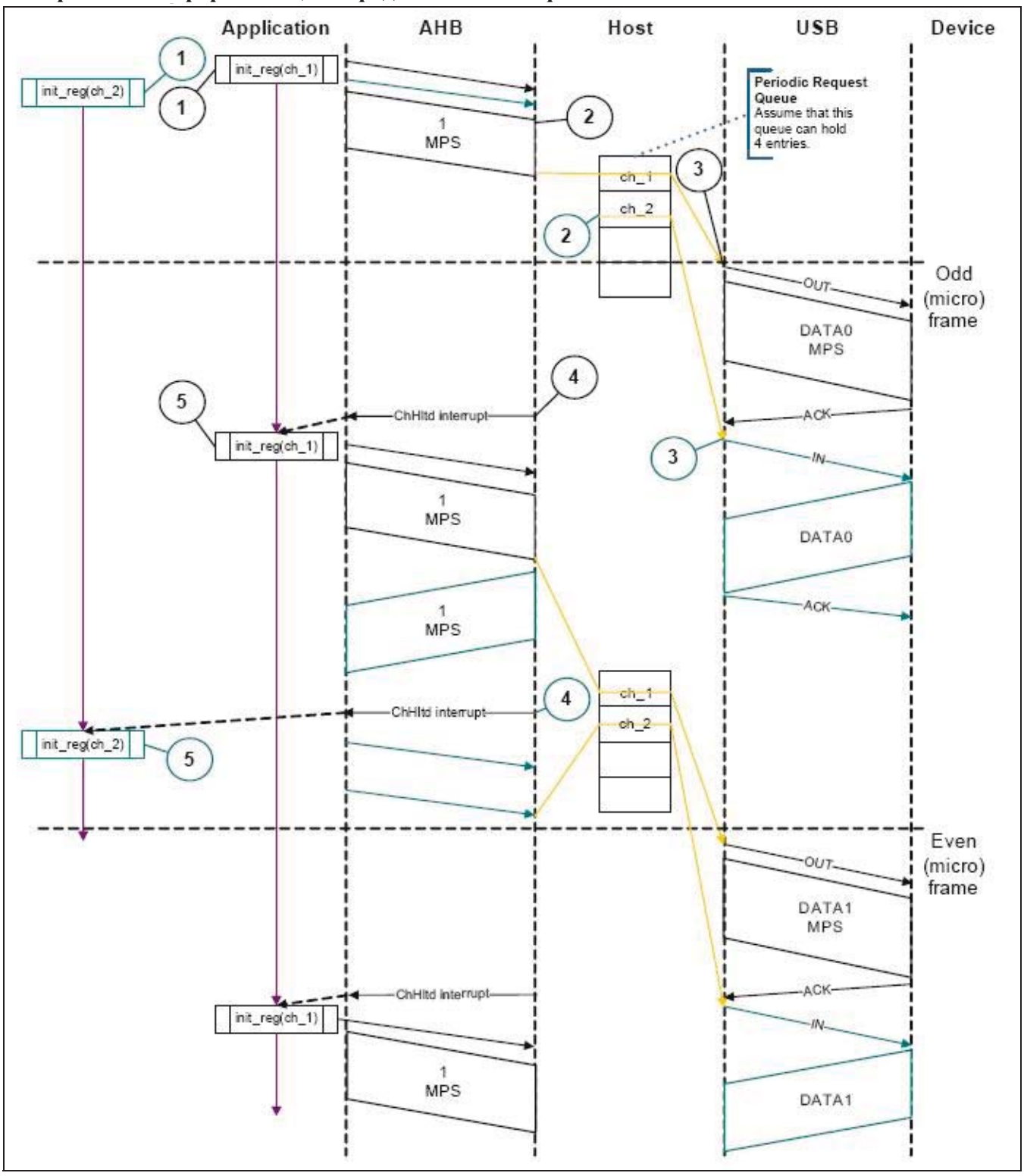
c) Вместе с записью последнего слова каждого пакета хост OTG_FS пишет запрос в периодическую очередь.

d) Хост OTG_FS пытается послать токен OUT в следующем (нечётном) фрейме.

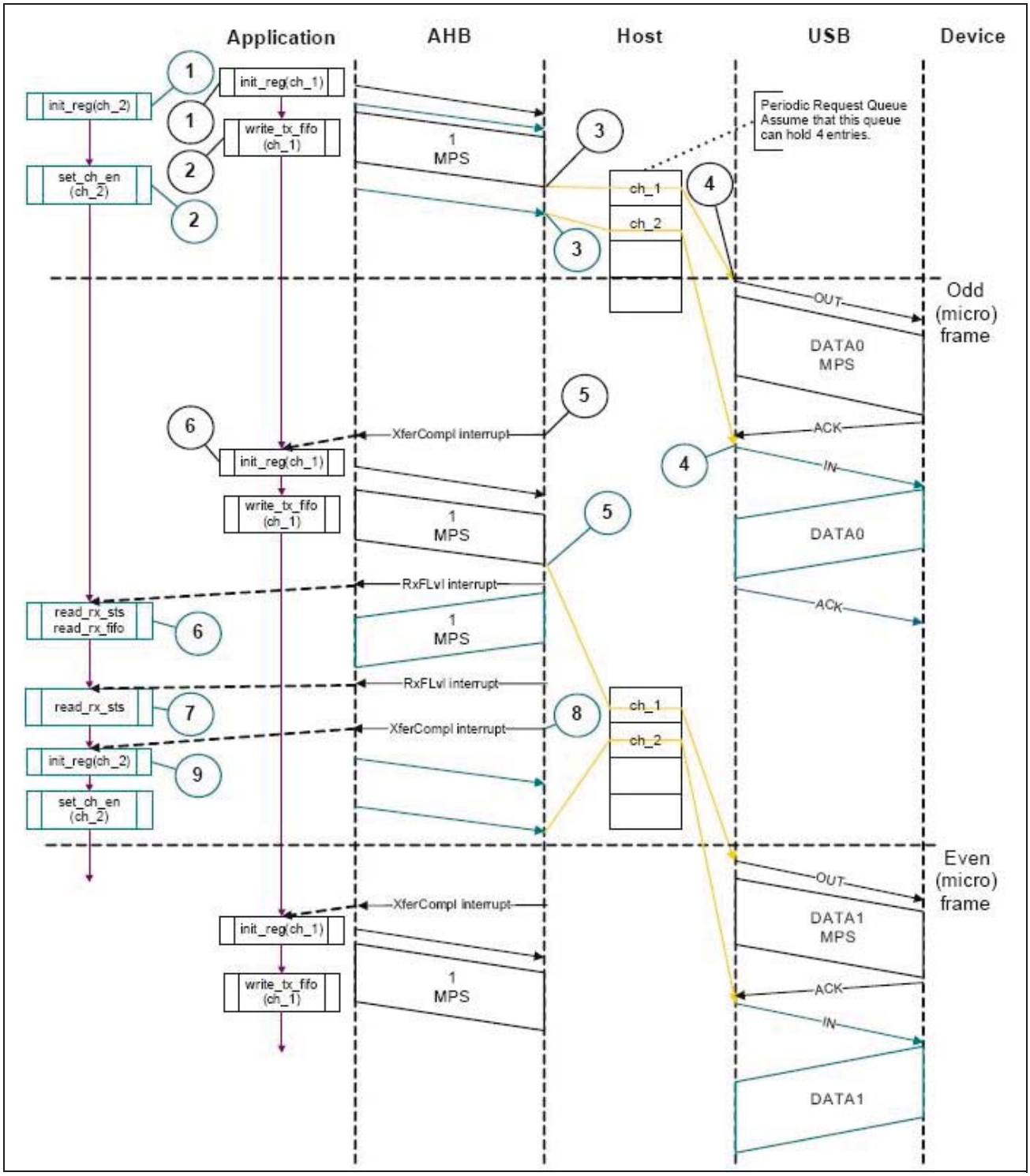
e) Прерывание XFRC выдаётся после успешной передачи последнего пакета.

f) После прерывания XFRC инициализируем канал для следующей передачи.

Нормальные прерывающие передачи OUT/IN - режим DMA



Нормальные прерывающие передачи OUT/IN - режим ведомого



• Программа обработки прерываний для прерывающих передач OUT/IN

а) Прерывающие OUT

Место в передающем FIFO определяется по прерыванию `NPTXFE` в `OTG_HS_GINTSTS`.

```
Unmask (NAK/TXERR/STALL/XFRC/FRMOR)
if (XFRC) {
    Reset Error Count
    Mask ACK
    De-allocate Channel
    }
else
    if (STALL or FRMOR) {
        Mask ACK
        Unmask CHH
        Disable Channel
```

```
        if (STALL) {
            Transfer Done = 1
            }
        }
    else
        if (NAK or TXERR) {
            Rewind Buffer Pointers
            Reset Error Count
            Mask ACK
    Unmask CHH
            Disable Channel
            }
        else
            if (CHH) {
            Mask CHH
            if (Transfer Done or (Error_count == 3)) {
                        De-allocate Channel
                        }
            else  {
                    Re-initialize Channel (in next b_interval - 1 Frame)
                    }
            }
        else
            if (ACK)  {
            Reset Error Count
            Mask ACK
            }
```

b) Прерывающие IN

```
Unmask (NAK/TXERR/XFRC/BBERR/STALL/FRMOR/DTERR)
if (XFRC) {
    Reset Error Count
    Mask ACK
    if (OTG_HS_HCTSIZx.PKTCNT == 0 {
        De-allocate Channel
        }
    else {
        Transfer Done = 1
        Unmask CHH
        Disable Channel
        }
    }
else
    if (STALL or FRMOR or NAK or DTERR or BBERR)  {
        Mask ACK
        Unmask CHH
        Disable Channel
        if (STALL or BBERR)  {
            Reset Error Count
            Transfer Done = 1
            }
        else
            if (!FRMOR) {
            Reset Error Count
            }
        }
else
    if (TXERR)  {
        Increment Error Count
        Unmask ACK
        Unmask CHH
        Disable Channel
        }
    else
        if (CHH)  {
            Mask CHH
```

```
            if (Transfer Done or (Error_count == 3)) {
             De-allocate Channel
            }
            else
             Re-initialize Channel (in next b_interval - 1 /Frame)
            }
        }
    else
        if (ACK) {
            Reset Error Count
            Mask ACK }
```

- **Прерывающие передачи IN**

Полагаем, что:

- Принимаем один пакет максимальной длины в каждом фрейме, начиная с нечётного (размер = 1024 байта).
- Приёмный FIFO может хранить один максимальный пакет и два слова состояния на один пакет (1031 байт).
- Глубина периодической очереди = 4.

- **Нормальная прерывающая операция IN**

Последовательность операций такова:

a) Инициализируем канал 2. Бит ODDFRM в OTG_HS_HCCHAR2 должен стоять.
b) Ставим бит CHENA в OTG_HS_HCCHAR2 для записи запроса IN в периодическую очередь.
c) Хост пишет запрос IN в очередь по каждой установке бита CHENA в OTG_HS_HCCHAR2.
d) Хост OTG_HS пробует послать токен IN в следующем (нечётном) фрейме.
e) После приёма пакета IN в FIFO, хост OTG_HS выдаёт прерывание RXFLVL.
f) По прерыванию RXFLVL в слове состояния узнаём число принятых байтов и читаем FIFO. Перед чтением FIFO прерывание RXFLVL надо маскировать и открывать его после извлечения всего пакета.
g) После получения в FIFO строки статуса выдаётся прерывание RXFLVL. Принятые не-IN пакеты данных (PKTSTS в GRXSTSR ≠ 0b0010) надо читать и игнорировать.
h) После чтения строки статуса выдаётся прерывание XFRC.
i) По прерыванию XFRC читаем поле PKTCNT регистра OTG_HS_HCTSIZ2. Если бит PKTCNT в OTG_HS_HCTSIZ2 стоит, то выключаем канал, иначе инициализируем канал для следующей передачи. При этом бит ODDFRM в OTG_HS_HCCHAR2 надо сбросить.

- **Изохронные передачи OUT**

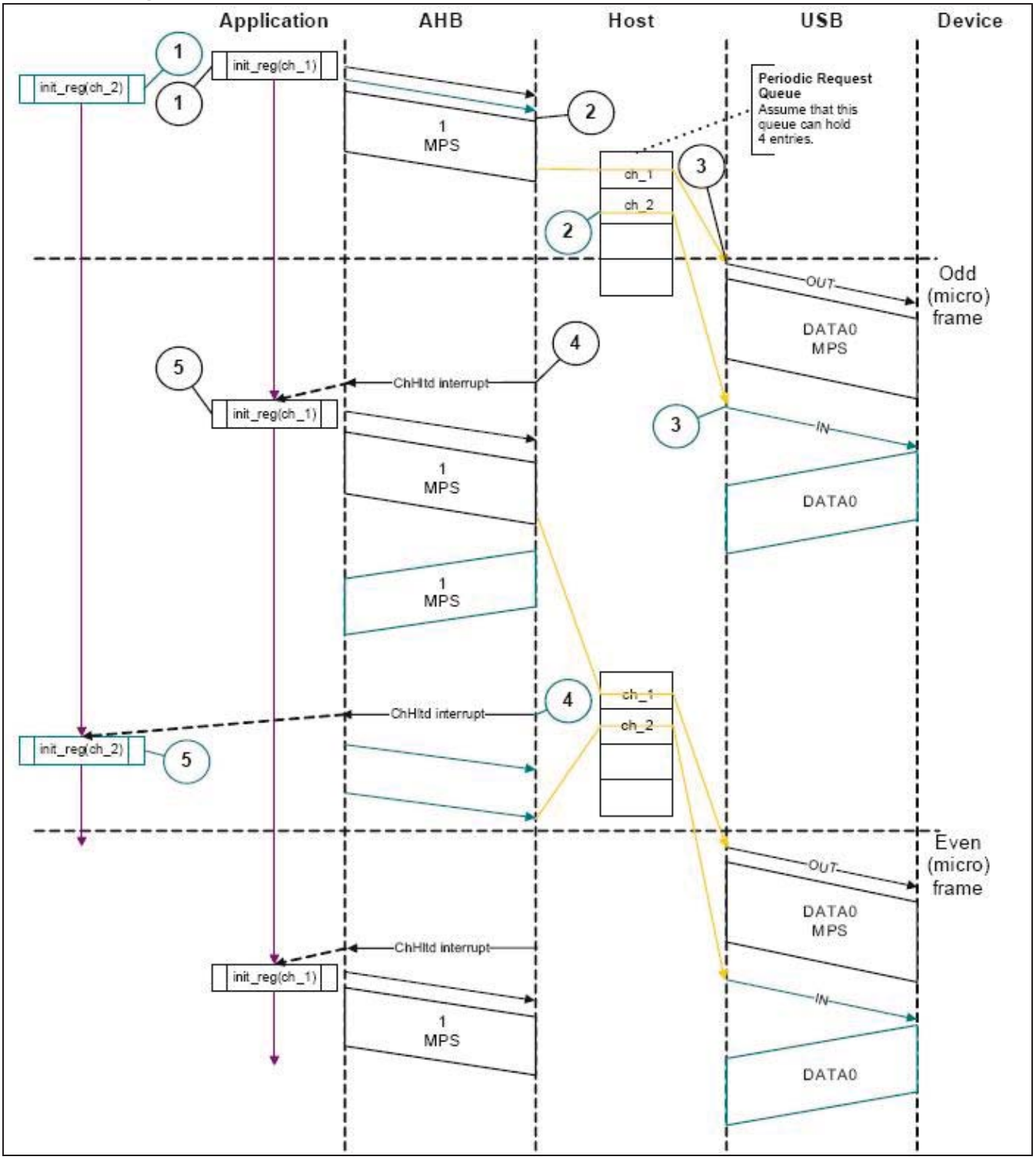
Пример показан на рисунке ниже. Полагаем, что:

- Посылается один пакет максимальной длины в каждом фрейме, начиная с нечётного (размер = 1024 байта).
- Периодический передающий FIFO может содержать один пакет (1 KB).
- Глубина периодической очереди = 4.

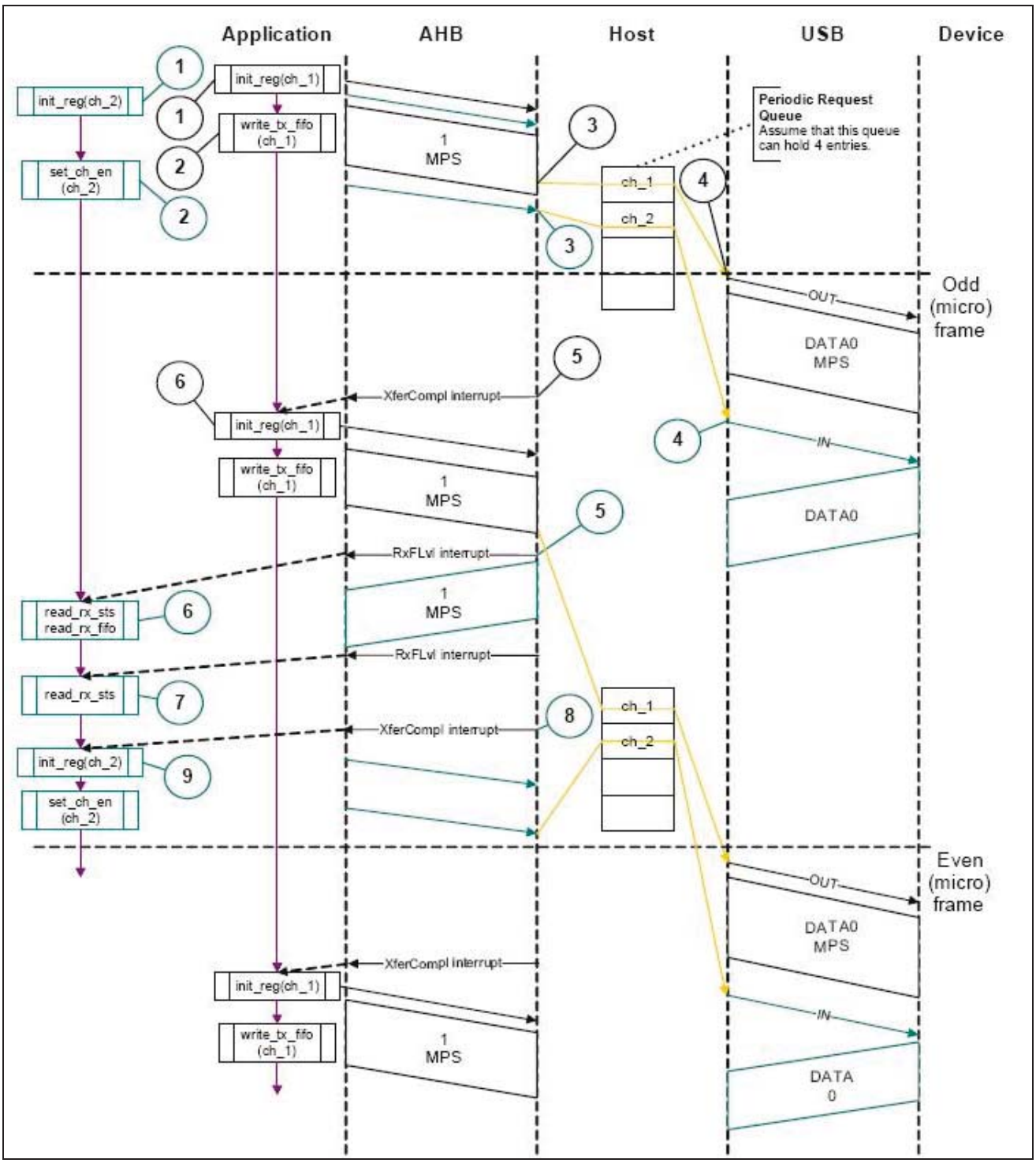
Последовательность операций такова:

a) Инициализируем канал 1. Бит ODDFRM в OTG_FS_HCCHAR1 должен стоять.
b) Пишем первый пакет канала 1.
c) Вместе с записью последнего слова пакета хост OTG_FS пишет запрос в периодическую очередь.
d) Хост OTG_FS пробует послать токен OUT в следующем (нечётном) фрейме.
e) Прерывание XFRC выдаётся после успешной передачи последнего пакета.
f) После него инициализируем канал для следующей передачи.
g) Обрабатываем не-ACK ответы.

Нормальные изохронные передачи OUT/IN - Режим DMA



Нормальные изохронные передачи OUT/IN - Режим ведомого



• **Программа обработки прерываний для изохронных передач OUT/IN**

Изохронные OUT

```
Unmask (FRMOR/XFRC)
if (XFRC) {
    De-allocate Channel
    }
else
    if (FRMOR)  {
        Unmask CHH
        Disable Channel
        }
    else
        if (CHH)  {
            Mask CHH
            De-allocate Channel
            }
```

Изохронные IN

```
Unmask (TXERR/XFRC/FRMOR/BBERR)
if (XFRC or FRMOR) {
    if (XFRC and (OTG_HS_HCTSIZx.PKTCNT == 0)) {
        Reset Error Count
        De-allocate Channel
        }
    else {
        Unmask CHH
        Disable Channel
        }
    }
    else
        if (TXERR or BBERR) {
            Increment Error Count
            Unmask CHH
            Disable Channel
        }
        else
    if (CHH) {
        Mask CHH
        if (Transfer Done or (Error_count == 3)) {
                    De-allocate Channel
                    }
            else {
             Re-initialize Channel
            }
            }
```

- **Изохронные передачи IN**

Полагаем, что:

- Принимаем один пакет максимальной длины в каждом фрейме, начиная с нечётного (размер = 1024 байта).
- Приёмный FIFO может содержать 1 максимальный пакет и два слова статуса на каждый пакет (1031 байт).
- Глубина периодической очереди = 4.

Последовательность операций такова:

a) Инициализируем канал 2. Бит ODDFRM в OTG_HS_HCCHAR2 должен стоять.

b) Ставим CHENA в OTG_HS_HCCHAR2 для записи запроса IN в периодическую очередь.

c) Запрос IN пишется в очередь по каждой установке бита CHENA в OTG_HS_HCCHAR2.

d) Хост пробует послать токен IN в следующем нечётном фрейме.

e) После приёма пакета IN в FIFO выдаётся прерывание RXFLVL.

f) По нему из состояния принятого пакета берём число полученных байтов и читаем FIFO. Перед чтением FIFO маскируем прерывание RXFLVL и открываем его после извлечения всего пакета.

g) Прерывание RXFLVL выдаётся после приёма строки состояния пакета в FIFO. Строки состояния не-IN пакетов данных (бит PKTSTS и OTG_HS_GRXSTSR ≠ 0b0010) читаются и игнорируются.

h) Прерывание XFRC выдаётся после чтения из FIFO строки состояния пакета.

i) По нему читаем поле PKTCNT регистра OTG_HS_HCTSIZ2. Если PKTCNT≠ 0, то выключаем канал, иначе инициализируем его для следующей передачи. В это время нужно сбросит бит ODDFRM в регистре OTG_HS_HCCHAR2.

- **Выбор глубины очереди**

Глубина периодической и не-периодической очередей должна соответствовать числу одновременно работающих соответствующих конечных точек.

Чем глубже не-периодической очереди (с соответствующим размером FIFO), тем чаще возможны не-периодические передачи.

Глубокая периодическая очередь позволяет планировать периодические передачи. Если глубина меньше запланированного числа периодических передач в микрофрейме, то возникает переполнение фрейма.

- **Обработка ошибки перекрёстных шумов**

Две типа ошибки: пакета и порта.

Ошибка пакета появляется когда устройство посылает данных больше, чем максимальный размер пакета канала. Ошибка порта появляется когда ядро продолжает принимать данные от устройства после EOF2 (конец фрейма 2, очень близкого к SOF).

Когда контроллер OTG_HS обнаруживает ошибку пакета, он перестаёт писать данные в буфер Rx и ждёт конца пакета (EOP). После обнаружения EOP он сливает уже записанные в Rx данные и выдаёт прерывание ошибки перекрёстных шумов.

Когда контроллер OTG_HS обнаруживает ошибку порта, он сливает RxFIFO и выключает порт. Выдаётся прерывание выключения порта (`HPRTINT` в `OTG_HS_GINTSTS`, `PENCHNG` в `OTG_HS_HPRT`). Программа по биту `POCA` in `OTG_HS_HPRT`, определяет не было ли это вызвано перегрузкой порта и выполняет программный сброс порта. После ошибки порта токены больше не посылаются.

- **Групповые и управляющие передачи OUT/SETUP в режиме DMA**

Последовательность такая:

a) Инициализируем и включаем канал как раньше.

b) Хост HS_OTG начинает загрузку первого пакета, используя заданный адрес DMA.

c) После выборки последнего слова второго (последнего) пакета хост OTG_HS маскирует канал 1 для дальнейшего арбитража.

d) После посылки последнего пакета хост HS_OTG выдаёт прерывание `CHH`.

e) В ответ на `CHH` мы освобождаем канал для будущих передач.

- **Обработка NAK и NYET с внутренним DMA**

Последовательность такая:

a) Хост OTG_HS посылает групповую передачу OUT.

b) Устройство отвечает `NAK` или `NYET`.

c) Если прерывания `NAK` или `NYET` демаскированы, то будет прерывание, но отвечать на него не надо, ядро само сбросит указатели буфера и инициализирует канал.

d) Ядро автоматически выдаёт токен Пинг.

e) После получения `ACK` ядро продолжит передачу. Конечно, прерывания `NAK` или `NYET` можно использовать, но это не наша забота.

Ядро не выдаёт отдельного прерывания, если `NAK` или `NYET` принимаются самим хостом.

- **Групповые и управляющие передачи IN в режиме DMA**

Последовательность такая:

a) Инициализируем и включаем канал как раньше.

b) После получения каналом гранта от арбитра (round-robin) хост OTG_HS ставит запрос IN в очередь.

c) После приёма последнего байта без ошибок хост OTG_HS начинает писать принятые данные в системную память.

d) После приёма последнего пакета хост OTG_HS ставит внутренний флаг для удаления лишних запросов IN из очереди.

e) Хост OTG_HS сливает лишние запросы.

f) Финальный запрос выключения канала x ставится в очередь. Тут канал 2 внутренне маскируется для дальнейшего арбитража.

g) При появлении наверху очереди запроса выключения, хост OTG_HS выдаёт прерывание `CHH`.

h) В ответ на `CHH` мы освобождаем канал для будущих передач.

- **Прерывающие передачи OUT в режиме DMA**

Последовательность такая:

a) Инициализируем и включаем канал как раньше.

b) После включения канала хост OTG_HS начинает выборку первого пакета и вместе в выборкой последнего слова пишет запрос OUT. При широкополосных передачах перед переключением на следующий канал хост HS_OTG продолжает грузить следующий пакет (до значения поля MC).

c) Хост OTG_HS пытается послать токен OUT в начале следующего (нечётного) фрейма/микрофрейма.

d) После успешной передачи пакета хост OTG_HS выдаёт прерывание CHH.

e) В ответ на CHH мы инициализируем канал для будущих передач.

- **Прерывающие передачи IN в режиме DMA**

Последовательность такая:

a) Инициализируем и включаем канал как раньше.

b) После получения каналом x гранта от арбитра (равноправный round-robin) хост OTG_HS ставит запрос IN в очередь. При широкополосных передачах хост OTG_HS последовательно пишет MC раз.

c) Хост OTG_HS пытается послать токен OUT в начале следующего (нечётного) фрейма/микрофрейма.

d) После записи принятого пакета в FIFO хост OTG_HS выдаёт прерывание CHH.

e) В ответ на CHH мы инициализируем канал для будущих передач.

- **Изохронные передачи OUT в режиме DMA**

Последовательность такая:

a) Инициализируем и включаем канал как раньше.

b) После включения канала хост OTG_HS начинает выборку первого пакета и вместе в выборкой последнего слова пишет запрос OUT. При широкополосных передачах перед переключением на следующий канал хост HS_OTG продолжает грузить следующий пакет (до значения поля MC).

c) Хост OTG_HS пытается послать токен OUT в начале следующего (нечётного) фрейма/микрофрейма.

d) После успешной передачи пакета хост OTG_HS выдаёт прерывание CHH.

e) В ответ на CHH мы инициализируем канал для будущих передач.

- **Изохронные передачи IN в режиме DMA**

Последовательность такая:

a) Инициализируем и включаем канал как раньше.

b) После получения каналом x гранта от арбитра (равноправный round-robin) хост OTG_HS ставит запрос IN в очередь. При широкополосных передачах хост OTG_HS последовательно пишет MC раз.

c) Хост OTG_HS пытается послать токен OUT в начале следующего (нечётного) фрейма/микрофрейма.

d) После записи принятого пакета в FIFO хост OTG_HS выдаёт прерывание CHH.

e) В ответ на CHH мы инициализируем канал для будущих передач.

- **Групповые и управляющие разделённые передачи OUT/SETUP в режиме DMA**

Последовательность такая:

a) Инициализируем и включаем канал как раньше.

b) После включения канала хост OTG_HS начинает выборку первого пакета и вместе в выборкой последнего слова пишет запрос OUT.

c) После успешной передачи стартовой части хост OTG_HS выдаёт прерывание CHH.

d) В ответ на CHH мы ставим бит COMPLSPLT в HCSPLT1 для посылки завершения.

e) После успешной передачи концевой части хост OTG_HS выдаёт прерывание CHH.

f) В ответ на CHH мы освобождаем канал.

- **Групповые и управляющие разделённые передачи IN в режиме DMA**

Последовательность такая:

a) Инициализируем и включаем канал как раньше.
b) После получения каналом x гранта от арбитра хост OTG_HS ставит запрос первой части IN в не-периодическую очередь и затем внутренне маскирует канал x для арбитража.
c) После передачи токена IN хост OTG_HS выдаёт прерывание CHH.
d) В ответ на CHH мы ставим бит COMPLSPLT в HCSPLT2 и разрешаем канал для посылки токена концевой части. Канал демаскируется для арбитража.
e) После получения гранта от арбитра хост OTG_HS ставит запрос концевой части IN в не-периодическую очередь.
f) После успешного приёма пакета хост OTG_HS начинает писать его в системную память.
g) После записи пакета в память хост OTG_HS выдаёт прерывание CHH.
h) В ответ на CHH мы освобождаем канал.

- **Прерывающие разделённые передачи OUT в режиме DMA**

Последовательность такая:

a) Инициализируем и включаем канал как раньше. Ставим бит ODDFRM bit in HCCHAR1.
b) Хост OTG_HS начинает чтение пакета.
c) Хост HS_OTG пытается послать первую часть передачи.
d) После успешной передачи стартовой части хост OTG_HS выдаёт прерывание CHH.
e) В ответ на CHH мы ставим бит COMPLSPLT в HCSPLT1 для посылки завершения.
f) После успешной передачи концевой части хост OTG_HS выдаёт прерывание CHH.
g) В ответ на CHH мы освобождаем канал.

- **Прерывающие разделённые передачи IN в режиме DMA**

Последовательность такая:

a) Инициализируем и включаем канал для начальной части.
b) После получения каналом x гранта от арбитра хост OTG_HS ставит запрос IN в очередь.
c) Хост OTG_HS пытается послать токен IN начальной части в начале следующего нечётного микро-фрейма.
d) После успешной передачи токена IN начальной части хост OTG_HS выдаёт прерывание CHH.
e) В ответ на CHH мы ставим бит COMPLSPLT в HCSPLT2 и посылаем конечную часть.
f) После успешного приёма пакета хост OTG_HS начинает писать его в системную память.
g) После записи пакета в память хост OTG_HS выдаёт прерывание CHH.
h) В ответ на CHH мы освобождаем канал или инициализируем его для следующей начальной части.

- **Изохронные разделённые передачи OUT в режиме DMA**

Последовательность такая:

a) Инициализируем и включаем канал как раньше. Ставим бит ODDFRM bit in HCCHAR1. Пишем поле MPS.
b) Хост OTG_HS начинает чтение пакета.
c) После успешной передачи стартовой части хост OTG_HS выдаёт прерывание CHH.
d) В ответ на CHH мы инициализируем регистры для посылки завершения.
e) После успешной передачи концевой части хост OTG_HS выдаёт прерывание CHH.
f) В ответ на CHH мы освобождаем канал.

- **Изохронные разделённые передачи IN в режиме DMA**

Последовательность такая:

a) Инициализируем и включаем канал для начальной части.
b) После получения каналом x гранта от арбитра хост OTG_HS ставит запрос IN в очередь.
c) Хост OTG_HS пытается послать токен IN начальной части в начале следующего нечётного микро-фрейма.
d) После успешной передачи токена IN начальной части хост OTG_HS выдаёт прерывание CHH.
e) В ответ на CHH мы ставим бит COMPLSPLT в HCSPLT2 и посылаем конечную часть.
f) После успешного приёма пакета хост OTG_HS начинает писать его в системную память.

g) После записи пакета в память хост OTG_HS выдаёт прерывание CHH.

h) В ответ на CHH мы освобождаем канал или инициализируем его для следующей начальной части.

35.13.6. Программная модель устройства

Инициализация точек по сбросу USB

1. Ставим бит NAK для всех конечных точек OUT
   - SNAK = 1 в OTG_HS_DOEPCTLx
2. Демаскируем прерывания:
   - INEP0 = 1 в OTG_HS_DAINTMSK (точки 0 IN)
   - OUTEP0 = 1 в OTG_HS_DAINTMSK (точки 0 OUT)
   - STUP = 1 в DOEPMSK
   - XFRC = 1 в DOEPMSK
   - XFRC = 1 в DIEPMSK
   - TOC = 1 в DIEPMSK
3. Определяем RAM для всех FIFO данных
   - В регистре OTG_HS_GRXFSIZ обеспечиваем возможность принимать управляющие OUT и SETUP. Если контроль порога не включён, то как минимум размер должен быть равен 1 максимальному пакету точки 0 + 2 слова (для состояния управляющего пакета данных OUT) + 10 слов (для пакетов SETUP).
   - В регистре OTG_HS_TX0FSIZ обеспечиваем возможность (в зависимости от номера выбранного FIFO) передавать управляющие данные IN. Как минимум размер должен быть равен 1 максимальному пакету конечной точки 0.
4. Пишем регистры конечной точки 0 OUT для приёма пакетов SETUP
   - STUPCNT = 3 в OTG_HS_DOEPTSIZ0 (для приёма череды 3 пакетов SETUP)

Всё, пакеты SETUP можно принимать.

Инициализация конечной точки по завершению переучёта

1. По прерыванию ENUMDNE в регистре OTG_HS_GINTSTS, из регистра OTG_HS_DSTS выясняем скорость переучёта.
2. В поле MPSIZ регистра OTG_HS_DIEPCTL0 пишем максимальный размер пакета точки 0. Он зависит от скорости переучёта.
3. В режиме DMA битом EPENA в DOEPCTL0 = 1 разрешаем точке 0 OUT принимать пакеты SETUP.

Отныне устройство может принимать пакеты SOF и передавать по точке 0.

Инициализация конечной точки по команде SetAddress

После приёма команды **SetAddress** в пакете SETUP:

1. В регистр OTG_HS_DCFG пишем полученный в команде **SetAddress** адрес
2. Вынуждаем ядро послать пакет состояния IN.

Инициализация конечной точки по команде SetConfiguration/SetInterface

После приёма команд **SetConfiguration** или **SetInterface** в пакете SETUP:

1. По команде **SetConfiguration** в регистрах конечных точек пишем новую конфигурацию.
2. По команде **SetInterface** пишем регистры затронутых командой конечных точек.
3. Некоторые точки в новой конфигурации и установках становятся нерабочими. Деактивируем такие точки.
4. В регистре the interrupt for each active endpoint and mask the interrupts for all inactive endpoints in the OTG_HS_DAINTMSK демаскируем прерывания активных точек и маскируем прерывания неактивных конечных точек.
5. Определяем RAM для всех FIFO.
6. Вынуждаем ядро послать пакет состояния IN.

Отныне устройство может принимать и передавать любые пакеты данных.

Активация конечной точки

Новый тип конечной точки задаём так:

1. В регистре OTG_HS_DIEPCTLx (для IN или двунаправленных точек) или регистре OTG_HS_DOEPCTLx (для OUT или двунаправленных точек) пишем поля:
   - Максимальный размер пакета
   - Активная точка USB = 1
   - Начальное значение переключения данных точки (прерывающие и групповые)
   - Тип точки
   - Номер TxFIFO

Теперь точка активирована и ядро начинает рассматривать получаемые точкой токены и отсылать нужные ответы.

Деактивация конечной точки

Делаем так:

1. Снимаем бит активной точки USB в В регистре OTG_HS_DIEPCTLx (для IN или двунаправленных точек) или регистре OTG_HS_DOEPCTLx (для OUT или двунаправленных точек).

Теперь ядро игнорирует направленные этой точке токены, что приводит к таймауту USB.

NB: Программа должна снять биты NPTXFEM и RXFLVLM в регистре OTG_HS_GINTMSK.

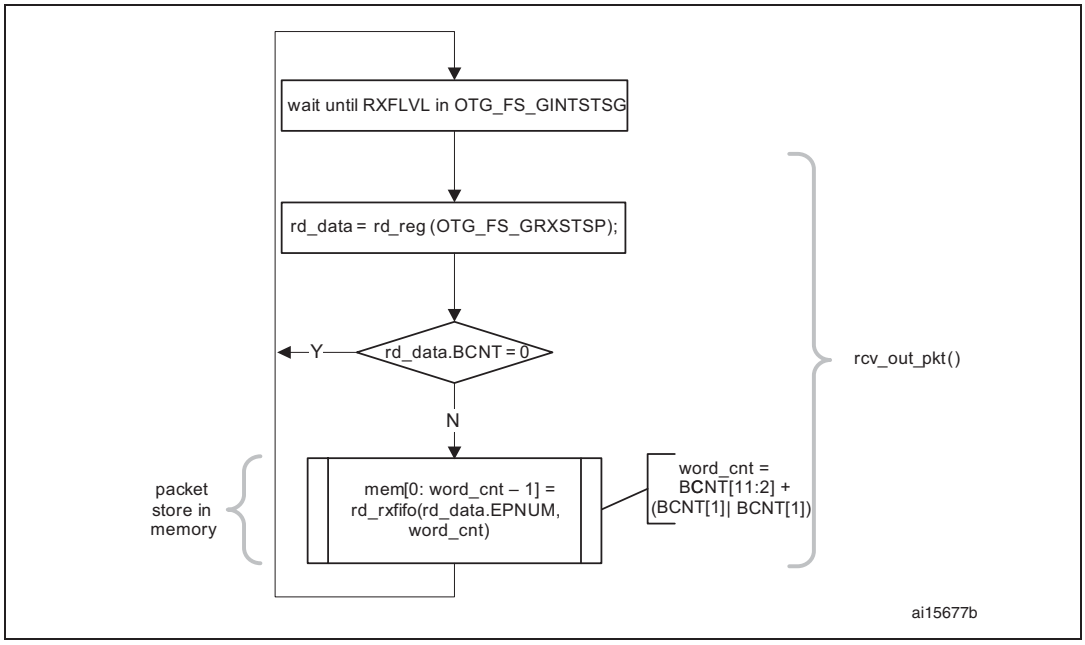
35.13.7. Модель работы

Передачи SETUP и OUT данных

- **Чтение данных пакета из приёмного FIFO в режиме ведомого**

1. По прерыванию RXFLVL регистра OTG_HS_GINTSTS читаем регистр состояния приёма (OTG_FS_GRXSTSP).
2. Записью RXFLVL = 0 (в OTG_HS_GINTMSK) маскируем прерывание RXFLVL регистра OTG_HS_GINTSTS вплоть до извлечения пакета из FIFO.
3. Если счётчик байтов пакета не равен 0, то извлекаем байты из FIFO в память.
4. Статус принятого в FIFO пакета может показывать:
   a) Действует Глобальный OUT NAK:
   PKTSTS = Глобальный OUT NAK, BCNT = 0x000, EPNUM = Всё Равно (0x0), DPID = Всё Равно (0b00).
   b) Пакет SETUP:
   PKTSTS = SETUP, BCNT = 0x008, EPNUM = Номер управляющей EP, DPID = D0.
   c) Конец фазы SETUP:
   PKTSTS = Конец фазы SETUP, BCNT = 0x0, EPNUM = Номер управляющей EP, DPID = Всё Равно (0b00).
   После чтения этой строки из FIFO выдаётся прерывание SETUP этой конечной точки.
   d) Пакет данных OUT:
   PKTSTS = DataOUT, BCNT = Размер пакета (0 ≤ BCNT ≤ 1 024), EPNUM = Номер точки, DPID = PID данных.
   e) Передача данных завершена:
   PKTSTS = Передача данных завершена, BCNT = 0x0, EPNUM = Номер точки, DPID = Всё Равно (0b00).
   После чтения этой строки из FIFO выдаётся прерывание Конца передачи этой точки.
5. После извлечения данных пакета из FIFO надо демаскировать прерывание RXFLVL в регистре (OTG_HS_GINTSTS).
6. Шаги 1–5 повторяются по каждому прерыванию RXFLVL в OTG_HS_GINTSTS. Чтение пустого приёмного FIFO сводит ядро с ума.

Чтение пакета приёмного FIFO



- **Передачи SETUP**

Программные требования

1. При ненулевом поле STUPCNT в регистре OTG_HS_DOEPTSIZx принимаемые пакеты SETUP точки OUT пишутся в FIFO независимо от состояния NAK и бита EPENA в регистре OTG_HS_DOEPCTLx. Поле STUPCNT декрементируется после каждого приёма пакета SETUP. При неверном значении поля STUPCNT пакеты принимаются и поле декрементируется, но определить число присланных пакетов невозможно.
   - STUPCNT = 3 в OTG_HS_DOEPTSIZx
2. В приёмном FIFO данных управляющей точки всегда надо отводить место для приёма трёх пакетов SETUP.
   - Резервируется 10 слов. Три слова для первого пакета SETUP, 1 слово для статуса Конца фазы SETUP и 6 слов для ещё двух пакетов SETUP всех управляющих точек.
   - В 3 словах пакета SETUP хранятся 8 байтов данных и 4 байта статуса.
   - Это место используется ядром только для данных SETUP.
3. Из FIFO надо читать 2 слова пакета SETUP.
4. Слово статуса Конца фазы SETUP надо выбрасывать.

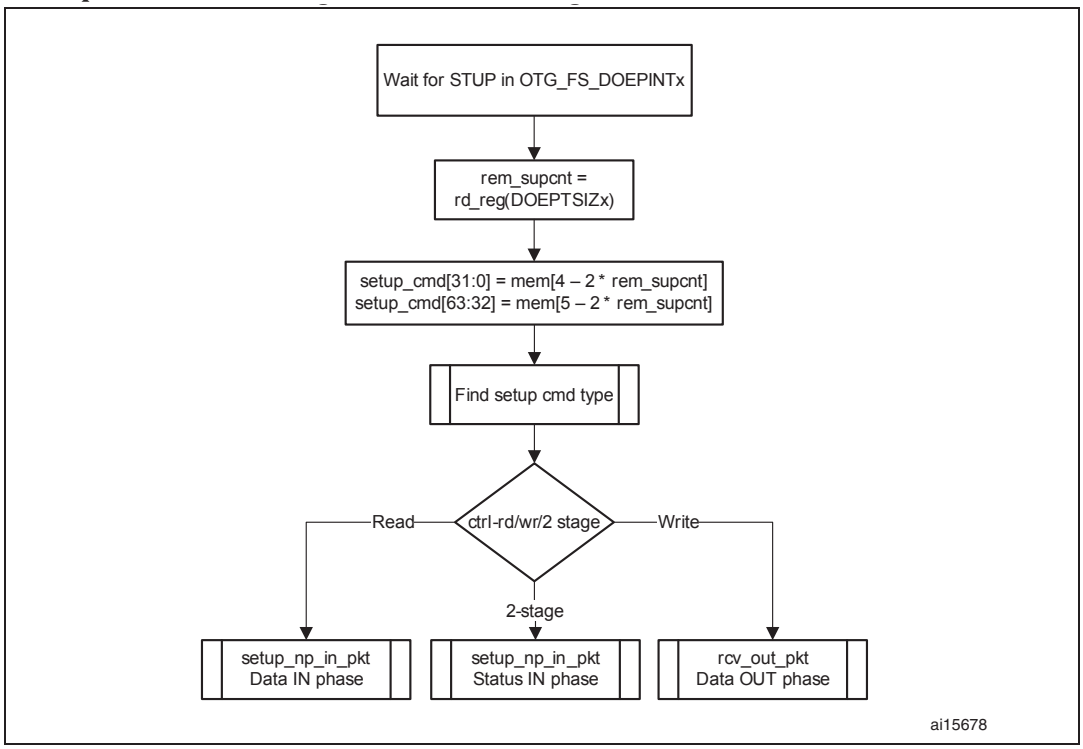
Внутренний ход данных

1. Принимаемые пакеты SETUP пишутся в FIFO без проверки доступного места и независимо от состояния NAK и бита STALL.
   - В точках IN/OUT, принявших пакет SETUP, ядро ставит биты IN NAK и OUT NAK.
2. По каждому пакету SETUP декрементируется поле STUPCNT и в FIFO пишутся 3 слова:
   - В первом слове информация для ядра
   - Во втором слове первые 4 байта команды SETUP
   - В третьем слове последние 4 байта команды SETUP
3. При изменении фазы SETUP на IN/OUT данных ядро пишет в FIFO слово Конца фазы SETUP.
4. На стороне AHB пакеты SETUP опустошаются программой.
5. По чтению из FIFO слова Конца фазы SETUP выдаётся прерывание STUP OTG_HS_DOEPINTx.
   - Ядро снимает бит разрешения управляющей точки OUT.

Программная последовательность

1. В регистр OTG_HS_DOEPTSIZx пишем STUPCNT = 3
2. Ждём прерывания RXFLVL (OTG_HS_GINTSTS) и читаем данные пакета из FIFO.
3. Прерывание STUP (OTG_HS_DOEPINTx) отмечает завершение передачи данных SETUP.
   - По нему из регистра OTG_HS_DOEPTSIZx узнаём число принятых пакетов SETUP и обрабатываем последний.

Обработка пакета SETUP



Обработка череды более 3 пакетов SETUP

Обычно при ошибке пакета SETUP хост не посылает подряд более трёх пакетов SETUP одной точке, но при нормальной работе это не исключено. В этом случае контроллер OTG_HS выдаёт прерывание B2BSTUP в регистре OTG_HS_DOEPINTx.

- **Установка Глобального OUT NAK**

Внутренний ход данных

1. При установке бита SGONAK в регистре OTG_HS_DCTL в FIFO пишутся только пакеты SETUP, не-изохронные пакеты OUT получают ответ NAK, изохронные пакеты OUT игнорируются.
2. В приёмный пишется слово Глобального OUT NAK. Места должно хватать.
3. При его чтении из FIFO выдаётся прерывание GONAKEFF в регистре OTG_HS_GINTSTS.
4. Теперь можно снять бит SGONAK в регистре OTG_HS_DCTL.

Программная последовательность

1. Пишем SGONAK = 1 в регистре OTG_HS_DCTL
2. Ждём прерывания GONAKEFF в регистре OTG_HS_GINTSTS.
3. Между установкой бита SGONAK в OTG_FS_DCTL и прерыванием GONAKEFF (OTG_HS_GINTSTS) нормальные пакеты OUT принимаются.
4. Это прерывание можно временно маскировать записью GINAKEFFM = 0 в регистре OTG_HS_GINTMSK
5. Из режима Глобального OUT NAK выходят снятием бита SGONAK в регистре OTG_HS_DCTL. Заодно очищается прерывание GONAKEFF (OTG_HS_GINTSTS).
6. Если оно было ранее маскировано, то демаскируем его.

- **Выключение точки OUT**

Программная последовательность

1. Сначала включаем Глобальный OUT NAK. SGONAK = 1 в OTG_FS_DCTL
2. Ждём прерывания GONAKEFF (OTG_HS_GINTSTS)
3. Выключаем точку OUT записью:
   - EPDIS = 1 в OTG_HS_DOEPCTLx
   - SNAK = 1 в OTG_HS_DOEPCTLx
4. Ждём прерывания EPDIS (OTG_HS_DOEPINTx), автоматически снимаются биты EPDIS = 0 и EPENA = 0 в регистре OTG_HS_DOEPCTLx

5. Чтобы принимать данные по другим точка OUT выходим из режима Глобального OUT NAK снятием бита SGONAK в регистре OTG_HS_DCTL.

• **Общие не-изохронные передачи данных OUT**

Программная последовательность

1. Для начала размещаем приёмный буфер в памяти.
2. Размер передачи точки должен быть кратен максимальному размеру пакета точки, выравненному на границу слова.
   - размер передачи[EPNUM] = $n \times$ (MPSIZ[EPNUM] + 4 – (MPSIZ[EPNUM] mod 4))
   - счётчик пакетов[EPNUM] = n
   - n>0
3. Размер переданных данных вычисляется по регистру размера передачи.
   - Принято данных = начальный размер – остаток в регистре
   - Число пакетов с принятыми данными = начальное число – остаток в регистре

Внутренний ход данных

1. Сначала в регистрах точки ставим размер передачи и счётчик пакетов, снимаем бит NAK и включаем точку на приём.
2. После снятия бита NAK ядро начинает писать данные в свободное место FIFO. Вместе с данными в FIFO пишется статус пакета и декрементируется счётчик пакетов.
   - Пакеты OUT с ошибкой CRC сливаются из приёмного FIFO автоматически.
   - Повторно посылаемые чередой не-изохронные пакеты OUT той же точке с тем же PID от хоста, который не воспринимает ответ ACK, отбрасываются ядром и счётчик пакетов не декрементируется.
   - При нехватке места в FIFO изохронные и не-изохронные пакеты данных игнорируются, на не-изохронные токены OUT выдаётся ответ NAK.
   - Во всех трёх случаях счётчик пакетов не декрементируется.
3. При обнулении счётчика пакетов или по приёму короткого пакета ставится бит NAK конечной точки и изохронные и не-изохронные пакеты данных игнорируются, на не-изохронные токены OUT выдаётся ответ NAK.
4. Из приёмного FIFO данные извлекается пакетами, по одному пакету за раз.
5. Размер передачи конечной точки уменьшается на размер пакета после его извлечения из FIFO.
6. Строка конца передачи данных OUT пишется в приёмный FIFO после:
   - Обнулении размера передачи и счётчика пакетов
   - Последний принятый в FIFO пакет OUT был коротким (0 ≤ размер < максимума)
7. При извлечении этой строки из FIFO выдаётся прерывание и точка выключается.

Программная последовательность

1. В регистр OTG_HS_DOEPTSIZx пишем размер передачи и счётчик пакетов.
2. В регистре OTG_HS_DOEPCTLx пишем характеристки точки и ставим биты EPENA и CNAK.
3. Ждём прерывания RXFLVL (в OTG_HS_GINTSTS) и извлекаем данные из FIFO.
   - Этот шаг может повторяться несколько раз в зависимости от размера передачи.
4. Прерывание XFRC (OTG_HS_DOEPINTx) отмечает конец не-изохронной передачи данных OUT.
5. По регистру OTG_HS_DOEPTSIZx определяем длину принятых данных.

• **Обычные изохронные передачи данных OUT**

Программные требования

1. Требования такие же, как и для не-изохронных передач OUT.
2. Размер передачи и счётчик пакетов всегда должны быть равны ожидаемому числу пакетов максимальной длины в одном фрейме и не больше. Изохронные передачи OUT не могут занимать более 1 фрейма.
3. Данные и статус изохронных передач OUT надо прочесть из FIFO до конца периодического фрейма (прерывание EOPF в регистре OTG_HS_GINTSTS).
4. Для приёма следующего фрейма изохронную точку OUT надо включить после EOPF (OTG_HS_GINTSTS) и до SOF (OTG_HS_GINTSTS).

Внутренний ход данных

1. Он немного отличается от хода данных не-изохронных точек OUT.
2. При включении изохронной точки OUT установкой бита EPENA и очисткой бита NAK, бит Чётного/Нечётного фрейма должен быть установлен правильно. Ядро принимает изохронные данные отдельным фреймом только при:
   - EONUM (в OTG_HS_DOEPCTLx) = SOFFN[0] (в OTG_FS_DSTS)
3. После чтения данных и статуса пакета из FIFO в поле RXDPID регистра OTG_HS_DOEPTSIZx пишется PID данных последнего принятого пакета.

Программная последовательность

1. В регистр OTG_HS_DOEPTSIZx пишем размер передачи и счётчик пакетов.
2. В регистре OTG_HS_DOEPTSIZx ставим характеристики точки и пишем EPENA = 1, CNAK=1 и EONUM = (0: Чётный/1: Нечётный)
3. Ждём прерывания RXFLVL (в OTG_HS_GINTSTS) и извлекаем данные пакета из FIFO
   - Этот шаг может повторяться несколько раз в зависимости от размера передачи.
4. Прерывание XFRC (OTG_HS_DOEPINTx) отмечает конец изохронной передачи данных OUT. При этом данные не обязательно будут достоверными.
5. Для изохронных передач OUT это прерывание выдаётся не всегда, лучше использовать прерывание IISOOXFRM в регистре OTG_HS_GINTSTS.
6. По регистру OTG_HS_DOEPTSIZx определяем длину принятых данных. Данные достоверны когда:
   - RXDPID = D0 (в OTG_HS_DOEPTSIZx) и число пакетов с данными = 1
   - RXDPID = D1 (в OTG_HS_DOEPTSIZx) и число пакетов с данными = 2
   - RXDPID = D2 (в OTG_HS_DOEPTSIZx) и число пакетов с данными = 3

   Число пакетов с данными = Начальное число пакетов – Конечное число пакетов.

   Недостоверные пакеты можно выбрасывать.

• **Незавершённые изохронные передачи данных OUT**

Внутренний ход данных

1. У изохронных точек OUT прерывание XFRC (в OTG_HS_DOEPINTx) не выдаётся когда:
   - Приёмный FIFO не вмещает весь пакет данных ISO OUT и он отбрасывается,
   - Возникла ошибка CRC принятого пакета,
   - Принятый токен OUT нарушен,
   - Программа не успевает читать данные из FIFO.
2. При появлении конца периодического фрейма до завершения передач всех изохронных точек OUT ядро выдаёт прерывание IISOOXFRM в регистре OTG_HS_GINTSTS, говоря, что не для всех точек выдавалось прерывание XFRC (в OTG_HS_DOEPINTx). Точка с незавершённой передачей остаётся включённой, но передачи по ней нет.

Программная последовательность

1. Прерывание IISOOXFRM (OTG_HS_GINTSTS) означает, что есть хоть одна изохронная точка OUT с незавершенной передачей.
2. Если оно возникло из-за не прочитанного FIFO, то из FIFO надо извлечь все данные и статус. При этом может выдаться прерывание XFRC (OTG_HS_DOEPINTx). В этом случае надо снова включить точку для приёма изохронных данных OUT в следующем фрейме.
3. По прерыванию IISOOXFRM (OTG_HS_GINTSTS) в регистрах OTG_HS_DOEPCTLx надо определить конечные точки с незавершёнными передачами. При этом:
   - Бит EONUM (в OTG_HS_DOEPCTLx) = SOFFN[0] (в OTG_HS_DSTS)
   - EPENA = 1 (в OTG_HS_DOEPCTLx)
4. Предыдущий шаг надо выполнить до прерывания SOF (в OTG_HS_GINTSTS), чтобы номер текущего фрейма не изменился.
5. У изохронных точек OUT с незавершёнными передачами надо отбросить данные и выключить точку установкой бита EPDIS в регистре OTG_HS_DOEPCTLx.
6. Ждём прерывания EPDIS (в OTG_HS_DOEPINTx) и включаем точку для следующего фрейма.

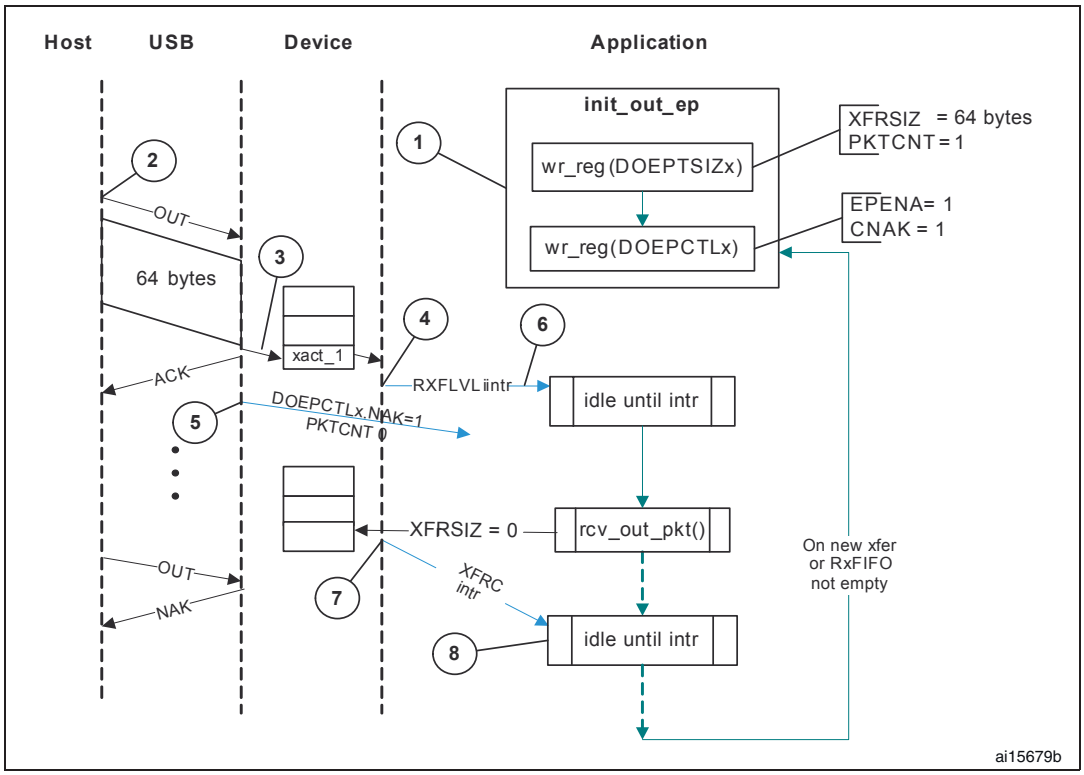
- Из-за задержки выключения точки следующий фрейм может быть не получен.

• Остановка не-изохронной точки OUT

1. Ставим ядро в режим Глобального OUT NAK.
2. Выключаем точку
   - Вместо установки бита SNAK в OTG_HS_DOEPCTL, ставим STALL = 1 (OTG_HS_DOEPCTL). Бит STALL старше бита a NAK.
3. При готовности отмены ответов STALL снимаем бит STALL в OTG_HS_DOEPCTLx.
4. При изменении бита STALL по командам **SetFeature.Endpoint Halt** или **ClearFeature.Endpoint Halt** сделать это надо до фазы передачи Статуса управляющей конечной точки.

Примеры

• Групповая точка OUT в режиме ведомого



После команд **SetConfiguration/SetInterface** надо инициировать все конечные точки OUT установкой CNAK = 1 и EPENA = 1 (в OTG_HS_DOEPCTLx) и записью подходящих XFRSIZ и PKTCNT в регистрах OTG_HS_DOEPTSIZx.

1. Хост пробует передавать токен OUT по конечной точке.
2. По нему ядро пишет пакет в доступное место RxFIFO.
3. По завершению записи ядро выдаёт прерывание RXFLVL (в OTG_HS_GINTSTS).
4. После приёма PKTCNT пакетов ядро ставит точке бит NAK для задержки приёма.
5. Программа обрабатывает прерывание и читает данные из RxFIFO.
6. По прочтению всех данных (равного XFRSIZ), ядро выдаёт прерывание XFRC в регистре OTG_HS_DOEPINTx.
7. Программа обрабатывает конец передачи.

Передачи данных IN

• Запись пакета

1. Работать можно по прерываниям или опросом.
   - При опросе программа по регистру OTG_HS_DTXFSTSx следит за наличием свободного места в передающем FIFO данных.
   - В режиме прерываний программа ждёт прерывания TXFE (в OTG_HS_DIEPINTx) и по регистру OTG_HS_DTXFSTSx определяет наличие места в FIFO данных для всего пакета ненулевой длины.
   - Для пакета нулевой длины места в FIFO не требуется.

2. Перед записью в FIFO данных надо записать регистр управления точки (OTG_HS_DIEPCTLx) командой чтение-модификация-запись во избежание изменения регистра (исключая установку бита включения точки).

При наличии места в передающем FIFO можно писать несколько пакетов для одной точки. Для периодических точек IN надо писать по одному пакету на микрофрейм. Новый пакет можно писать только после подтверждения завершения предыдущей передачи.

- **Установка NAK точки IN**

Внутренний ход данных

1. При установке IN NAK точки ядро прекращает передачи независимо от наличия места в наличии данных в передающем FIFO точки.
2. Не-изохронные токены IN получают ответ NAK
   - Изохронные токены IN получают ответ с нулевой длиной данных
3. В ответ на бит SNAK в OTG_HS_DIEPCTLx ядро выдаёт прерывание INEPNE (действует NAK точки IN) в OTG_HS_DIEPINTx.
4. Снимается режим IN NAK установкой бита CNAK в регистре OTG_HS_DIEPCTLx.

Программная последовательность

1. Для остановки передач по точке IN программа пишет SNAK = 1 в OTG_HS_DIEPCTLx
2. Ждём подтверждения остановки по прерыванию INEPNE в OTG_HS_DIEPINTx.
3. В промежутке между установкой бита NAK и прерыванием INEPNE ядро может передавать данные IN.
4. Программа маскировать это прерывание записью INEPNEM = 0 в регистр DIEPMSK
5. Из режима NAK точки выходят снятием бита NAKSTS в OTG_HS_DIEPCTLx (запись CNAK = 1 in OTG_HS_DIEPCTLx). Этим также чистится прерывание INEPNE (в OTG_HS_DIEPINTx).
6. Ранее маскированное прерывание демаскируют записью INEPNEM = 1 in DIEPMSK.

- **Выключение точки IN**

Программная последовательность

1. Программа перестаёт писать данные для выключаемой точки IN.
2. Ставим режим NAK записью SNAK = 1 в OTG_HS_DIEPCTLx.
3. Ждём прерывания INEPNE в OTG_HS_DIEPINTx.
4. В регистр OTG_HS_DIEPCTLx пишем EPDIS = 1 и SNAK = 1.
5. Прерывание EPDISD в OTG_HS_DIEPINTx подтверждает выключение точки. Вместе с этим ядро снимает биты EPENA и EPDIS в регистре OTG_HS_DIEPCTLx.
6. У периодических IN EP число переданных данных узнают по регистру OTG_HS_DIEPTSIZx.
7. Программа должна слить данные из передающего FIFO точки записью номера точки в TXFNUM, и TXFFLSH = 1 регистра OTG_HS_GRSTCTL

Конец слива определяют по снятию ядром бита TXFFLSH опросом регистра OTG_HS_GRSTCTL. Новые данные можно передавать после включения точки.

- **Обычные не-периодические передачи данных IN**

Программные требования

1. Сначала надо убедиться, что все передаваемые данные лежат в одном буфере.
2. Для передач IN размер передачи включает несколько пакетов максимальной длины и один короткий пакет, передаваемый последним.
   - Для передачи нескольких пакетов:
     Размер передачи[EPNUM] = $x \times$ MPSIZ[EPNUM] + sp
     Если (sp > 0), то Счётчик пакетов[EPNUM] = x + 1. Иначе, Счётчик пакетов[EPNUM] = x
   - Для передачи пакета без данных:
     Размер передачи[EPNUM] = 0
     Счётчик пакетов[EPNUM] = 1
   - Для передачи нескольких максимальных пакетов и пакета без данных передача разбивается на две части: пакеты с данными и отдельно пакет без данных.

Первая передача: Размер передачи[EPNUM] = x × MPSIZ[EPNUM]; Счётчик пакетов = n;
Вторая передача: Размер передачи[EPNUM] = 0; Счётчик пакетов = 1;

3. После включения точки на передачу ядро изменяет регистр размера передачи. В конце передачи IN по нему определяют сколько данных было передано из FIFO.
4. В FIFO извлечено = Начальный размер передачи – Остаток размера передачи
   - Передано по USB = (Начальный счётчик пакетов – Остаток счётчика пакетов) × MPSIZ[EPNUM]
   - Ещё передавать по USB = (Начальный размер передачи – Передано по USB)

Внутренний ход данных

1. В регистрах точки надо записать размер передачи и счётчик пакетов и включить точку на передачу.
2. Конечно же, данные надо записать в FIFO точки.
3. После записи пакета в FIFO размер передачи уменьшается на размер пакета. Данные пишутся в FIFO до обнуления размера передачи. После записи данных в FIFO инкрементируется "счётчик пакетов в FIFO" (Это 3-бит счётчик, управляемый ядром для каждого FIFO). Для пакетов без данных в FIFO ставится особый флаг, данные не пишутся.
4. Данные из FIFO передаются ядром по токену IN. По каждому ответу ACK не-изохронного пакета IN декрементируется счётчик пакетов вплоть до его обнуления. При таймауте счётчик пакетов не декрементируется.
5. Пакет нулевой длины высылается по токену IN и счётчик пакетов декрементируется.
6. По приёму токена IN при пустом FIFO и нулевом счётчике пакетов ядро выдаёт прерывание ITTXFE, извещая, что бит не стоит NAK. Не-изохронные точки отсылают ответ NAK.
7. Ядро само отматывает указатели FIFO и прерывание таймаута не выдаётся.
8. При нулевых размере передачи и счётчике пакетов выдаётся прерывание конца передачи (XFRC) и точка выключается.

Программная последовательность

1. В регистр OTG_HS_DIEPTSIZx пишем размер передачи и счётчик пакетов.
2. В регистр OTG_HS_DIEPCTLx пишем характеристики точки и ставим биты CNAK и EPENA.
3. При передаче пакета без данных надо опрашивать регистр OTG_HS_DTXFSTSx для определения наличия места в FIFO, или использовать TXFE (в OTG_HS_DIEPINTx).

• **Обычные периодические передачи данных IN**

Программные требования

1. Они совпадают с пунктами 1, 2, 3 и 4 *Общих не-периодических передач данных IN* с некоторым изменением пункта 2.
   - Передавать можно только несколько максимальных пакетов данных с возможным коротким пакетом с данными в конце передачи. При этом:
     Размер передачи[EPNUM] = x × MPSIZ[EPNUM] + sp (где $x \geq 0$, и $0 \leq sp < MPSIZ[EPNUM]$)
     Если ($sp > 0$), то Счётчик пакетов[EPNUM] = $x + 1$ Иначе, Счётчик пакетов[EPNUM] = x;
     MCNT[EPNUM] = Счётчик пакетов[EPNUM]
   - Пакеты без данных передаются отдельно. При этом:
   - Размер передачи[EPNUM] = 0
     Счётчик пакетов[EPNUM] = 1; MCNT[EPNUM] = Счётчик пакетов[EPNUM]
2. Программа может планировать передачу только по одному фрейму за раз.
   - (MCNT – 1) × MPSIZ ≤ XFERSIZ ≤ MCNT × MPSIZ
   - PKTCNT = MCNT (in OTG_HS_DIEPTSIZx)
   - Если XFERSIZ < MCNT × MPSIZ, то последний пакет короткий.
   - **NB**: MCNT, PKTCNT и XFERSIZ лежат в OTG_HS_DIEPTSIZx, MPSIZ лежит в OTG_FS_DIEPCTLx.
3. В FIFO должны быть записаны все передаваемые данные до приёма токена IN. При нехватке в FIFO хоть одного слова он будет считаться пустым. Тогда:
   - Изохронная точка IN передаст пакет без данных

- Прерывающая точка IN передаст ответ NAK.

4. Для широкополосных точек IN с 3 пакетами в фрейме размер FIFO точки должен быть *2 × max_pkt_size*, третий пакет надо писать после передачи первого по USB.

Внутренний ход данных

1. В регистрах точки пишем размер передачи и счётчик пакетов и включаем точку на передачу.
2. Также пишем в FIFO точки передаваемые данные.
3. При записи очередного пакета в FIFO размер передачи уменьшается на размер пакета. Донные пишутся до обнуления размера передачи.
4. По приёму пакета IN периодическая точка передаёт данные из FIFO. Если там нет полного пакета, то ядро выдаёт прерывание пустого TxFIFO точки.
   - Изохронная точка IN передаст пакет без данных
   - Прерывающая точка IN передаст ответ NAK.
5. Счётчик пакетов точки декрементируется когда:
   - Изохронная точка IN передаст пакет без данных
   - Прерывающая точка IN передаст ответ NAK.
   - Обнулятся размер передачи и счётчик пакетов, будет выдано прерывание завершения передачи и точка выключится.
6. Если по "Интервалу периодического фрейма" (`PFIVL` в `OTG_HS_DCFG`) ядро найдёт непустые FIFO изохронных точек IN для текущего фрейма, то оно выдаст прерывание `IISOIXFR` в `OTG_HS_GINTSTS`.

Программная последовательность

1. В регистр `OTG_HS_DIEPCTLx` пишем характеристики точки и ставим биты `CNAK` `EPENA`.
2. Пишем передаваемые данные следующего фрейма в FIFO.
3. Прерывание `ITTXFE` (в `OTG_HS_DIEPINTx`) покажет, что в FIFO записаны ещё не все данные.
4. Если прерывающая точка включена, то игнорируем прерывание, иначе включаем её, чтобы она передала данные по следующему токену IN.
5. Прерывание `XFRC` (в `OTG_HS_DIEPINTx`) без прерывания `ITTXFE` покажет успешное завершение изохронной передачи IN. При этом размер передачи и счётчик пакетов в регистре `OTG_HS_DIEPTSIZx` должны быть нулевыми.
6. Прерывание `XFRC` (в `OTG_FS_DIEPINTx`) независимо от наличия прерывания `ITTXFE` interrupt (in `OTG_FS_DIEPINTx`) покажет успешное завершение прерывающей передачи IN. При этом размер передачи и счётчик пакетов в регистре `OTG_HS_DIEPTSIZx` должны быть нулевыми.
7. Прерывание `IISOIXFR` в `OTG_HS_GINTSTS` без упомянутых прерываний покажет, что ядро не приняло хоть один периодический токен IN в текущем фрейме.

- **Незавершённые изохронные передачи данных IN**

Внутренний ход данных

1. Изохронная передача IN считается незавершённой когда:
   a) Ядро принимает нарушенный токен IN по какой-либо изохронной точке. В этом случае выдаётся прерывание `IISOIXFR` в `OTG_HS_GINTSTS`.
   b) Программа не успевает записать в FIFO все передаваемые данные до получения токена IN. В этом случае выдаётся прерывание пустого TxFIFO в `OTG_HS_DIEPINTx`. Его можно игнорировать, поскольку оно приведёт к прерыванию `IISOIXFR` в `OTG_HS_GINTSTS` в конце периодического фрейма. В ответ на токен IN ядро пошлёт пакет без данных.
2. Запись данных в передающий FIFO надо немедленно остановить.
3. Ставим бит `NAK` и бит выключения точки.
4. Ядро выключает точку, снимает бит выключения и выдаёт прерывание Выключения точки.

Программная последовательность

1. Прерывание пустого TxFIFO изохронной точки можно игнорировать, поскольку оно приведёт к прерыванию `IISOIXFR` в `OTG_HS_GINTSTS`.
2. Оно извещает о незавершённой передаче на какой-либо изохронной точке IN.

3. Точка с незавершённой передачей определяется чтением Управляющих регистров всех изохронных точек IN.
4. Останавливаем запись в периодические передающие FIFO этих точек.
5. Выключаем точку записью SNAK = 1 и EPDIS = 1 в OTG_HS_DIEPCTLx
6. Выключение точки подтверждается соответствующим прерыванием в регистре OTG_HS_DIEPINTx.
   - В этот момент надо слить данные из FIFO с помощью регистра OTG_HS_GRSTCTL или переписать их включением точки для новой передачи в следующем микрофрейме.

- **Остановка не-изохронной точки IN**

Программная последовательность

1. Выключаем точку IN и затем ставим бит STALL в регистре OTG_HS_DIEPCTLx
   - Бит STALL всегда старше бита NAK
2. Прерывание Выключения точки (в OTG_HS_DIEPCTLx) извещает об этом.
3. Сливаем ассоциированный FIFO. В случае не-периодической точки надо повторно включить остальные не-периодические точки.
4. При готовности отмены ответов STALL для точки снимаем бит STALL в OTG_HS_DIEPCTLx.
5. При изменении бита STALL по командам **SetFeature.Endpoint Halt** или **ClearFeature.Endpoint Halt** изменяем его до фазы Статуса управляющей конечной точки.

Особый случай: остановка управляющей точки OUT

Если хост посылает токенов IN/OUT больше, чем предусмотрено а пакете SETUP, то их надо остановить. В этом случае, в фазе данных, после передачи предусмотренных пакетом SETUP данных надо разрешить прерывания ITTXFE в OTG_HS_DIEPINTx и OTEPDIS в OTG_HS_DOEPINTx. Затем при обработке прерываний ставим STALL точки и чистим прерывание.

35.13.8. Худшее время ответа

Это ответ на токен, следующий за изохронным OUT при работе OTG_FS устройством. Время ответа зависит от частоты тактов AHB (7 тактов PHY при равных частотах тактов PHY и AHB, и меньше при более высокой частоте AHB).

Регистры ядра лежат в домене AHB и оно не принимает новых токенов до обновления регистров, а изохронные передачи OUT не требуют ответа, и следующий групповой/прерывающий токен может получить ответ NAK. Изохронные токены отбрасываются и выдаётся прерывание IISOIXFR.

Хост воспринимает отсутствие ответа на токен SETUP как таймаут и повторяет его передачу.

Выбор значения TRDT в OTG_HS_GUSBCFG

Поле TRDT (OTG_HS_GUSBCFG) это число тактов PHY, отведённое MAC для получения статуса FIFO и чтения первого данного из блока PFC после получения токена IN. Это включает задержку синхронизации тактов PHY и AHB (в худшем случае 5 тактов).

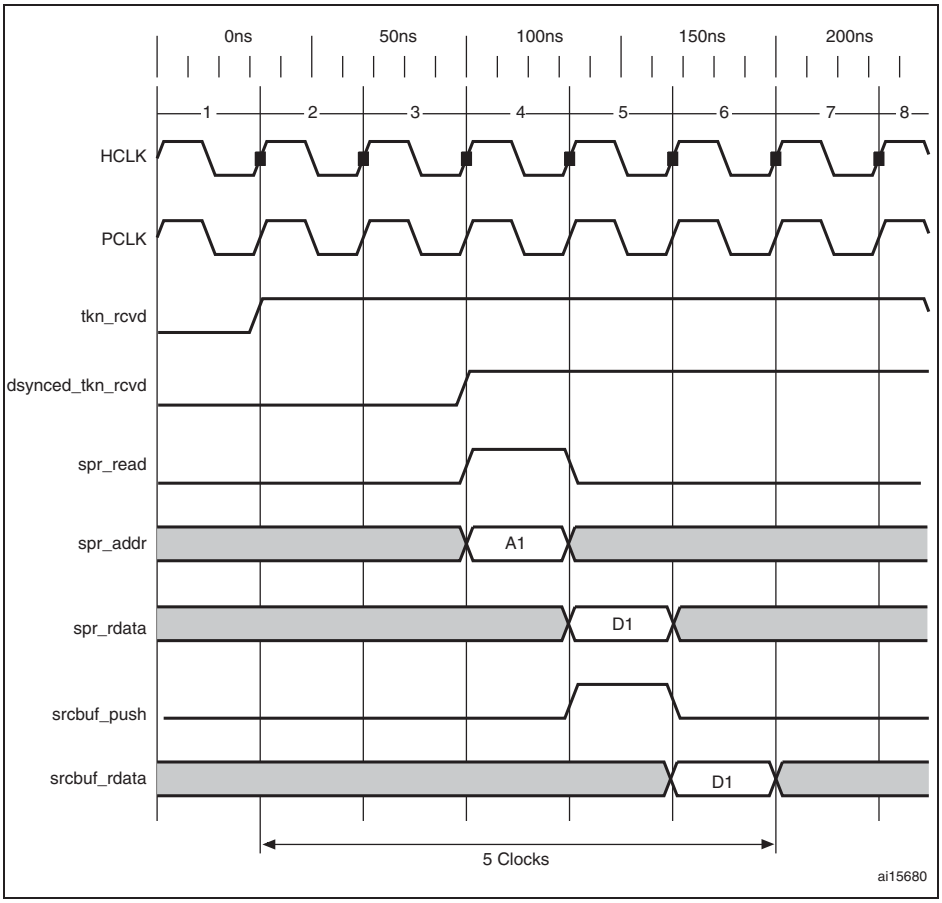
Информация о получении MAC токена IN синхронизируется с частотой PFC (такты AHB). Затем PFC читает данные из SPRAM и пишет их в буфер с двойным тактированием (глубиной 4), откуда их читает MAC.

Если частота AHB выше частоты PHY, то можно использовать меньшие значения TRDT (в OTG_HS_GUSBCFG).

На рисунке ниже есть сигналы:

- tkn_rcvd: Сигнал PFC от MAC о получении токена
- dynced_tkn_rcvd: Двухчастотный tkn_rcvd, от PCLK в домен HCLK
- spr_read: Чтение в SPRAM
- spr_addr: Адрес в SPRAM
- spr_rdata: Чтение данных из SPRAM
- srcbuf_push: Запись в буфер
- srcbuf_rdata: Чтение данных из буфера. Данные видны для MAC

Максимальные значения TRDT

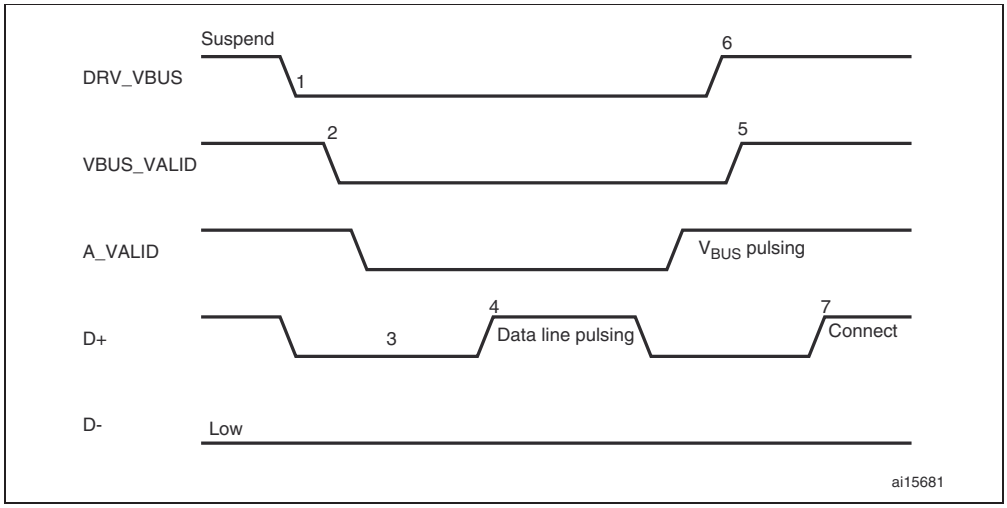


35.13.9. Программная модель OTG

Контроллер OTG_HS поддерживает протоколы HNP и SRP. Если ядро подключено к концу “A”, то именуется A-устройством, если к концу “B” то B-устройством. В режиме хоста контролер OTG_HS выключает V_{BUS}. Протокол SRP позволяет B-устройству попросить A-устройство включить V_{BUS}. Устройство выдаёт импульсы по линии данных и V_{BUS}, но при SRP хост обнаруживает импульс только на линии данных или V_{BUS}. Протокол HNP позволяет B-устройству переговорить и стать хостом. По переговорам после HNP B-устройство приостанавливает шину и оборачивается в периферию.

Протокол запроса сессии A-устройства

Программа ставит бит разрешения SRP в регистре Конфигурации Ядра и контроллер становится A-устройством.



DRV_VBUS = Сигнал PHY подачи V_{BUS}

VBUS_VALID = Сигнал рабочего V_{BUS} от PHY

A_VALID = Сигнал рабочего уровня V_{BUS} от A-периферии к PHY

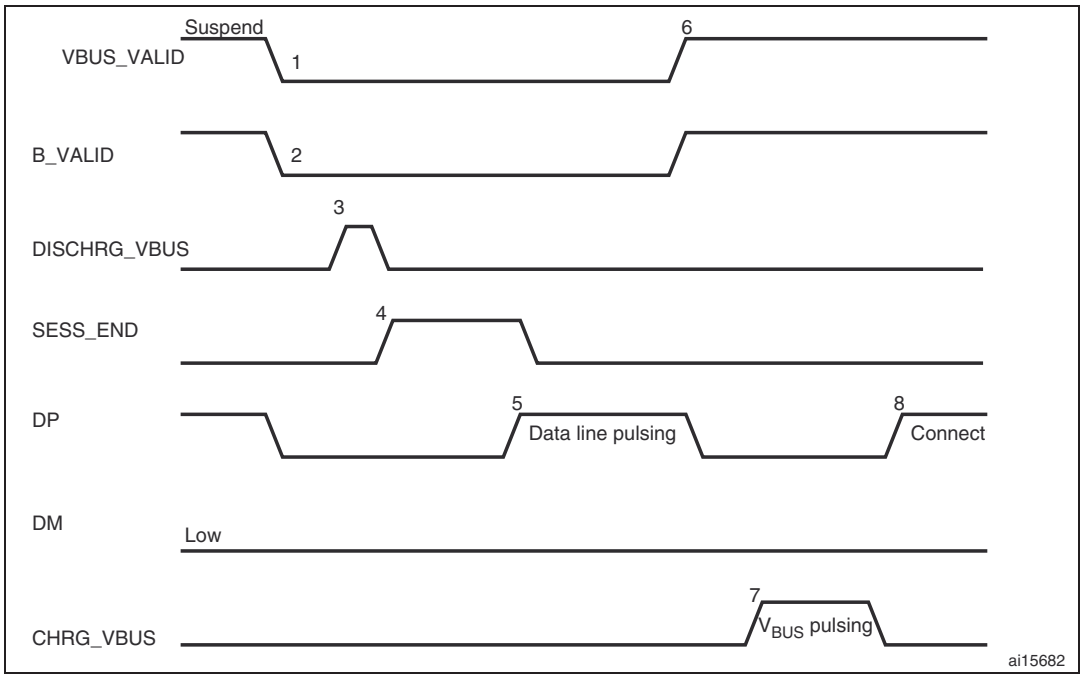
D+ = Линия данных плюс

D- = Линия данных минус

1. Программа останавливает порт и выключает питание простаивающей шины установкой битов остановки порта и выключения питания в регистре управления хоста.
2. PHY показывает выключение питания снятием сигнала VBUS_VALID.
3. Устройство должно обнаружить SE0 не менее чем за 2 ms до пуска SRP при выключенном V_{BUS}.
4. Для запуска SRP устройство подключает свой резистор подпорки линии данных на время от 5 до 10 ms. Контроллер OTG_HS обнаруживает этот импульс.
5. Устройство поднимает V_{BUS} выше порога (2.0 V минимум). Контроллер OTG_HS ставит бит запроса сессии (SRQINT в OTG_HS_GINTSTS) и выдаёт прерывание.
6. По прерыванию программа включает питание установкой бита в регистре управления хоста. PHY показывает включение питания выдачей сигнала VBUS_VALID.
7. USB запитан, устройство подключено, процесс SRP завершён.

Протокол запроса сессии B-устройства

Программа ставит бит разрешения SRP в регистре Конфигурации Ядра и контроллер становится B-устройством. Можно запросить сессию от хоста.



VBUS_VALID = Сигнал рабочего V_{BUS} от PHY
B_VALID = B-peripheral valid session to PHY
DISCHRG_VBUS = Сигнал разрядки для PHY
SESS_END = Сигнал конца сессии для PHY
CHRG_VBUS = Сигнал зарядки V_{BUS} для PHY
DP = Линия данных плюс
DM = Линия данных минус

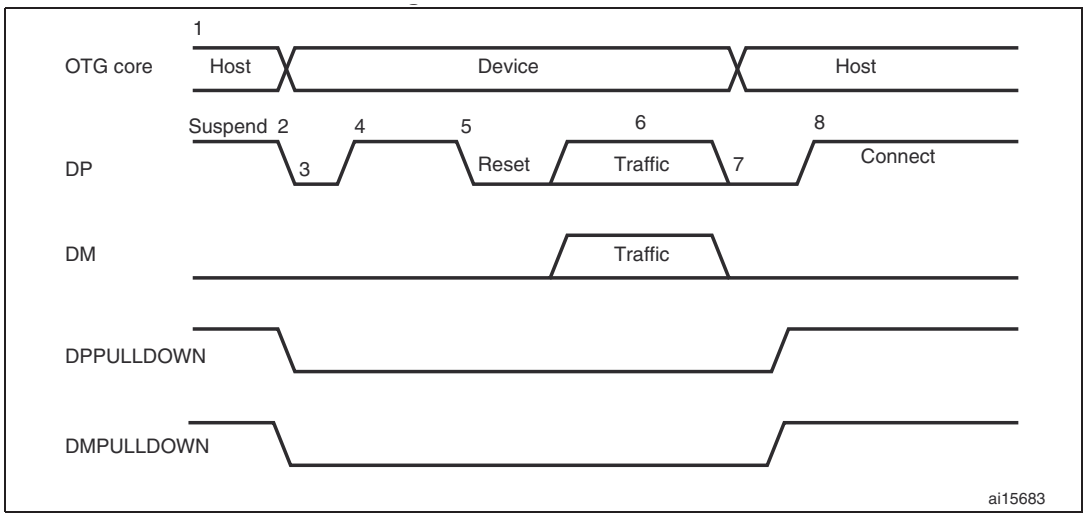
1. При простое шины хост останавливает порт и выключает питание. Через 3 ms простоя контроллер OTG_HS ставит бит ранней остановки в регистре прерываний ядра. Затем он там же ставит бит остановки USB и подаёт PHY сигнал разрядки V_{BUS}.
2. PHY выдаёт устройству конец сессии. Это исходное состояние SRP. Контроллеру OTG_HS нужно 2 ms SE0 перед запуском SRP.
   Для полноскоростных трансиверов USB 1.1, после снятия BSVLD (в OTG_FS_GOTGCTL) программе надо дождаться разряда V_{BUS} до 0.2 V. Время указывает производитель.
3. Ядро USB OTG говорит PHY ускорить разряд V_{BUS}.
4. Программа пускает SRP записью бита запроса а регистре Управления OTG. Контроллер OTG_HS выдаёт импульс на линии данных с последующим импульсом на V_{BUS}.
5. Хост обнаруживает SRP по одному из импульсов и включает V_{BUS}. PHY показывает устройству включение V_{BUS}.
6. Контроллер OTG_HS выдаёт импульс V_{BUS}.
   Хост начинает новую сессию включением V_{BUS}. Контроллер OTG_FS выдаёт прерывание

изменения состояния запроса сессии в регистре состояния прерываний OTG. Программа бит успешной сессии в регистре Управления OTG.

7. USB запитан, контроллер OTG_FS подключен, процесс SRP завершён.

Протокол беседы хоста A-устройства (HNP)

Протокол HNP переключает хост USB из A-устройства в B-устройство. Программа ставит бит разрешения HNP в регистре конфигурации ядра USB и контроллер становится A-устройством.

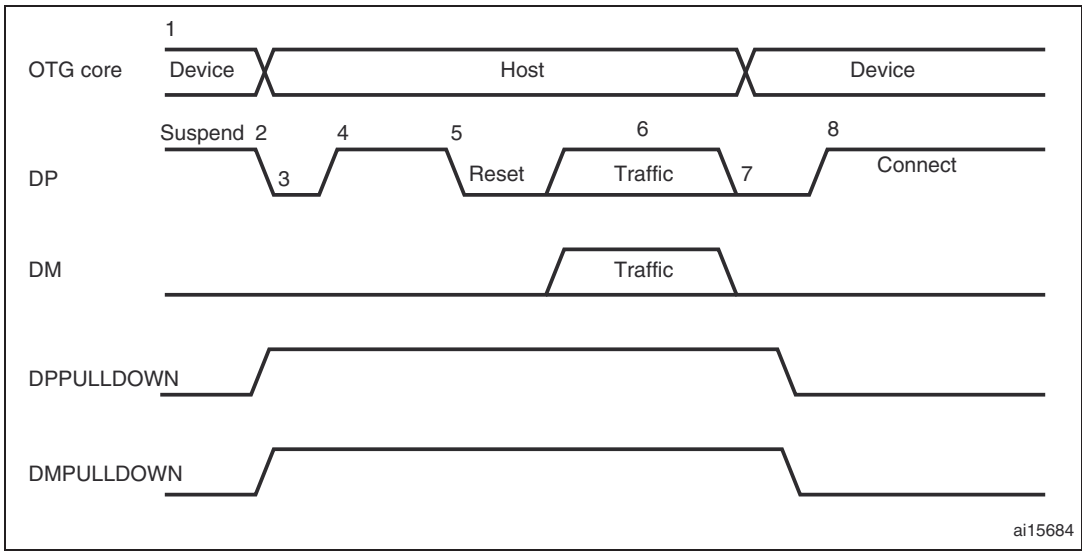


DPPULLDOWN = signal from core to PHY to enable/disable the pull-down on the DP line inside the PHY.
DMPULLDOWN = signal from core to PHY to enable/disable the pull-down on the DM line inside the PHY.

1. Контроллер OTG_HS посылает B-устройству дескриптор **SetFeature b_hnp_enable** для включения поддержки HNP. Оно ответом ACK подтверждает поддержку HNP. Программа установкой бита включения HNP в регистре управления говорит об этом контроллеру OTG_HS.
2. По завершению работы с шиной программа пишет бит остановки порта в его регистр управления и статуса.
   B-устройство видит остановку USB, отключается, говоря о начале HNP. B-устройство инициирует HNP только для переключения в режим хоста, иначе шина продолжает стоять.
   Контроллер OTG_HS выдаёт прерывание начала переговоров хоста
   Контроллер OTG_HS отключает резисторы подпорки и подтяжки линии DM в PHY, показывая роль устройства. PHY подключает резистор подпорки линии DP, показывая подключение.
   Работу режима устройства определяют по биту текущего режима в регистре управления и статуса OTG.
4. B-устройство видит подключение, выдаёт сброс USB и задаёт переучёт контроллера OTG_HS.
5. B-устройство остаётся хостом, пускает трафик и в конце останавливает шину.
   Контроллер OTG_HS через 3 ms простоя шины ставит бит раннего останова в регистре прерываний ядра и затем бит остановки USB там же.
6. В Уговорённом режиме контроллер OTG_HS видит останов, отключается, вертается в роль хоста и, принимая роль хоста, подключает резисторы подпорки и подтяжки линии DM в PHY.
7. Контроллер OTG_HS выдаёт прерывание Изменения статуса подключения. По статусу подключения в регистре управления и статуса OTG определяют, что контроллер OTG_HS работает как A-устройство. Это конец HNP. Текущий режим хоста определяют по биту текущего режима там же.
8. B-устройство подключается, завершая процесс HNP.

HNP B-устройства

HNP переключает хост USB из B-устройства в A-устройство установкой бита разрешения HNP в регистре конфигурации ядра USB.



DPPULLDOWN = сигнал от ядра к PHY включения/выключения резистора подтяжки линии DP.
DMPULLDOWN = сигнал от ядра к PHY t включения/выключения резистора подтяжки линии DM.

1. A-устройство посылает дескриптор **SetFeature b_hnp_enable** для включения поддержки HNP. ACK от контроллера OTG_FS подтверждает сей факт. Программа ставит бит разрешения HNP устройства в регистре управления OTG извещая о поддержке HNP.
   Программа ставит бит запроса HNP в регистре управления OTG извещая контроллер о необходимости запуска HNP.
2. По завершению работы шины A-устройство останавливает порт в регистре Управления портом хоста. После 3 ms простоя шины контроллер OTG_FS ставит бит прерывания Раннего останова в регистре прерываний ядра. Затем контроллер OTG_FS ставит бит останова USB в регистре прерываний ядра.
   Контроллер OTG_FS отключается и A-устройство обнаруживает на шине SE0, показывающий HNP. Контроллер OTG_FS включает подтяжку DP и DM в PHY, подтверждая роль хоста. A-устройство в ответ включает свой резистор подтяжки OTG_FS_DP в течении 3 ms обнаружения SE0. Контроллер OTG_FS понимает это как подключение.
   Контроллер OTG_FS ставит прерывание изменения состояния беседы хоста в регистре состояния прерываний OTG, показывая статус HNP. Программа определяет его по биту в регистре Управления OTG. Текущий режим выясняют в регистре `OTG_HS_GINTSTS`.
3. Программа ставит бит сброса `PRST` в `OTG_HS_HPRT` и контроллер OTG_FS выдаёт сброс USB и выполняет переучёт A-устройства.
4. По завершению работы хостом контроллер OTG_FS останавливает шину записью бита останова порта в регистр Управления портом хоста.
5. В режиме Беседы, когда A-устройство обнаруживает останов, оно отключается и переключается в хост. Контроллер OTG_FS отключает резисторы подтверждая роль устройства.
6. Текущий режим читают в регистре `OTG_HS_GINTSTS`.
7. Контроллер OTG_FS подключается, завершая процесс HNP.

36. Контроллер статической памяти (FSMC)

36.1. Основные свойства FSMC

Блок FSMC это интерфейс с синхронной и асинхронной памятью и 16-бит PC Card. Основное назначение:

- Трансляция передач AHB в протокол внешних устройств
- Согласование временных требований внешних устройств

Все виды внешней памяти используют одни адреса, данные и управляющие сигналы с контроллером. Доступ FSMC выполняется по одному обращению за раз к выбранному внешнему устройству.

Основные свойства FSMC:

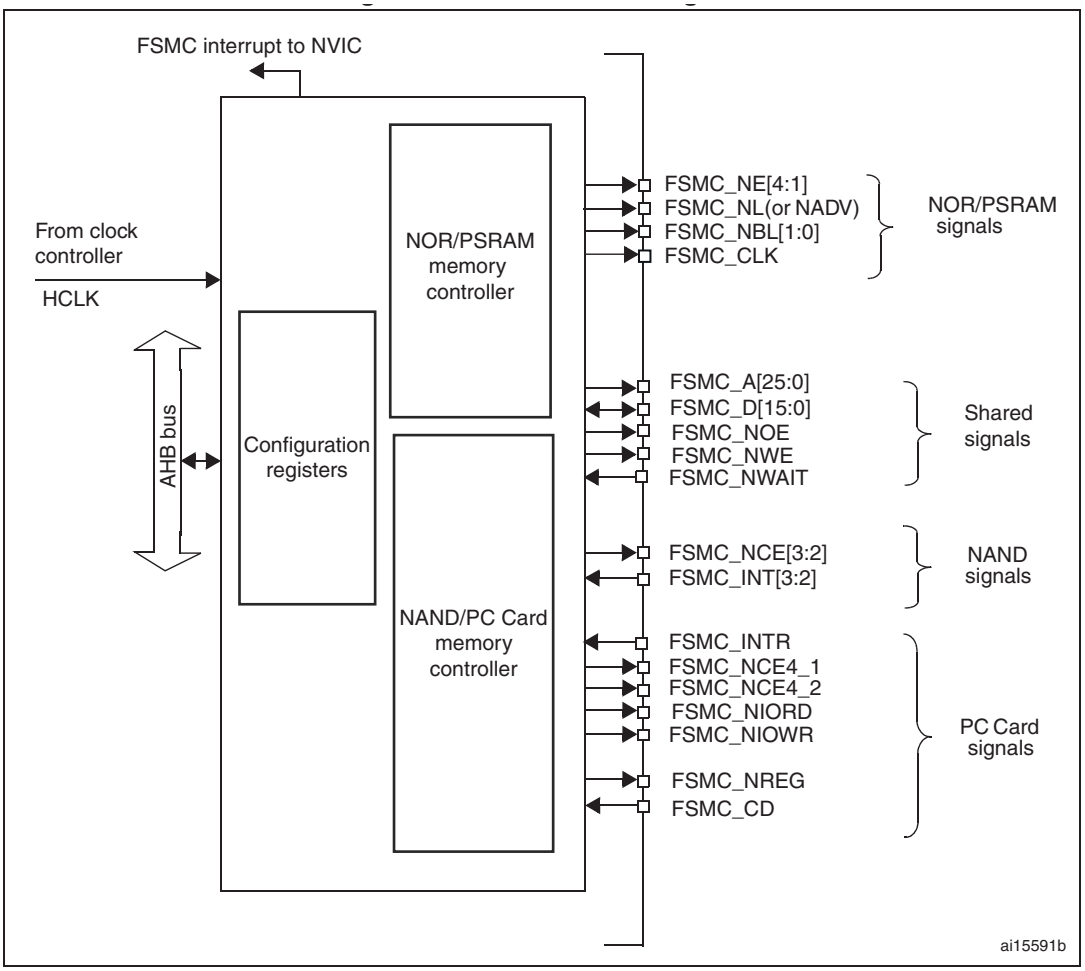
- Интерфейс со статическими устройствами:
  - Статическая память (SRAM)
  - NOR Flash память
  - PSRAM (4 банка памяти)
- Два банка NAND Flash с коррекцией ошибок до 8 КБ данных
- 16-бит PC Card совместимые устройства
- Поддерживает пакетный режим доступа синхронных устройств (NOR Flash и PSRAM)
- Шина данных 8- или 16-бит ширины
- Независимый выбор чипа для каждого банка памяти
- Независимая конфигурация каждого банка памяти
- Программируемые времянки для множества устройств:
  - Программируемые состояния ожидания (до 15)
  - Программируемые циклы реверсных передач шины (до 15)
  - Программируемые задержки разрешения вывода и записи (до 15)
  - Независимые времянки чтения/записи и протоколов
- Выходы разрешения записи и выбора байтовых линий для устройств PSRAM и SRAM
- Трансляция 32-бит передач AHB в последовательный 16- или 8-бит доступ внешних устройств
- FIFO записи из двух 32-бит слов для хранения данных (не адресов) как буфер пакетных передач записи AHB. Так можно писать в медленную память, быстро освобождая AHB для других операций. Буферизуется только один пакет: при появлении нового пакета или отдельной передачи при незавершённой операции AHB сливается FIFO. FSMC вставит состояния ожидания до завершения текущего доступа.
- Внешнее управление асинхронным ожиданием

Регистры описания устройств FSMC обычно пишутся во время загрузки и не меняются до сброса. Но их можно поменять в любое время.

36.2. Блок-схема FSMC

FSMC состоит из четырёх основных блоков:

- Интерфейс AHB (включая регистры конфигурации FSMC)
- Контроллер NOR Flash/PSRAM
- Контроллер NAND Flash/PC Card
- Интерфейс внешних устройств



36.3. Интерфейс AHB

Интерфейс ведомого AHB позволяет CPU и другим ведущим устройствам на шине обращаться к внешней статической памяти.

Передачи AHB транслируются в протокол внешнего устройства. В частности, 32-бит передачи AHB разбиваются на последовательные 16- или 8-бит передачи. Chip Select удерживается низким для 32-бит выравненных передач или переключается между последовательными 32-бит невыравненными передачами.

FSMC выдаёт ошибку AHB:

- При обращении к не разрешённому банку FSMC
- При обращении к банку NOR Flash со сброшенным битом `FACCEN` в регистре `FSMC_BCRx`.
- При обращении к банку PC Card с низким сигналом на входе `FSMC_CD` (Card Presence Detection).

Действие ошибки AHB зависит от ведущего на шине:

- Если это Cortex-M4 CPU, то выдаётся тяжёлый сбой
- Если это DMA, то выдаётся ошибка DMA и этот канал автоматически отключается.

FSMC тактируется от AHB (HCLK).

36.3.1. Поддерживаемая память и передачи

Общие правила передачи

Передачи данных AHB могут быть шириной 8-, 16- или 32-бита, а у внешних устройств она фиксированная. Это может привести к несовместимости.

Стало быть простые правила передачи таковы:

- Размер передачи AHB и данных в памяти равны. Проблем никаких.
- Размер передачи AHB больше размера памяти. FSMC разбивает передачу AHB на несколько меньших с размером данных внешней памяти памяти.
- Размер передачи AHB меньше размера памяти. Асинхронные передачи могут быть несовместимы в зависимости от типа внешнего устройства.

- Асинхронный доступ к устройствам с выбором байтов (SRAM, ROM, PSRAM).
  a) FSMC дозволяет запись нужных данных с помощью байтовых линий NBL[1:0]
  b) Чтение разрешено. Читаются все байты и ненужные отбрасываются. Линии NBL[1:0] удерживаются низкими.
- Асинхронный доступ к устройствам без выбора байтов (NOR и 16-бит NAND Flash). Эта ситуация возникает при байтовом доступе к 16-бит Flash памяти. Понятно, байтовый доступ невозможен, так что:
  a) Запись невозможна.
  b) Чтение разрешено. Читаются все байты и ненужные отбрасываются. Линии NBL[1:0] удерживаются низкими.

Регистры конфигурации

Установки регистров FSMC описаны *ниже*.

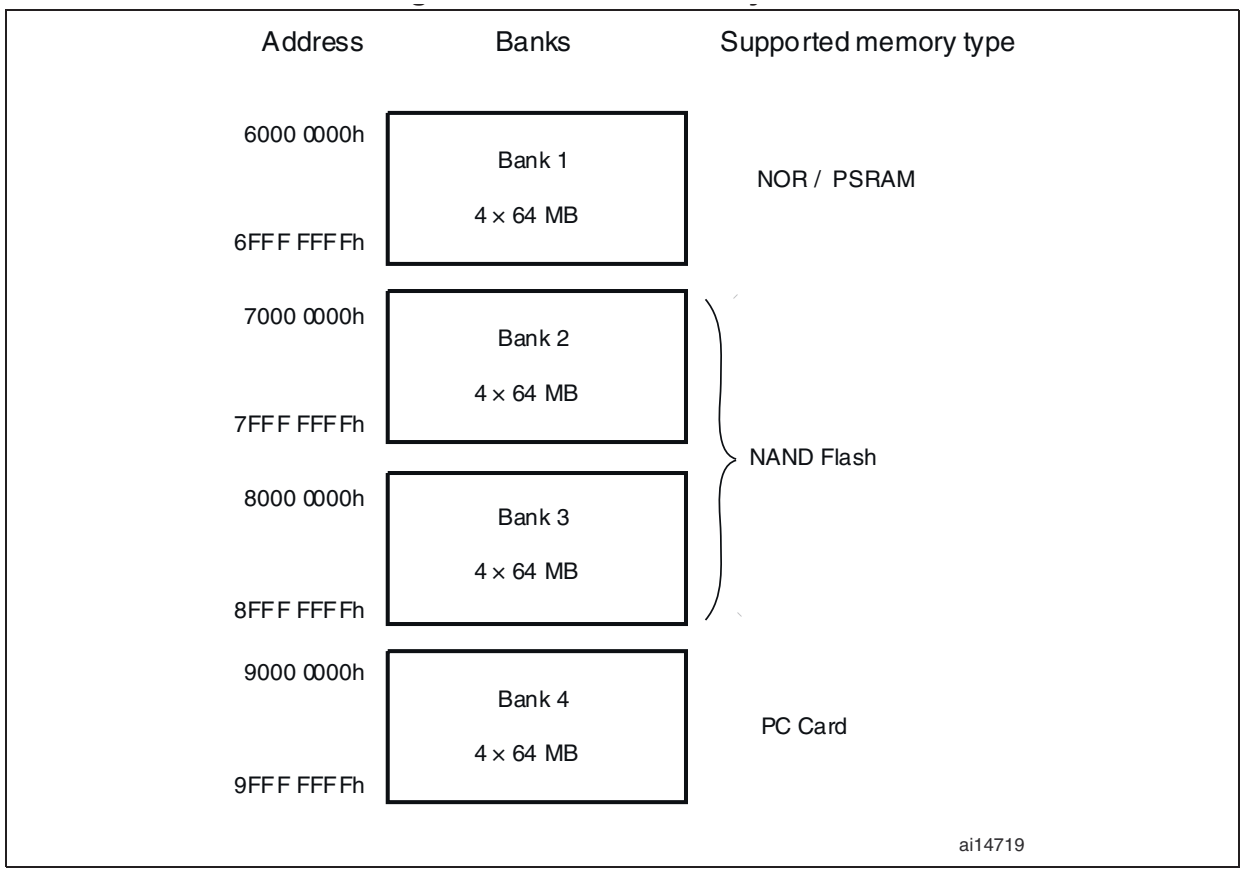
36.4. Отображение адресов внешних устройств

С точки зрения FSMC внешняя память разделяется на 4 банка по 256 Мбайт:

- Банк 1 адресует до 4 NOR Flash или PSRAM устройств памяти. Он разделён на 4 подбанка NOR/ PSRAM четырьмя с 4 выделенными Chip Selects:
  - Банк 1 - NOR/PSRAM 1
  - Банк 1 - NOR/PSRAM 2
  - Банк 1 - NOR/PSRAM 3
  - Банк 1 - NOR/PSRAM 4
- Банки 2 и 3 адресуют устройства NAND Flash (по одному на банк)
- Банк 4 адресует устройства PC Card

Тип каждого банка определяется в регистре конфигурации.

Банки памяти FSMC.



36.4.1. Адресация NOR/PSRAM

Биты HADDR[27:26] выбирают банк памяти:

HADDR[27:26](1)	Банк
00	Банк 1 - NOR/PSRAM 1
01	Банк 1 - NOR/PSRAM 2
10	Банк 1 - NOR/PSRAM 3
11	Банк 1 - NOR/PSRAM 4

1. HADDR это внутренние линии адреса AHB, транслируемые во внешнюю память.

Биты HADDR[25:0] содержат адрес внешней памяти. В таблице ниже приведено изменение HADDR при использовании разных типов памяти.

Ширина памяти(1)	Адрес в памяти	Максимальная ёмкость памяти (бит)
8 бит	HADDR[25:0]	64 Mbyte x 8 = 512 Mbit
16 бит	HADDR[25:1] >> 1	64 Mbyte/2 x 16 = 512 Mbit

1. При 16-бит внешней памяти FSMC для адреса внешней памяти FSMC_A[24:0] использует HADDR[25:1]. При ширине памяти (16-бит или 8-бит) FSMC_A[0] надо подключать к адресу внешней памяти A[0].

Поддержка закольцевания NOR Flash/PSRAM.

Кольцевой пакетный режим синхронной памяти не поддерживается. Память нужно ставить в линейный пакетный режим с неопределённой длиной.

36.4.2. Адресация NAND/PC Card

В этом случае доступны три банка, поделённые на пространства.

Адрес начала	Адрес конца	Банк FSMC	Пространство	Регистр времянки
0x9C00 0000	0x9FFF FFFF	Банк 4 - PC card	I/O	FSMC_PIO4 (0xB0)
0x9800 0000	0x9BFF FFFF		Attribute	FSMC_PATT4 (0xAC)
0x9000 0000	0x93FF FFFF		Common	FSMC_PMEM4 (0xA8)
0x8800 0000	0x8BFF FFFF	Банк 3 - NAND Flash	Attribute	FSMC_PATT3 (0x8C)
0x8000 0000	0x83FF FFFF		Common	FSMC_PMEM3 (0x88)
0x7800 0000	0x7BFF FFFF	Банк 2- NAND Flash	Attribute	FSMC_PATT2 (0x6C)
0x7000 0000	0x73FF FFFF		Common	FSMC_PMEM2 (0x68)

В NAND Flash памяти пространства общей памяти и памяти атрибутов подразделяются на три секции, размещённые в младших 256 КБ:

- Секция данных (первые 64 КБ в пространстве памяти общей/атрибутов)
- Секция команд (вторые 64 КБ в пространстве памяти общей/атрибутов)
- Секция адресов (следующие 128 КБ в пространстве памяти общей/атрибутов)

Имя секции	HADDR[17:16]	Диапазон адресов
Секция адресов	1X	0x020000-0x03FFFF
Секция команд	01	0x010000-0x01FFFF
Секция данных	00	0x000000-0x0FFFF

При доступе к NAND Flash памяти используются 3 секции:

- **Для посылки команд**: используется секция команд.
- **Для задания адресов чтения и записи**: используется секция адресов. Поскольку адрес может быть 4 или 5 байтов длиной, то требуются несколько последовательных циклов записи.
- **Для чтения и записи данных**: используется секция данных.

Так как NAND Flash память автоматически инкрементирует адреса, то для последовательного доступа к памяти данных адреса доступа изменять не надо.

36.5. Контроллер NOR Flash/PSRAM

FSMC выдаёт нужную времянку сигналов для следующих типов памяти:

- Асинхронные SRAM и ROM
  - 8-бит
  - 16-бит
  - 32-бит
- PSRAM (Сотовая RAM)
  - Асинхронный режим
  - Пакетный режим синхронного доступа
- NOR Flash
  - Асинхронный режим
  - Пакетный режим синхронного доступа
  - Мультиплексированный или немультиплексированный

FSMC выдаёт раздельные сигналы выбора на банк `NE[4:1]`. Все другие сигналы (адреса, данные и управление) общие.

При синхронном доступе чтении/записи FSMC подаёт на выбранное внешнее устройство такты (CLK), кратные HCLK. Размер банков фиксированный (64 МБ).

Банки конфигурируются специально выделенными регистрами (см. *Секцию 21.5.6*).

Параметры программируемой памяти включают времянку доступа и поддержку управления ожиданием (для PSRAM и NOR Flash в пакетном режиме).

Параметры доступа программируемой NOR/PSRAM.

Параметр	Функция	Режим доступа	Единицы	Min.	Max.
Установка адреса	Длительность фазы установки адреса	Асинхронный	Такты AHB (HCLK)	1	16
Удержание адреса	Длительность фазы удержания адреса	Асинхронный, мультиплекс. В/Выв.	Такты AHB (HCLK)	2	16
Установка данных	Длительность фазы установки данных	Асинхронный	Такты AHB (HCLK)	2	256
Реверс шины	Длительность фазы реверса шины	Асинхронный и синхронные чтение/запись	Такты AHB (HCLK)	1	16
Делитель частоты	Число тактов AHB (HCLK) на один такт памяти (CLK)	Синхронный	Такты AHB (HCLK)	2	16
Задержка данных	Число тактов, подаваемых памяти перед первым данным пакета	Синхронный	Такты памяти (CLK)	2	17

36.5.1. Сигналы интерфейса внешней памяти

Префикс "N" указывает на низкий активный уровень сигнала. NOR Flash и PSRAM адресуются 16-бит словами. Максимальная ёмкость 512 Мбит (26 линий адреса).

NOR Flash, немультиплексированный Вв./Выв.

Сигнал FSMC	Вв./Выв.	Функция
CLK	Выв.	Такты (для синхронного доступа)
A[25:0]	Выв.	Шина адреса
D[15:0]	Вв./Выв.	Двунаправленная шина данных

Сигнал FSMC	Вв./Выв.	Функция
NE[x]	Выв.	Выбор чипа, x = 1..4
NOE	Выв.	Разрешение выхода
NWE	Выв.	Разрешение записи
NL(=NADV)	Выв.	Разрешение защёлки (в некоторых NOR Flash называется достоверным адресом, NADV)
NWAIT	Вв.	Входной сигнал ожидания от NOR Flash

NOR Flash, мультиплексированный Вв./Выв.

Сигнал FSMC	Вв./Выв.	Функция
CLK	Выв.	Такты (для синхронного доступа)
A[25:16]	Выв.	Шина адреса
AD[15:0]	Вв./Выв.	16-бит мультиплексированная, двунаправленная шина адреса/данных
NE[x]	Выв.	Выбор чипа, x = 1..4
NOE	Выв.	Разрешение выхода
NWE	Выв.	Разрешение записи
NL(=NADV)	Выв.	Разрешение защёлки (в некоторых NOR Flash называется достоверным адресом, NADV)
NWAIT	Вв.	Входной сигнал ожидания от NOR Flash

PSRAM/SRAM, немультиплексированный Вв./Выв.

Сигнал FSMC	Вв./Выв.	Функция
CLK	Выв.	Такты (для синхронного доступа)
A[25:0]	Выв.	Шина адреса
D[15:0]	Вв./Выв.	Двунаправленная шина данных
NE[x]	Выв.	Выбор чипа, x = 1..4 (называемая NCE в PSRAM (Cellular RAM т.е. CRAM))
NOE	Выв.	Разрешение выхода
NWE	Выв.	Разрешение записи
NL(=NADV)	Выв.	Достоверный адрес ввода в PSRAM (имя в памяти NADV)
NWAIT	Вв.	Входной сигнал ожидания от PSRAM
NBL[1]	Выв.	Разрешение старшего байта (имя в памяти NUB)
NBL[0]	Выв.	Разрешение младшего байта (имя в памяти NLB)

36.5.2. Поддерживаемая память и передачи

Таблица ниже даёт пример поддерживаемых устройств, режимов доступа и передач с 16-бит шиной данных для NOR, PSRAM и SRAM. Недоступные (или неподдерживаемые) передачи FSMC выделены красным цветом.

Уст-во	Режим	Ч/З	Размер данных AHB	Размер данных памяти	Можно	Комментарии
NOR Flash (мультипл. и немультипл.)	Асинхр.	Ч	8	16	Д	—
	Асинхр.	З	8	16	Н	—
	Асинхр.	Ч	16	16	Д	—
	Асинхр.	З	16	16	Д	—
	Асинхр.	Ч	32	16	Д	За два обращения FSMC
	Асинхр.	З	32	16	Д	За два обращения FSMC
	Асинхр. страничн.	Ч	-	16	Н	Не поддерживается
	Синхр.	Ч	8	16	Н	—
	Синхр.	Ч	16	16	Д	—
	Синхр.	Ч	32	16	Д	—
PSRAM (мультипл. и немультипл.)	Асинхр.	Ч	8	16	Д	—
	Асинхр.	З	8	16	Д	Используются линии байтов NBL[1:0]
	Асинхр.	Ч	16	16	Д	—
	Асинхр.	З	16	16	Д	—
	Асинхр.	Ч	32	16	Д	За два обращения FSMC
	Асинхр.	З	32	16	Д	Используются линии байтов NBL[1:0]
	Асинхр. страничн.	Ч	-	16	Н	Не поддерживается
	Синхр.	Ч	8	16	Н	—
	Синхр.	Ч	16	16	Д	—
	Синхр.	Ч	32	16	Д	—
	Синхр.	З	8	16	Д	Используются линии байтов NBL[1:0]
	Синхр.	З	16/32	16	Д	—
SRAM и ROM	Асинхр.	Ч	8/16	16	Д	—
	Асинхр.	З	8/16	16	Д	Используются линии байтов NBL[1:0]
	Асинхр.	Ч	32	16	Д	За два обращения FSMC
	Асинхр.	З	32	16	Д	За два обращения FSMC. Используются линии байтов NBL[1:0]

36.5.3. Синхронизация сигналов

- Все выходные сигналы контроллера изменяются по переднему фронту HCLK
- В синхронном режиме (чтение и запись), все выходные сигналы изменяются по переднему фронту HCLK. Независимо от значения CLKDIV все выходы меняются так:
  - NOEL/NWEL/NEL/NADVL/NADVH/NBLL/Адрес_Достоверен меняются по заднему фронту FSMC_CLK.
  - NOEH/NWEH/NEH/NOEH/NBLH/Адрес_Недостоверен меняются по переднему фронту FSMC_CLK.

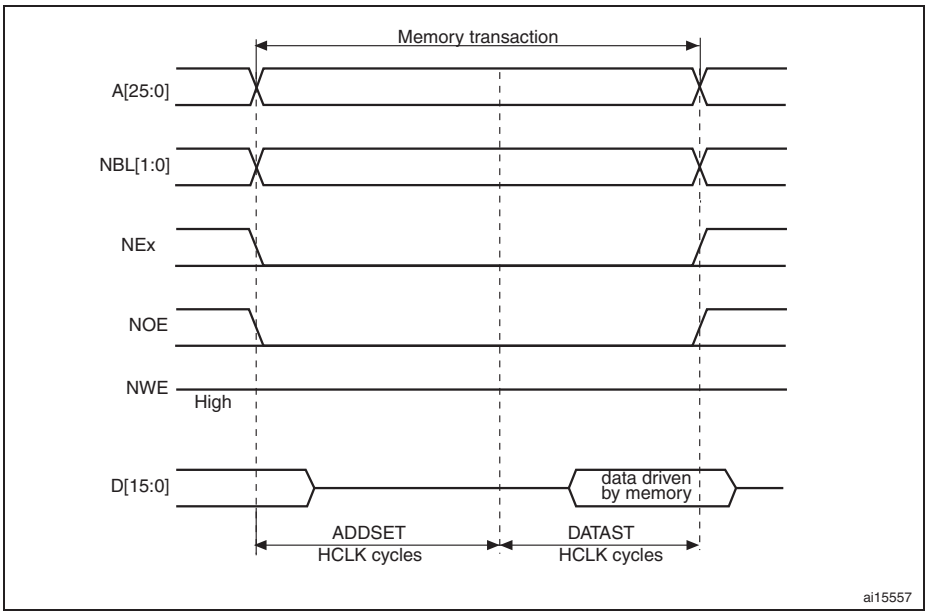
36.5.4. Асинхронные передачи контроллера NOR Flash/PSRAM

Асинхронная статическая память (NOR Flash, PSRAM, SRAM)

- Сигналы синхронизированы внутренними тактами HCLK, на память они не передаются.
- FSMC всегда считывает данные перед снятием сигналов NOE. Так соблюдается время удержания данных памятью.
- Если разрешён расширенный режим (стоит бит `EXTMOD` в регистре `FSMC_BCRx`), то доступны до 4 расширенных режима (A, B, C и D). При чтении и записи их можно смешивать. Например, читать можно в режиме A, а писать в режиме B.
- Если расширенный режим отключён (бит `EXTMOD` в регистре `FSMC_BCRx` сброшен), то FSMC может работать в режиме 1 или 2:
  - С памятью типа SRAM/PSRAM (`MTYP[0:1]` = `0x0` или `0x01` в `FSMC_BCRx`) по умолчанию используется Режим 1.
  - С памятью типа NOR (`MTYP[0:1]` = `0x10` в `FSMC_BCRx`) по умолчанию используется Режим 2.

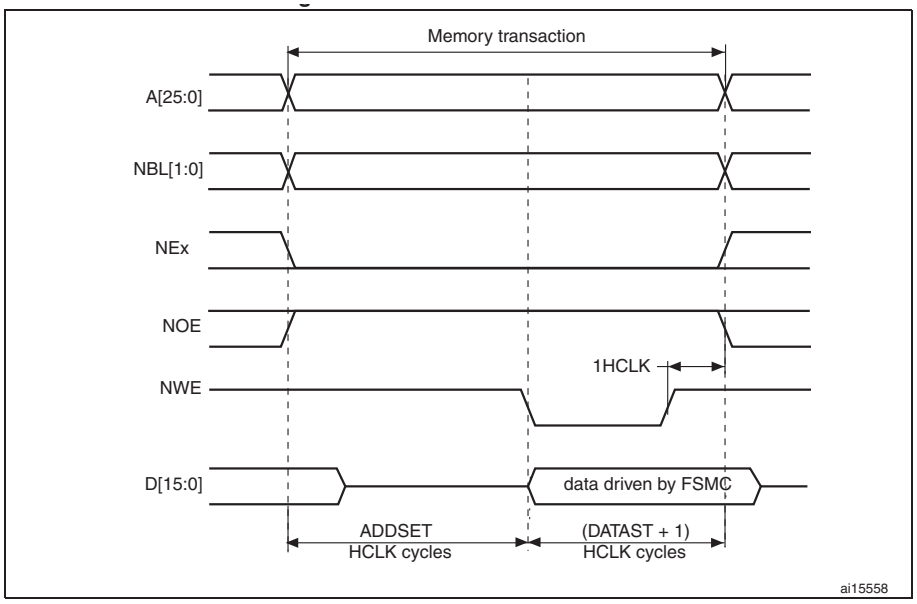
Режим 1 - SRAM/PSRAM (CRAM)

Режим 1, чтение.



1. При чтении NBL[1:0] удерживается низким.

Режим 1, запись.



Один такт HCLK в конце записи гарантирует удержание адреса и данных после снятия сигнала NWE. Из-за этого значение DATAST должно быть больше нуля (DATAST > 0).

Таблица 226. Биты FSMC_BCRx.

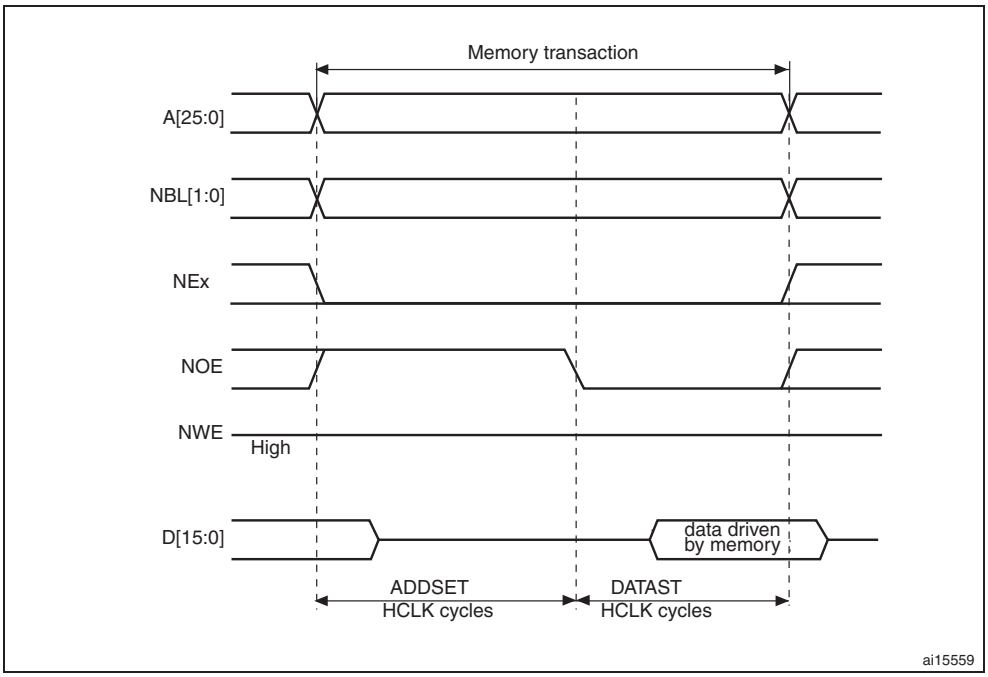
Номер	Имя	Устанавливаемое значение
31-20	Резерв	0x000
19	CBURSTRW	0x0 (в асинхронном режиме не влияет)
18:16	CPSIZE	0x0 (в асинхронном режиме не влияет)
15	ASYNCWAIT	1 при поддержке памятью. Иначе 0.
14	EXTMOD	0x0
13	WAITEN	0x0 (в асинхронном режиме не влияет)
12	WREN	Как нужно
11	WAITCFG	Всё равно.
10	WRAPMOD	0x0
9	WAITPOL	Имеет смысл только при стоящем бите 15
8	BURSTEN	0x0
7	Резерв	0x1
6	FACCEN	Всё равно.
5-4	MWID	Как нужно
3-2	MTYP[0:1]	Как нужно, исключая 0x2 (NOR Flash)
1	MUXEN	0x0
0	MBKEN	0x1

Таблица 227. Биты FSMC_BTRx.

Номер	Имя	Устанавливаемое значение
31-30	Резерв	0x0
29-28	ACCMOD	Всё равно.
27-24	CPSIZE	Всё равно.
23-20	CLKDIV	Всё равно.
19-16	BUSTURN	Время между высоким и низким NEx (BUSTURN HCLK)
15-8	DATAST	Длительность второй фазы доступа (DATAST+1 циклов HCLK при записи, DATAST+3 циклов HCLK при чтении). Не может быть равным 0 (минимум 1).
7-4	ADDHLD	Всё равно.
3-0	ADDSET[3:0]	Длительность первой фазы доступа (ADDSET+1 циклов HCLK).

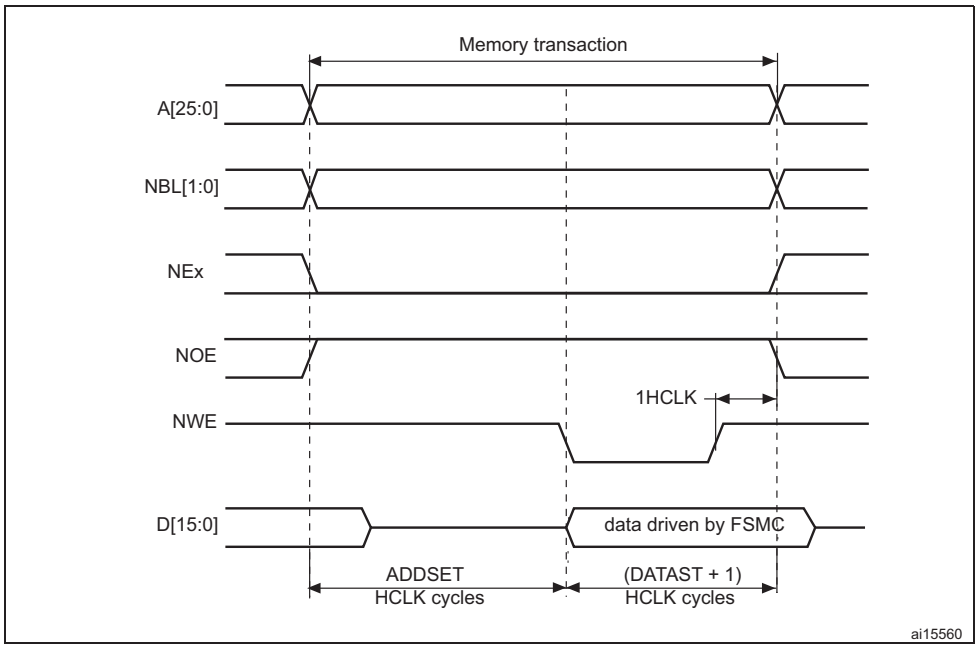
Режим A - SRAM/PSRAM (CRAM), переключение OE.

Режим A, чтение.



1. При чтении NBL[1:0] удерживается низким.

Режим A, запись.



В отличие от Режима 1 сигнал NOE переключается и времена чтения и записи независимые.

Таблица 228. Биты FSMC_BCRx.

Номер	Имя	Устанавливаемое значение
31-20	Резерв	0x000
19	CBURSTRW	0x0 (в асинхронном режиме не влияет)
18:16	CPSIZE	0x0 (в асинхронном режиме не влияет)
15	ASYNCWAIT	1 при поддержке памятью. Иначе 0.
14	EXTMOD	0x1
13	WAITEN	0x0 (в асинхронном режиме не влияет)
12	WREN	Как нужно
11	WAITCFG	Всё равно.

Номер	Имя	Устанавливаемое значение
10	WRAPMOD	0x0
9	WAITPOL	Имеет смысл только при стоящем бите 15
8	BURSTEN	0x0
7	Резерв	0x1
6	FACCEN	Всё равно.
5-4	MWID	Как нужно
3-2	MTYP[0:1]	Как нужно, исключая 0x2 (NOR Flash)
1	MUXEN	0x0
0	MBKEN	0x1

Таблица 229. Биты FSMC_BTRx.

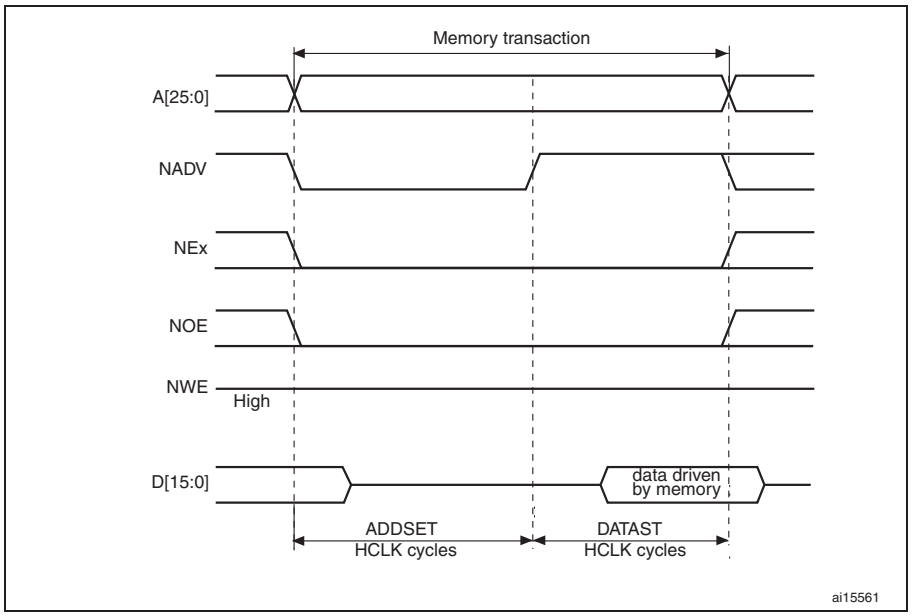
Номер	Имя	Устанавливаемое значение
31-30	Резерв	0x0
29-28	ACCMOD	0x0
27-24	CPSIZE	Всё равно.
23-20	CLKDIV	Всё равно.
19-16	BUSTURN	Время между высоким и низким NEx (BUSTURN HCLK)
15-8	DATAST	Длительность второй фазы доступа (DATAST+3 циклов HCLK при чтении). Не может быть равным 0 (минимум 1).
7-4	ADDHLD	Всё равно.
3-0	ADDSET[3:0]	Длительность первой фазы доступа (ADDSET+1 циклов HCLK).

Таблица 230. Биты FSMC_BWTRx.

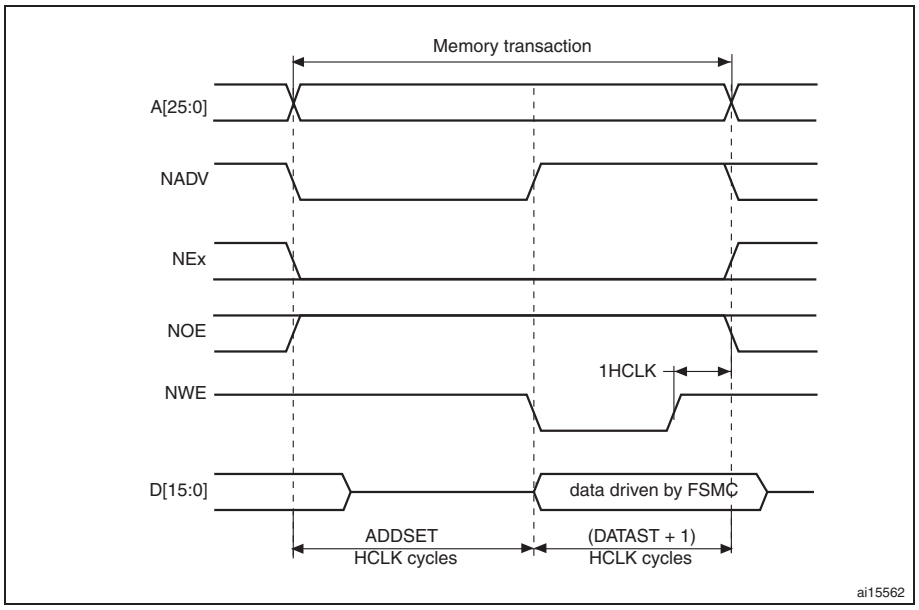
Номер	Имя	Устанавливаемое значение
31-30	Резерв	0x0
29-28	ACCMOD	0x0
27-24	DATLAT	Всё равно.
23-20	CLKDIV	Всё равно.
19-16	BUSTURN	Время между высоким и низким NEx (BUSTURN HCLK)
15-8	DATAST	Длительность второй фазы доступа (DATAST+1 циклов HCLK при записи, DATAST+3 циклов HCLK при чтении). Не может быть равным 0 (минимум 1).
7-4	ADDHLD	Всё равно.
3-0	ADDSET[3:0]	Длительность первой фазы доступа (ADDSET+1 циклов HCLK для записи).

Режим 2/B - NOR Flash

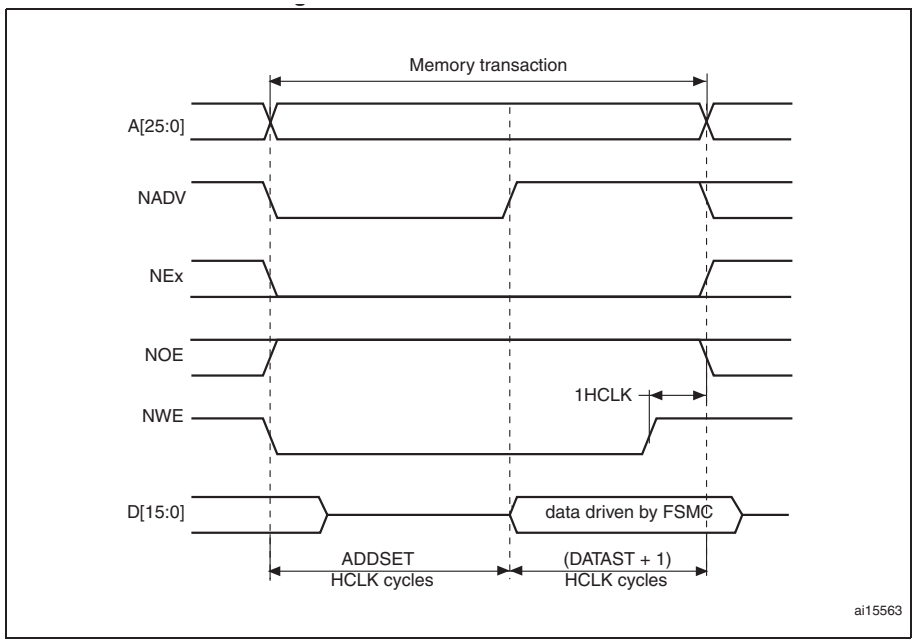
Режим 2/B, чтение.



Режим 2, запись.



Режим B, запись.



Отличие с Режимом 1 в переключении NWE и независимыми временами чтения и записи в расширенном режиме (Mode B).

Таблица 231. Биты FSMC_BCRx.

Номер	Имя	Устанавливаемое значение
31-20	Резерв	0x000
19	CBURSTRW	0x0 (в асинхронном режиме не влияет)
18:16	CPSIZE	0x0 (в асинхронном режиме не влияет)
15	ASYNCWAIT	1 при поддержке памятью. Иначе 0.
14	EXTMOD	0x1 в Режиме B, 0x0 в Режиме 2
13	WAITEN	0x0 (в асинхронном режиме не влияет)
12	WREN	Как нужно
11	WAITCFG	Всё равно.
10	WRAPMOD	0x0
9	WAITPOL	Имеет смысл только при стоящем бите 15
8	BURSTEN	0x0
7	Резерв	0x1
6	FACCEN	0x1
5-4	MWID	Как нужно
3-2	MTYP[0:1]	0x2 (NOR Flash)
1	MUXEN	0x0
0	MBKEN	0x1

Таблица 232. Биты FSMC_BTRx.

Номер	Имя	Устанавливаемое значение
31-30	Резерв	0x0
29-28	ACCMOD	0x1
27-24	DATLAT	Всё равно.
23-20	CLKDIV	Всё равно.
19-16	BUSTURN	Время между высоким и низким NEx (BUSTURN HCLK)
15-8	DATAST	Длительность второй фазы доступа (DATAST+3 циклов HCLK при чтении). Не может быть равным 0 (минимум 1).
7-4	ADDHLD	Всё равно.
3-0	ADDSET[3:0]	Длительность первой фазы доступа (ADDSET+1 циклов HCLK).

Таблица 233. Биты FSMC_BWTRx.

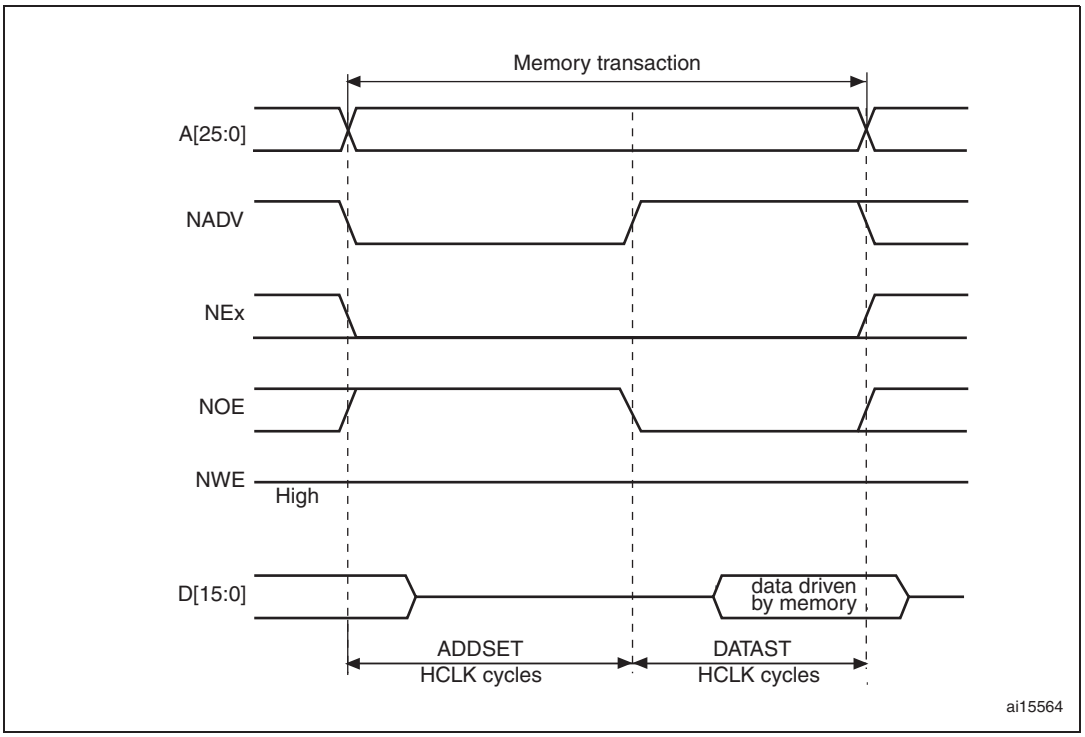
Номер	Имя	Устанавливаемое значение
31-30	Резерв	0x0
29-28	ACCMOD	0x1
27-24	DATLAT	Всё равно.
23-20	CLKDIV	Всё равно.
19-16	BUSTURN	Время между высоким и низким NEx (BUSTURN HCLK)

Номер	Имя	Устанавливаемое значение
15-8	DATAST	Длительность второй фазы доступа (DATAST+1 циклов HCLK при записи, DATAST+3 циклов HCLK при чтении). Не может быть равным 0 (минимум 1).
7-4	ADDHLD	Всё равно.
3-0	ADDSET[3:0]	Длительность первой фазы доступа (ADDSET+1 циклов HCLK для записи).

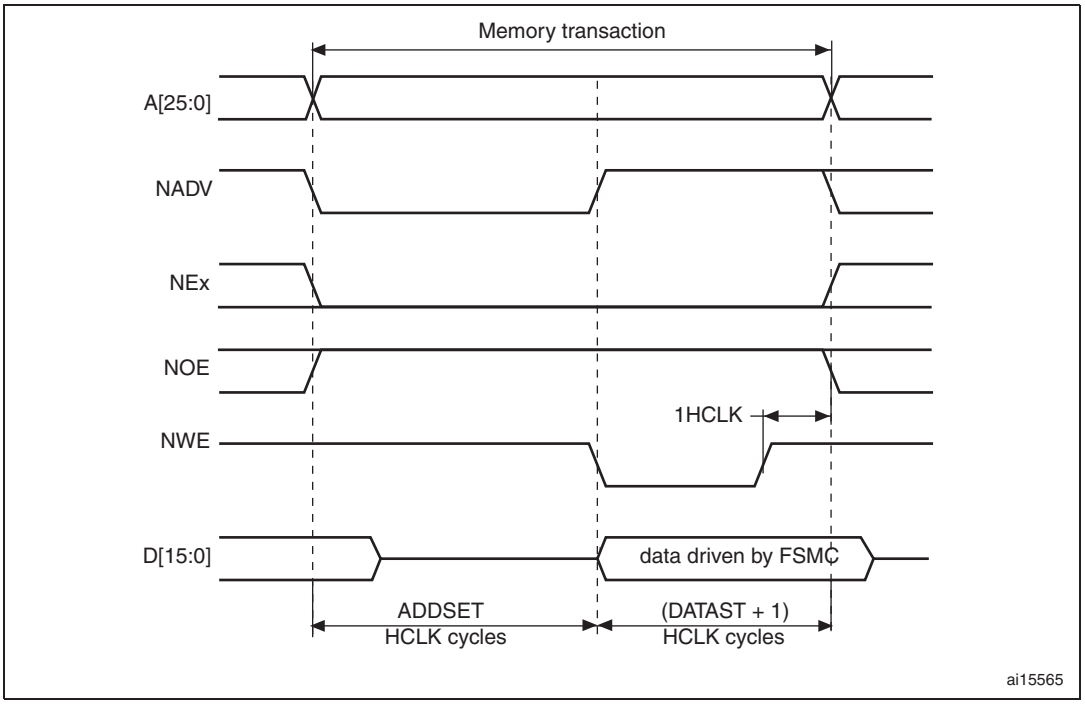
Регистр FSMC_BWTRx достоверен только в расширенном режиме B, иначе его содержимое бестолково.

Режим C - NOR Flash - переключение OE.

Режим C, чтение.



Режим C, запись.



От Режима 1 отличается переключением NOE и независимыми времянками чтения и записи.

Таблица 234. Биты FSMC_BCRx.

Номер	Имя	Устанавливаемое значение
31-20	Резерв	0x000
19	CBURSTRW	0x0 (в асинхронном режиме не влияет)
18:16	CPSIZE	0x0 (в асинхронном режиме не влияет)
15	ASYNCWAIT	1 при поддержке памятью. Иначе 0.
14	EXTMOD	0x1 в Режиме B, 0x0 в Режиме 2
13	WAITEN	0x0 (в асинхронном режиме не влияет)
12	WREN	Как нужно
11	WAITCFG	Всё равно.
10	WRAPMOD	0x0
9	WAITPOL	Имеет смысл только при стоящем бите 15
8	BURSTEN	0x0
7	Резерв	0x1
6	FACCEN	0x1
5-4	MWID	Как нужно
3-2	MTYP[0:1]	0x2 (NOR Flash)
1	MUXEN	0x0
0	MBKEN	0x1

Таблица 235. Биты FSMC_BTRx.

Номер	Имя	Устанавливаемое значение
31-30	Резерв	0x0
29-28	ACCMOD	0x2
27-24	DATLAT	0x0
23-20	CLKDIV	0x0
19-16	BUSTURN	Время между высоким и низким NEx (BUSTURN HCLK)
15-8	DATAST	Длительность второй фазы доступа (DATAST+3 циклов HCLK при чтении). Не может быть равным 0 (минимум 1).
7-4	ADDHLD	Всё равно.
3-0	ADDSET[3:0]	Длительность первой фазы доступа (ADDSET+1 циклов HCLK).

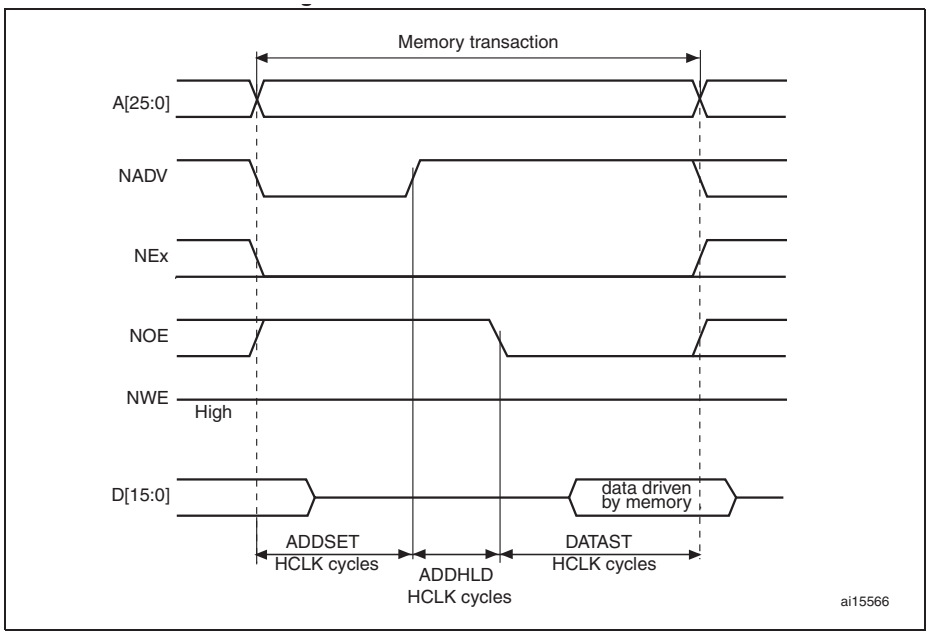
Таблица 236. Биты FSMC_BWTRx.

Номер	Имя	Устанавливаемое значение
31-30	Резерв	0x0
29-28	ACCMOD	0x2
27-24	DATLAT	Всё равно.
23-20	CLKDIV	Всё равно.
19-16	BUSTURN	Время между высоким и низким NEx (BUSTURN HCLK)

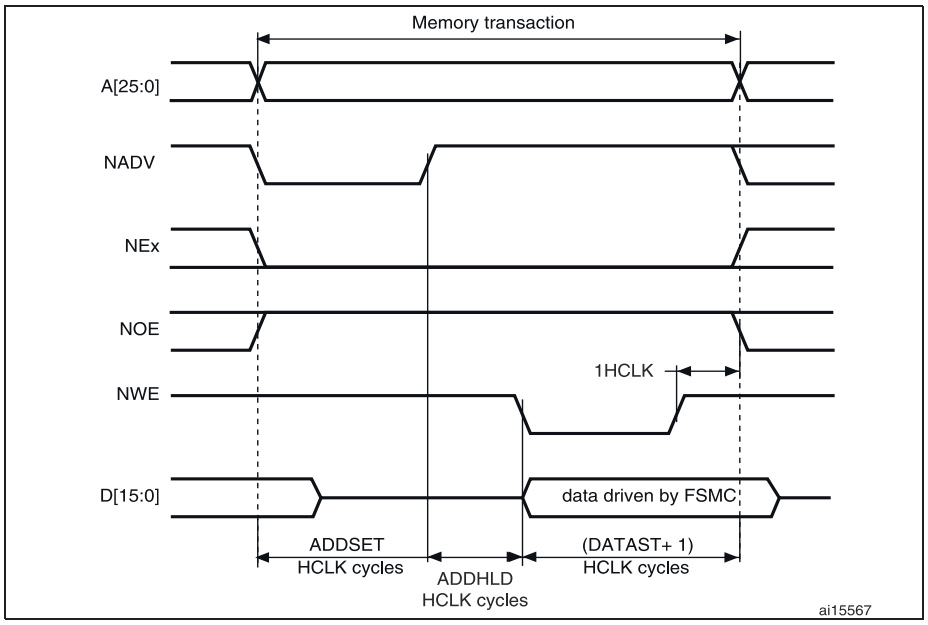
Номер	Имя	Устанавливаемое значение
15-8	DATAST	Длительность второй фазы доступа (DATAST+1 циклов HCLK при записи, DATAST+3 циклов HCLK при чтении). Не может быть равным 0 (минимум 1).
7-4	ADDHLD	Всё равно.
3-0	ADDSET[3:0]	Длительность первой фазы доступа (ADDSET+1 циклов HCLK для записи).

Режим D, асинхронный доступ с расширенным адресом.

Режим D, чтение.



Режим D, запись.



От режима 1 отличается переключением NOE после изменения NADV и независимыми времянками чтения и записи.

Таблица 237. Биты FSMC_BCRx.

Номер	Имя	Устанавливаемое значение
31-20	Резерв	0x000
19	CBURSTRW	0x0 (в асинхронном режиме не влияет)

Номер	Имя	Устанавливаемое значение
18:16	CPSIZE	0x0 (в асинхронном режиме не влияет)
15	ASYNCWAIT	1 при поддержке памятью. Иначе 0.
14	EXTMOD	0x1
13	WAITEN	0x0 (в асинхронном режиме не влияет)
12	WREN	Как нужно
11	WAITCFG	Всё равно.
10	WRAPMOD	0x0
9	WAITPOL	Имеет смысл только при стоящем бите 15
8	BURSTEN	0x0
7	Резерв	0x1
6	FACCEN	Соответственно памяти.
5-4	MWID	Как нужно
3-2	MTYP[0:1]	Как нужно
1	MUXEN	0x0
0	MBKEN	0x1

Таблица 238. Биты FSMC_BTRx.

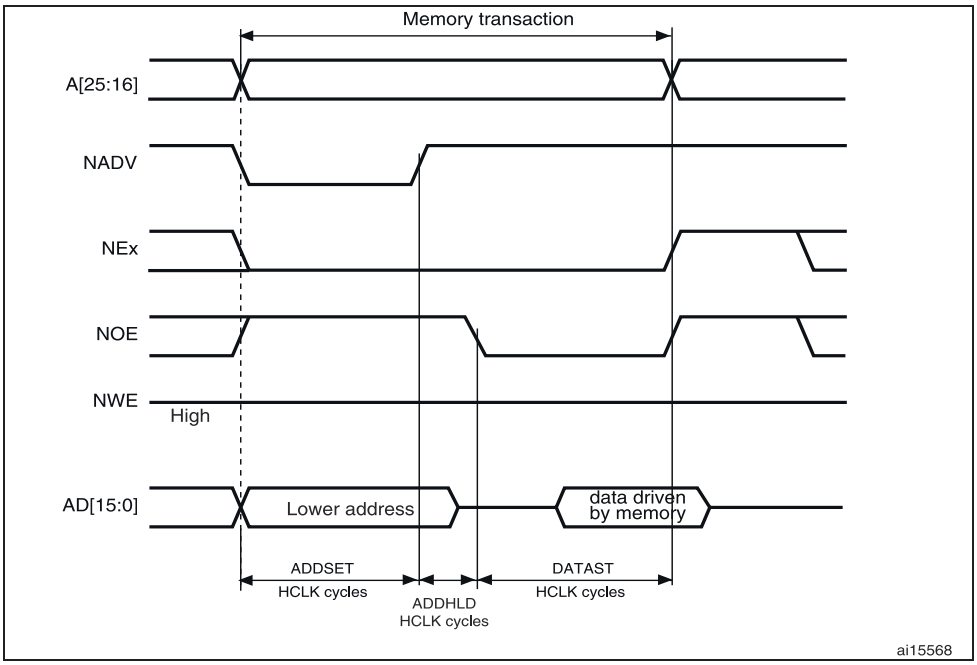
Номер	Имя	Устанавливаемое значение
31-30	Резерв	0x0
29-28	ACCMOD	0x3
27-24	DATLAT	Всё равно.
23-20	CLKDIV	Всё равно.
19-16	BUSTURN	Время между высоким и низким NEx (BUSTURN HCLK)
15-8	DATAST	Длительность второй фазы доступа (DATAST+3 циклов HCLK при чтении). Не может быть равным 0 (минимум 1).
7-4	ADDHLD	Длительность средней фазы чтения (ADDHLD+1 циклов HCLK).
3-0	ADDSET[3:0]	Длительность первой фазы доступа (ADDSET+1 циклов HCLK).

Таблица 239. Биты FSMC_BWTRx.

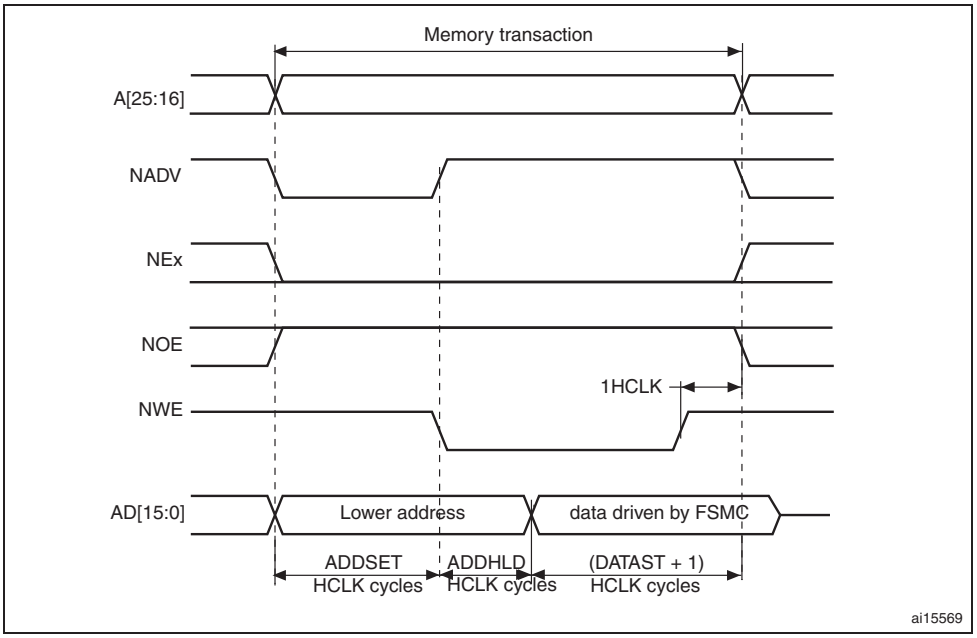
Номер	Имя	Устанавливаемое значение
31-30	Резерв	0x0
29-28	ACCMOD	0x3
27-24	DATLAT	0x0
23-20	CLKDIV	0x0
19-16	BUSTURN	Время между высоким и низким NEx (BUSTURN HCLK)
15-8	DATAST	Длительность второй фазы доступа (DATAST+3 циклов HCLK при записи). Не может быть равным 0 (минимум 1).
7-4	ADDHLD	Длительность средней фазы записи (ADDHLD+1 циклов HCLK).
3-0	ADDSET[3:0]	Длительность первой фазы доступа (ADDSET+1 циклов HCLK для записи).

Мультиплексный режим - мультиплексированный асинхронный доступ к NOR Flash памяти.

Мультиплексный режим, чтение.



Мультиплексный режим, запись.



От режима D отличается управлением младшей частью адреса на шине данных.

Таблица 240. Биты FSMC_BCRx.

Номер	Имя	Устанавливаемое значение
31-21	Резерв	0x000
19	CBURSTRW	0x0 (в асинхронном режиме не влияет)
18:16	CPSIZE	0x0 (в асинхронном режиме не влияет)
15	ASYNCWAIT	1 при поддержке памятью. Иначе 0.
14	EXTMOD	0x0
13	WAITEN	0x0 (в асинхронном режиме не влияет)
12	WREN	Как нужно

Номер	Имя	Устанавливаемое значение
11	WAITCFG	Всё равно.
10	WRAPMOD	0x0
9	WAITPOL	Имеет смысл только при стоящем бите 15
8	BURSTEN	0x0
7	Резерв	0x1
6	FACCEN	0x1
5-4	MWID	Как нужно
3-2	MTYP[0:1]	0x2 (NOR Flash memory)
1	MUXEN	0x0
0	MBKEN	0x1

Таблица 241. Биты FSMC_BTRx.

Номер	Имя	Устанавливаемое значение
31-30	Резерв	0x0
29-28	ACCMOD	0x0
27-24	DATLAT	Всё равно.
23-20	CLKDIV	Всё равно.
19-16	BUSTURN	Длительность последней фазы доступа (BUSTURN+1 HCLK)
15-8	DATAST	Длительность второй фазы доступа (DATAST+3 циклов HCLK при чтении, DATAST+1 циклов HCLK при записи). Не может быть равным 0 (минимум 1).
7-4	ADDHLD	Длительность средней фазы доступа (ADDHLD+1 циклов HCLK). Не может быть равным 0 (минимум 1).
3-0	ADDSET[3:0]	Длительность первой фазы доступа (ADDSET+1 циклов HCLK).

Управление ожиданием при асинхронном доступе.

Если асинхронная память умеет выставлять сигнал WAIT при своей неготовности к работе, то в регистре FSMC_BCRx надо ставить бит ASYNCWAIT.

Если сигнал WAIT активен (в зависимости от бита WAITPOL), то вторая фаза доступа (установка данных), определённая битами DATAST, удлиняется вплоть до снятия сигнала WAIT. В отличие от фазы данных, первые фазы доступа (Установка и удержание адреса), определённые битами ADDSET[3:0] и ADDHLD, к сигналу WAIT не чувствительны.

Фазу установки данных (DATAST в регистре FSMC_BCRx) нужно определять так, чтобы сигнал WAIT соответствовал следующим условиям:

1. Память выставляет сигнал WAIT выравненным к переключаемым NOE/NWE:
   $DATAST \geq (4 \times HCLK) + max\_wait\_assertion\_time$
2. Память выставляет сигнал WAIT выравненным к NEx (или NOE/NWE не переключаются):

если

$$max\_wait\_assertion\_time > фаза\_адреса + фаза\_удержания$$

то

$$DATAST \geq (4 \times HCLK) + (max\_wait\_assertion\_time - фаза\_адреса - фаза\_удержания)$$

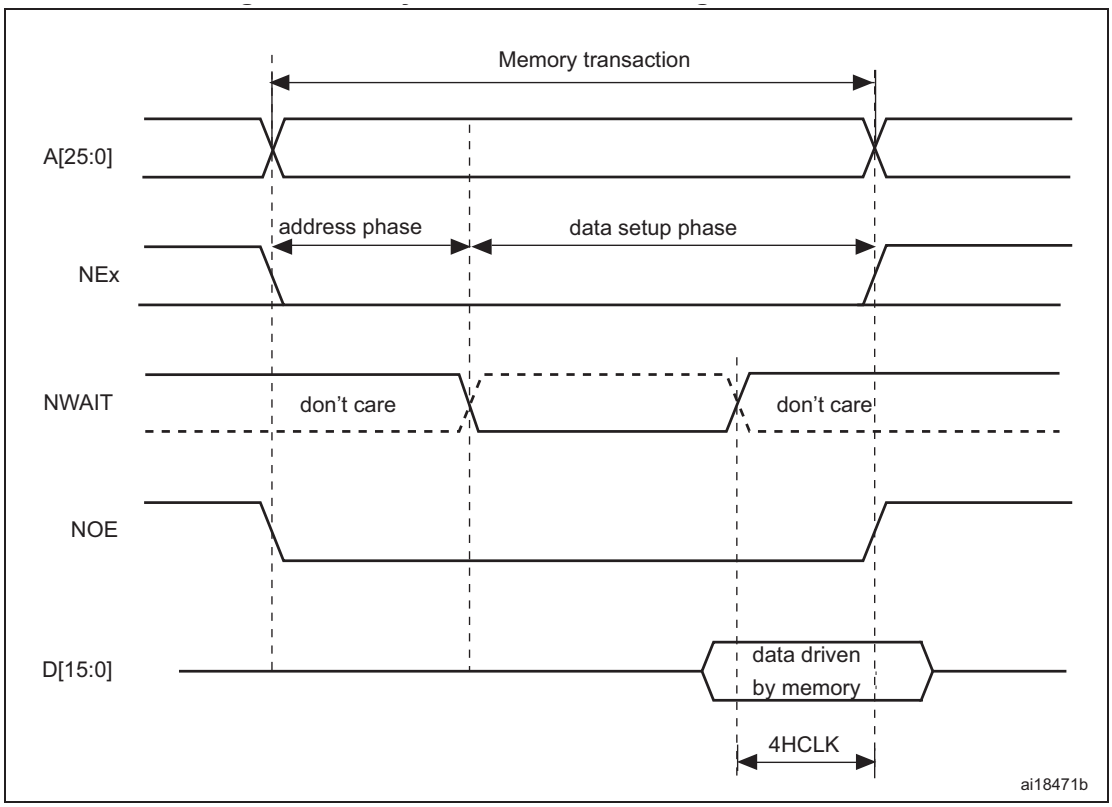
иначе

$$DATAST \geq 4 \times HCLK$$

где max_wait_assertion_time это максимальное время установки сигнала WAIT по низкому уровню на NEx/NOE/NWE.

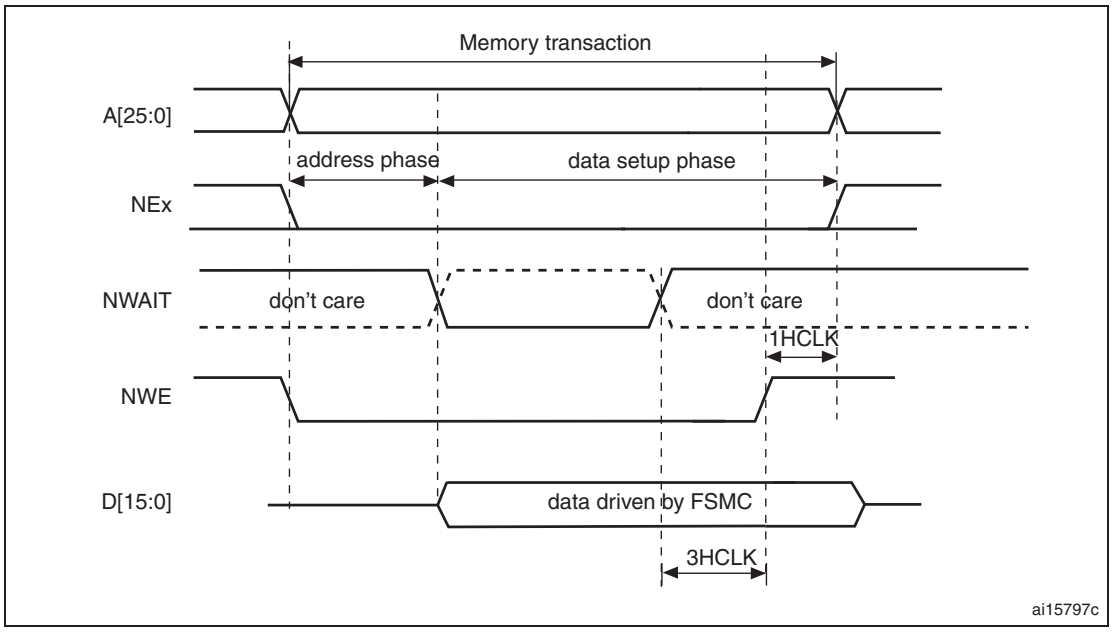
Рисунки ниже показывают число добавляемых тактов HCLK для доступа к памяти после снятия ею сигнала WAIT (независимо от приведённых случаев).

Асинхронное ожидание при чтении.



Полярность NWAIT зависит от бита WAITPOL в регистре FSMC_BCRx.

Асинхронное ожидание при записи.



Полярность NWAIT зависит от бита WAITPOL в регистре FSMC_BCRx.

36.5.5. Синхронные передачи

Кратность тактов памяти, CLK, и тактов HCLK задаётся параметром CLKDIV.

NOR Flash определяет минимальное время от выдачи NADV до переднего фронта CLK. Для его соблюдения FSMC не должен подавать такт на память во время первого цикла синхронного доступа (перед появлением NADV). Так гарантируется появление переднего фронта в середине импульса импульса NADV.

Задержки Данных и NOR Flash.

Задержка данных это число циклов ожидания перед считыванием данных. Значение DATLAT должно быть совместимым с значением в регистре конфигурации NOR Flash. При вычислении задержки данных FSMC не учитывает такты при низком NADV.

Внимание:

Некоторые устройства NOR Flash памяти не учитывают такты низкого NADV при подсчёте запаздывания данных. Так что точное соотношение запаздывания NOR Flash и параметром FMSC DATLAT может быть одним из двух:

- Запаздывание NOR Flash = (DATLAT + 2) тактов CLK
- Запаздывание NOR Flash = (DATLAT + 3) тактов CLK

Некоторые современные типы памяти выставляют NWAIT в фазе задержки. В таких случаях DATLAT может быть минимальным. В результате FSMC читает данные и ждёт достаточно долго для определения достоверности данных. Так FSMC определяет конец задержки памяти и достоверность данных.

Другие типы памяти во время запаздывания NWAIT не выставляют. В этом случае задержки и FSMC и памяти должны быть корректными, чтобы не терять данные в первой фазе доступа к памяти.

Однопакетная передача.

Если выбранный банк включён в пакетный синхронный доступ и, например, AHB запросил однопакетный доступ к 16-бит памяти, то FSMC выполняет пакетную передачу длиной 1 (при 16-бит передаче AHB) или длиной 2 (при 32-бит передаче AHB) и снимает выбор чипа после стробирования последних данных.

Понятно, что такая передача не самая эффективная по сравнению с асинхронным чтением по затратам тактов. Тем не менее, асинхронное чтение с произвольным доступом потребует установки режима доступа к памяти, что в итоге много длиннее.

Переход границы страниц в Cellular RAM 1.5

Cellular RAM 1.5 не разрешает пакетному доступу пересекать границу страниц. FSMC позволяет автоматически автоматически разбивать пакетный доступ при достижении границы страниц установкой битов `CPSIZE` в регистре `FSMC_BCR1` следуя размеру страницы.

Управление ожиданием

Для синхронной NOR Flash памяти состояние `NWAIT` определяется после заданного периода задержки, (DATLAT+2) тактов CLK.

Если `NWAIT` активен (низкий при `WAITPOL` = 0, высокий при `WAITPOL` = 1), то вставляются такты ожидания до снятия NWAIT (высокий при `WAITPOL` = 0, низкий при `WAITPOL` = 1).

Если `NWAIT` неактивен, данные считаются достоверными либо немедленно (бит `WAITCFG` = 1), либо по следующему фронту такта (бит `WAITCFG` = 0).

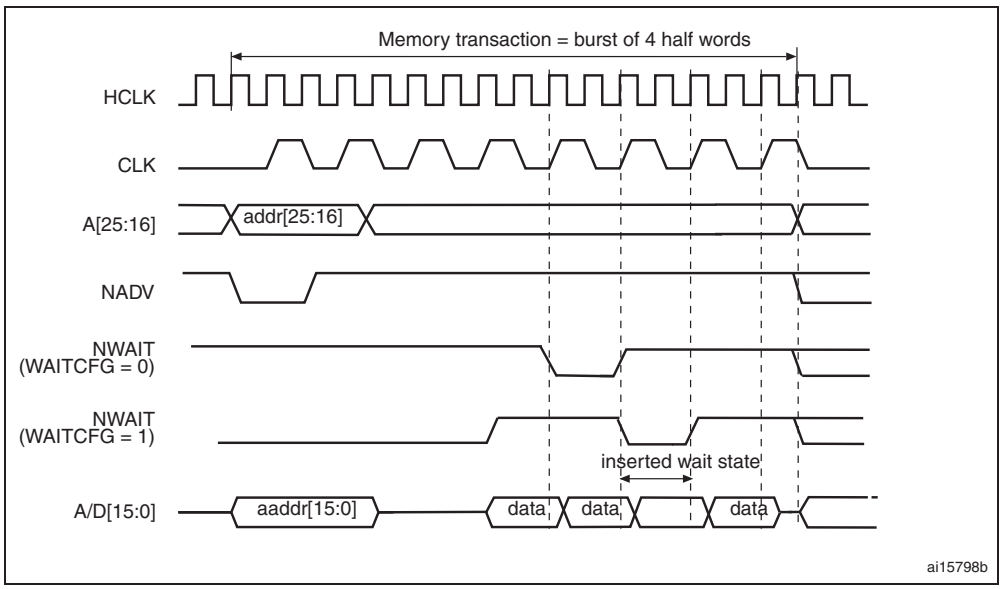
Во время вставки тактов ожидания сигналом `NWAIT` контроллер продолжает посылать такты памяти, сохраняет выбор чипа и разрешение вывода данных, но не рассматривает их достоверными.

Есть две конфигурации сигнала `NWAIT` для NOR Flash в пакетном режиме:

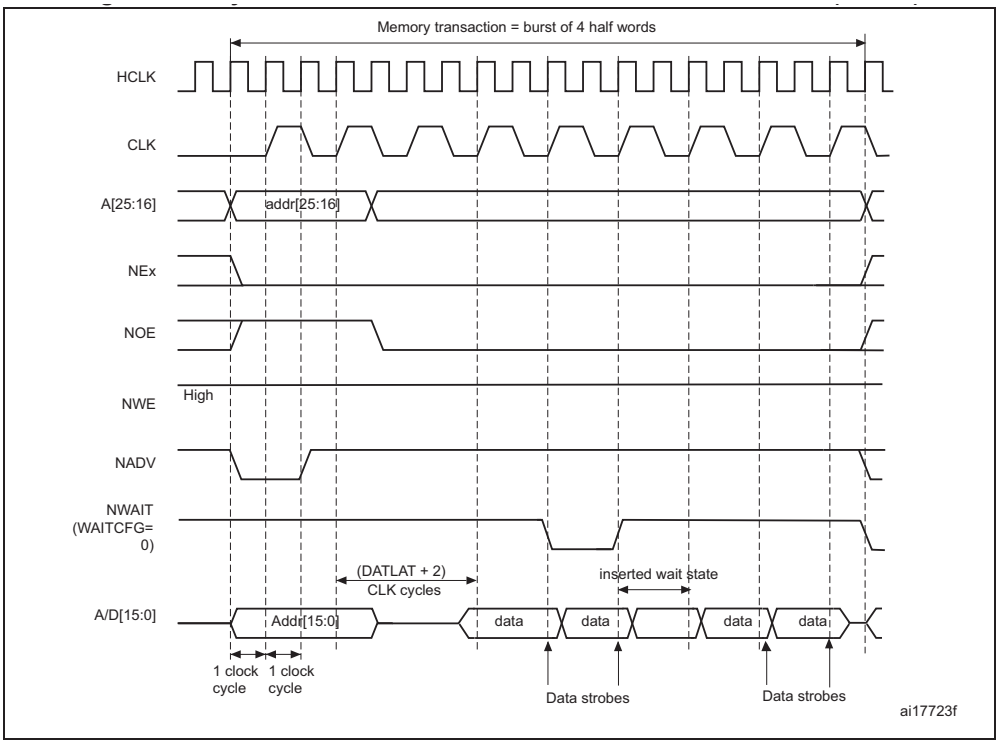
- Flash память выставляет сигнал `NWAIT` за один цикл до состояния ожидания (по умолчанию после сброса).
- Flash память выставляет сигнал `NWAIT` во время состояния ожидания.

Они поддерживаются FSMC раздельно для каждого выбора чипа битом `WAITCFG` в регистрах `FSMC_BCRx` (x = 0..3).

Конфигурации ожидания.



Синхронное мультиплексное чтение - NOR, PSRAM (CRAM).



1. Выходы байтовых линий BL не показаны; для NOR они высокие, для PSRAM (CRAM) низкие.
2. Полярность NWAIT ставится в 0.

Таблица 242. Биты FSMC_BCRx.

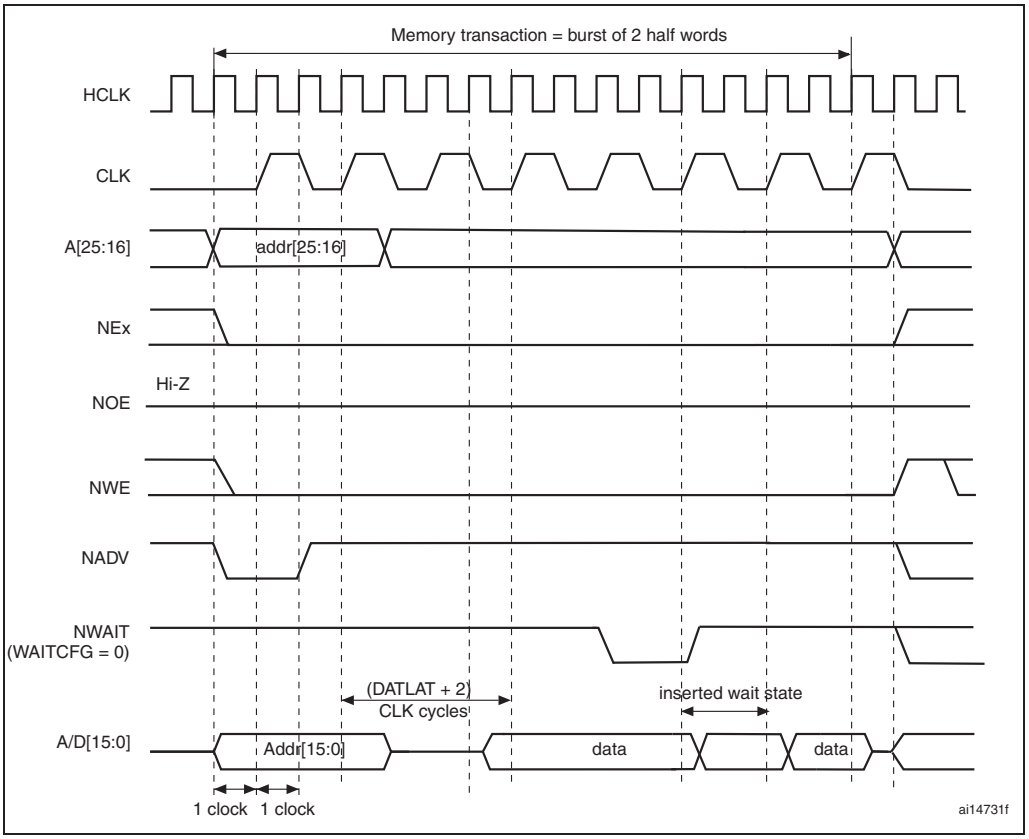
Номер	Имя	Устанавливаемое значение
31-21	Резерв	0x000
19	CBURSTRW	При синхронном чтении не влияет
18:16	CPSIZE	Как нужно (0x1 для CRAM 1.5)
15	ASYNCWAIT	0x0
14	EXTMOD	0x0
13	WAITEN	1, если память поддерживает, иначе 0.
12	WREN	При синхронном чтении не влияет
11	WAITCFG	Соответственно памяти.
10	WRAPMOD	0x0

Номер	Имя	Устанавливаемое значение
9	WAITPOL	Соответственно памяти.
8	BURSTEN	0x1
7	Резерв	0x1
6	FACCEN	Соответственно памяти (NOR Flash).
5-4	MWID	Как нужно
3-2	MTYP[0:1]	0x1 или 0x2
1	MUXEN	0x0
0	MBKEN	0x1

Таблица 243. Биты FSMC_BTRx.

Номер	Имя	Устанавливаемое значение
31-30	Резерв	0x0
29-28	ACCMOD	0x0
27-24	DATLAT	Задержка данных.
23-20	CLKDIV	0x0 для CLK = HCLK (не поддерживается) 0x1 для CLK = 2 × HCLK
19-16	BUSTURN	Не влияет.
15-8	DATAST	Всё равно.
7-4	ADDHLD	Всё равно.
3-0	ADDSET[3:0]	Всё равно.

Синхронная мультиплексная запись - PSRAM (CRAM).



1. Память должна выдавать NWAIT на такт раньше и WAITCFG должен быть равен 0.
2. Полярность NWAIT ставится в 0.
3. Байтовые линии (NBL) не показаны, при активном NEx удерживается низким.

Таблица 244. Биты FSMC_BCRx.

Номер	Имя	Устанавливаемое значение
31-20	Резерв	0x000
19	CBURSTRW	0x1
18:16	CPSIZE	Как нужно (0x1 для CRAM 1.5)
15	ASYNCWAIT	0x0
14	EXTMOD	0x0
13	WAITEN	1, если память поддерживает, иначе 0.
12	WREN	0x1
11	WAITCFG	0x0
10	WRAPMOD	0x0
9	WAITPOL	Соответственно памяти.
8	BURSTEN	На синхронную запись не влияет.
7	Резерв	0x1
6	FACCEN	Соответственно памяти.
5-4	MWID	Как нужно
3-2	MTYP[0:1]	0x1
1	MUXEN	Как нужно
0	MBKEN	0x1

Таблица 245. Биты FSMC_BTRx.

Номер	Имя	Устанавливаемое значение
31-30	Резерв	0x0
29-28	ACCMOD	0x0
27-24	DATLAT	Задержка данных.
23-20	CLKDIV	0x0 для CLK = HCLK (не поддерживается) 0x1 для CLK = 2 × HCLK
19-16	BUSTURN	Время между высоким и низким NEx (BUSTURN HCLK)
15-8	DATAST	Всё равно.
7-4	ADDHLD	Всё равно.
3-0	ADDSET[3:0]	Всё равно.

36.5.6. Регистры управления NOR/PSRAM

Они доступны словами.

Регистры выбора чипов SRAM/NOR-Flash 1..4 (FSMC_BCR1..4)

Смещение адреса: 0xA000 0000 + 8 * (x – 1), x = 1…4

По сбросу: 0x0000 30DB для Банка 1 и 0x0000 30D2 для Банков 2 - 4

Управляющая информация банков SRAM, PSRAM и NOR Flash памяти.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												CBURSTRW	CPSIZE[2:0]			ASCYCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	WRAPMOD	WAITPOL	BURSTEN	Reserved	FACCEN	MWID[1:0]		MTYP[1:0]		MUXEN	MBKEN
												rw				rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

— Биты 31:20 Резерв, не трогать.

— Бит 19 **CBURSTRW:** Разрешение синхронной пакетной записи Cellular RAM (PSRAM).

Синхронное пакетное чтение разрешается битом BURSTEN в регистре FSMC_BCRx.

0: Запись асинхронная

1: Запись синхронная.

— Биты 18:16 **CPSIZE[2:0]:** Размер страницы CRAM.

Используется Cellular RAM 1.5, не позволяющей пересечения границы страниц пакетными передачами. При установке этих битов FSMC автоматически разбивает пакетные передачи при достижении границы страниц.

000: Без автоматического разделения пакетов (по умолчанию после сброса)

001: 128 байт

010: 256 байт

011: 512 байт

100: 1024 байта

Остальные: резерв.

— Бит 15 **ASYNCWAIT**: Разрешение сигнала ожидания при асинхронных передачах.

0: В асинхронном протоколе сигнал NWAIT не учитывается (по умолчанию после сброса)

1: Сигнал NWAIT учитывается

— Бит 14 **EXTMOD:** Разрешение расширенного режима.

Разрешает учитывать биты времянки записи немультиплексного асинхронного доступа в регистре FSMC_BWTR, устанавливая различные времянки чтения и записи.

0: Значения FSMC_BWTR не учитываются (по умолчанию после сброса)

1: Значения FSMC_BWTR учитываются

NB: При выключенном расширенном режиме FSMC может работать в Режимах 1 или 2:

– Режим 1 используется по умолчанию для SRAM/PSRAM памяти (MTYP [0:1]=0x0 or 0x01)

– Режим 2 используется по умолчанию для NOR памяти (MTYP [0:1]= 0x10).

— Бит 13 **WAITEN**: Разрешение ожидания.

Разрешает вставку тактов ожидания по сигналу NWAIT при синхронном доступе к Flash памяти.

0: Сигнал NWAIT не учитывается, такты ожидания после периода задержки Flash не вставляются

1: Сигнал NWAIT учитывается, вставляются такты ожидания после периода задержки Flash (по умолчанию после сброса)

— Бит 12 **WREN**: Разрешение записи FSMC в банк памяти.

0: Запись запрещена, выдаётся ошибка AHB,

1: Запись разрешена.

— Бит 11 **WAITCFG**: Конфигурация времянки ожидания.

Сигнал NWAIT показывает достоверность данных из памяти, или необходимость вставки состояний ожидания при доступе к Flash памяти в синхронном режиме. Этот бит определяет установку NWAIT за один такт до состояния ожидания или во время его:

0: Сигнал NWAIT ставится за один такт до состояния ожидания (по умолчанию после сброса),

1: Сигнал NWAIT Ставится во время состояния ожидания (не используется в PRAM).

— Бит 10 **WRAPMOD**: Поддержка закольцованных пакетов.

Определяет разбиение закольцованного пакета AHB на два последовательных обращения. Действует только при пакетном доступе.

0: Прямой закольцованный пакет запрещён (по умолчанию после сброса),

1: Прямой закольцованный пакет разрешён.

NB: Бит не имеет значения если CPU и DMA не могут выдавать такие запросы.

— Бит 9 **WAITPOL**: Полярность сигнала ожидания.

Действует только при пакетном доступе:

0: NWAIT активен низким (по умолчанию после сброса),

1: NWAIT активен высоким.

— Бит 8 **BURSTEN**: Разрешение пакетов.

Разрешает синхронный доступ при чтении в пакетном режиме:

0: Пакеты запрещены (по умолчанию после сброса). Чтение в асинхронном режиме.

1: Пакеты разрешены.Чтение в синхронном режиме.

NB: При выключенном расширенном режиме FSMC может работать в Режимах 1 или 2:

– Режим 1 используется по умолчанию для SRAM/PSRAM памяти (MTYP [0:1]=0x0 or 0x01)

– Режим 2 используется по умолчанию для NOR памяти (MTYP [0:1]= 0x10).

— Бит 7 Резерв, не трогать.

— Бит 6 **FACCEN:** Разрешение доступа к NOR Flash

0: Доступ к NOR Flash запрещён

1: Доступ к NOR Flash разрешён (по умолчанию после сброса).

— Биты 5:4 **MWID[1:0]**: Ширина шины данных памяти.

00: 8 бит,

01: 16 бит (по умолчанию после сброса),

10: Резерв, не трогать.

11: Резерв, не трогать.

— Биты 3:2 **MTYP[1:0]:** Тип памяти.

Тип внешней памяти в банке:

00: SRAM (по умолчанию после сброса для Банков 2...4)

01: PSRAM (CRAM)

10: NOR Flash(по умолчанию после сброса для Банка 1)

11: Резерв, не трогать.

— Бит 1 **MUXEN**: Разрешение мультиплекса Адреса/Данных.

Стоящий бит разрешает мультиплексирование Адреса/Данных на шине данных для NOR и PSRAM памяти:

0: Адрес/Данные не мультиплексируются

1: Адрес/Данные мультиплексируются (по умолчанию после сброса).

— Бит 0 **MBKEN**: Разрешение банка памяти.

После сброса разрешён Банк 1, остальные запрещены.

Доступ к запрещённому банку вызывает Ошибку на шине AHB.

0: Соответствующий банк выключен

1: Соответствующий банк включен.

Регистры времянки выбора чипов 1..4 SRAM/NOR-Flash (FSMC_BTR1..4).

Смещение адреса: 0xA000 0000 + 0x04 + 8 * (x – 1), x = 1..4

По сбросу: 0x0FFF FFFF

Биты FSMC_BTRx программно добавляют задержку в конец передач чтения/записи. Она позволяет согласовать минимальное время между последовательными передачами (t_{EHEL} от высокого NEx до низкого FSMC_NEx) и максимального времени освобождения шины данных памятью после операции чтения (t_{EHQZ}).

Содержат управляющую информацию для всех банков SRAM, PSRAM и NOR Flash. Если бит EXTMOD в регистре FSMC_BCRx стоит, то появляются 2 регистра: для чтения (этот регистр) и для записи (FSMC_BWTRx).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		ACCMOD[1:0]		DATLAT[3:0]				CLKDIV[3:0]				BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:30 Резерв, не трогать.

— Биты 29:28 **ACCMOD[1:0]:** Режим доступа

Определяет режим асинхронного доступа. Работает только при стоящем бите EXTMOD в регистре FSMC_BCRx.

00: Режим A

01: Режим B

10: Режим C

11: Режим D

— Биты 27:24 **DATLAT[3:0]:** Задержка данных для синхронной NOR Flash памяти

Определяет число тактов (+2) синхронной NOR Flash памяти, подаваемых на память перед чтением/записью первых данных.

Выражается не в периодах HCLK, а в периодах FSMC_CLK. В случае PSRAM (CRAM) это поле должно быть нулевым. В асинхронных NOR Flash, SRAM или PSRAM это поле никого не волнует.

0000: Задержка в 2 такта CLK перед первым доступом пакета

1111: Задержка в 17 тактов CLK перед первым доступом пакета (по умолчанию после сброса)

— Биты 23:20 **CLKDIV[3:0]:** Делитель частоты сигнала FSMC_CLK в тактах HCLK

0000: Резерв

0001: Период FSMC_CLK = 2 × HCLK периодов

0010: Период FSMC_CLK = 3 × HCLK периодов

1111: Период FSMC_CLK = 16 × HCLK периодов (по умолчанию после сброса)

В асинхронных NOR Flash, SRAM или PSRAM это поле никого вообще не волнует.

— Биты 19:16 **BUSTURN[3:0]:** Длительность фазы реверса шины

Эти биты программно добавляют задержку в конец передач чтение/запись и запись/чтение. Программируемая задержка вставляется между асинхронными чтением (мультиплекс или режим D) или записью и любым другим асинхронно/синхронным доступом к статическому банку (при чтении банк может быть тем же или другим; при записи банк может быть различным исключая мультиплекс и режим D).

В некоторых случаях, задержка реверса шины фиксирована при любом значении BUSTURN:

- Задержка реверса шины не вставляется между двумя последовательными записями в один банк статической памяти, исключая мультиплекс и режим D.
- Задержка реверса шины в 1 такт FSMC вставляется между:
  - Двумя последовательными асинхронными чтениями из одного банка статической памяти, исключая мультиплекс и режим D.
  - Асинхронным чтением и асинхронной или синхронной записью в любой статический или динамический банк, исключая мультиплекс и режим D.
  - Асинхронным чтением (режимы 1, 2, A, B or C) и чтением из другого статического банка.
- Задержка реверса шины в 2 такта FSMC вставляется между:
  - Двумя последовательными синхронными записями (пакетными или одиночными) в один банк
  - Синхронной записью (пакетной или одиночной) и асинхронным обменом с статическим банком памяти (банк может быть тем же или другим в случае чтения).
  - Двумя последовательными синхронными чтениями (пакетными или одиночными) с последующим синхронным/асинхронным обменом с другим банком статической памяти.
- Задержка реверса шины в 3 такта FSMC вставляется между:
  - Двумя последовательными синхронными записями (пакетными или одиночными) в разные статические банки.
  - Синхронной записью (пакетной или одиночной) и синхронным чтением из одного или разных банков.

0000: Длительность фазы BUSTURN = 1 добавленный такт HCLK

…

1111: Длительность фазы BUSTURN = 16 × HCLK тактов (по умолчанию после сброса)

— Биты 15:8 **DATAST[7:0]:** Длительность фазы данных асинхронного доступа.

0000 0000: Резерв.

0000 0001: Длительность фазы DATAST = 2 × HCLK тактов

0000 0010: Длительность фазы DATAST = 3 × HCLK тактов

…

1111 1111: Длительность фазы DATAST = 256 × HCLK тактов (по умолчанию после сброса)

Пример: Режим 1, чтение, DATAST=1: Фаза данных = DATAST+3 = 4 HCLK тактов.

NB: При синхронном доступе никого не волнует.

— Биты 7:4 **ADDHLD[3:0]:** Длительность фазы удержания адреса.

Используется при мультиплексе и режиме D:

0000: Резерв

0001: Длительность фазы ADDHLD = 2 × HCLK тактов

0010: Длительность фазы ADDHLD = 3 × HCLK тактов

…

1111: Длительность фазы ADDHLD = 16 × HCLK тактов (по умолчанию после сброса)

Пример: Режим D, чтение, ADDHLD=1: Фаза удержания адреса = ADDHLD + 1 = 2 HCLK такта.

NB: При синхронном доступе не используется, фаза всегда равна 1 такту памяти.

— Биты 3:0 **ADDSET[3:0]:** Длительность фазы установки адреса.

Используется в SRAM, ROM и асинхронных NOR Flash и PSRAM:

0000: Длительность фазы ADDSET = 1 × HCLK тактов

…

1111: Длительность фазы ADDSET = × HCLK тактов (по умолчанию после сброса)

Пример: Режим 2, чтение, ADDSET=1: Фаза установки адреса = ADDSET + 1 = 2 HCLK тактов.

NB: При синхронном доступе к NOR Flash и PSRAM никому не интересно.

NB: PSRAM (CRAM) имеет переменную задержку из-за внутренней регенерации. Так что эта память выдаёт сигнал NWAIT на всё нужное время.

С памятью PSRAM (CRAM) поле DATLAT должно быть обнулено, так что FSMC вскоре выходит из задержки и смотрит на сигнал NWAIT от памяти memory, затем при готовности читает или пишет в память.

Этот метод можно использовать и с позднейшими поколениями Flash памяти, выдающими сигнал NWAIT, в отличие от старой Flash памяти.

Регистры времянки записи 1..4 SRAM/NOR-Flash (FSMC_BWTR1..4).

Смещение адреса: 0xA000 0000 + 0x104 + 8 * (x – 1), x = 1..4

По сбросу: 0x0FFF FFFF

Содержит управляющую информацию всех банков памяти SRAM, PSRAM и NOR Flash. Регистр работает только при асинхронной записи (стоит бит `EXTMOD` в регистре `FSMC_BCRx`).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.		ACCMOD[2:0]		Reserved								BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
		rw	rw									rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:30 Резерв, не трогать.

— Биты 29:28 **ACCMOD[1:0]:** Режим доступа

Определяет режим асинхронного доступа. Работает только при стоящем бите EXTMOD в регистре FSMC_BCRx.

00: Режим A

01: Режим B

10: Режим C

11: Режим D

— Биты 27:20 Резерв, не трогать.

— Биты 19:16 **BUSTURN[3:0]:** Длительность фазы реверса шины

Программируемая задержка вставляется между асинхронной записью и любым другим асинхронно/синхронным доступом к статическому банку (при чтении банк может быть тем же или другим; при записи банк может быть различным, исключая мультиплекс и режим D).

В некоторых случаях, задержка реверса шины фиксирована при любом значении BUSTURN:

– Задержка реверса шины не вставляется между двумя последовательными записями в один банк статической памяти, исключая мультиплекс и режим D.

– Задержка реверса шины в 2 такта FSMC вставляется между:

  – Двумя последовательными синхронными записями (пакетными или одиночными) в один банк

  – Синхронной записью (пакетной или одиночной) и асинхронным обменом с статическим банком памяти (банк может быть тем же или другим в случае чтения).

– Задержка реверса шины в 3 такта FSMC вставляется между:

  – Двумя последовательными синхронными записями (пакетными или одиночными) в разные статические банки.

  – Синхронной записью (пакетной или одиночной) и синхронным чтением из одного или разных банков.

0000: Длительность фазы BUSTURN = 1 добавленный такт HCLK

…

1111: Длительность фазы BUSTURN = 16 × HCLK тактов (по умолчанию после сброса)

— Биты 15:8 **DATAST[7:0]:** Длительность фазы данных асинхронного доступа.

0000 0000: Резерв.
0000 0001: Длительность фазы DATAST = 2 × HCLK тактов
0000 0010: Длительность фазы DATAST = 3 × HCLK тактов
…
1111 1111: Длительность фазы DATAST = 256 × HCLK тактов (по умолчанию после сброса)
Пример: Режим 1, чтение, DATAST=1: Фаза данных = DATAST+3 = 4 HCLK тактов.
NB: При синхронном доступе никого не волнует.

— Биты 7:4 **ADDHLD[3:0]:** Длительность фазы удержания адреса.
Используется при мультиплексе и режиме D:
0000: Резерв
0001: Длительность фазы ADDHLD = 2 × HCLK тактов
0010: Длительность фазы ADDHLD = 3 × HCLK тактов
…
1111: Длительность фазы ADDHLD = 16 × HCLK тактов (по умолчанию после сброса)
Пример: Режим D, чтение, ADDHLD=1: Фаза удержания адреса = ADDHLD + 1 = 2 HCLK такта.
NB: При синхронном доступе не используется, фаза всегда равна 1 такту памяти.

— Биты 3:0 **ADDSET[3:0]:** Длительность фазы установки адреса.
Используется в SRAM, ROM и асинхронных NOR Flash и PSRAM:
0000: Длительность фазы ADDSET = 1 × HCLK тактов
…
1111: Длительность фазы ADDSET = × HCLK тактов (по умолчанию после сброса)
Пример: Режим 2, чтение, ADDSET=1: Фаза установки адреса = ADDSET + 1 = 2 HCLK тактов.
NB: При синхронном доступе к NOR Flash и PSRAM никому не интересно.

36.6. Контроллер NAND Flash/PC Card

FSMC выдаёт нужную времянку сигналов для следующих типов памяти:

- NAND Flash
  - 8-бит
  - 16-бит
- 16-bit PC Card совместимые устройства

Контроллер NAND/PC Card управляет тремя внешними банками. Банк 2 и банк 3 поддерживают NAND Flash устройства. Банк 4 поддерживает устройства PC Card.

Каждый банк конфигурируется своими регистрами. Программируемые параметры памяти включают времянку доступа и конфигурацию контроля ошибок (ECC).

Таблица 246. Параметры доступа NAND Flash/PC Card.

Параметр памяти	Функция	Режим доступа	Единицы	Min.	Max.
Время установки	Число тактов HCLK установки адреса перед подачей команды	Ч/З	HCLK	1	255
Ожидание	Минимальная длительность подачи команды	Ч/З	HCLK	2	256
Удержание	Число тактов удержания адреса (и данных при записи) после снятия команды	Ч/З	HCLK	1	254
Hi-Z шины данных	Число тактов удержания шиной данных high-Z состояния после начала записи	З	HCLK	0	255

36.6.1. Сигналы интерфейса внешней памяти

Внимание: При использовании PC Card или CompactFlash в режиме I/O, входная ножка NIOS16 должна быть заземлена, а не подключаться к карте, иначе FSMC забастует, откажется работать и пр.

Теоретически ограничений ёмкости нет, так как FSMC может использовать произвольное число тактов адреса.

NB: Префикс “N” указывает на активный низкий уровень сигнала.

Таблица 247. 8-бит NAND Flash

Сигнал FSMC	Вв./Выв.	Функция
A[17]	Ы	Разрешение защёлки адреса NAND Flash (ALE)
A[16]	Ы	Разрешение защёлки команды NAND Flash (CLE)
D[7:0]	В/Ы	8-бит мультиплексная, двунаправленная шина адреса/данных
NCE[x]	Ы	Выбор чипа, x = 2, 3
NOE(= NRE)	Ы	Разрешение выхода (в памяти: разрешение чтения, NRE)
NWE	Ы	Разрешение записи
NWAIT/INT[3:2]	В	Сигнал готов/занят NAND Flash для FSMC

Таблица 248. 16-бит NAND Flash

Сигнал FSMC	Вв./Выв.	Функция
A[17]	Ы	Разрешение защёлки адреса NAND Flash (ALE)
A[16]	Ы	Разрешение защёлки команды NAND Flash (CLE)
D[7:0]	В/Ы	16-бит мультиплексная, двунаправленная шина адреса/данных
NCE[x]	Ы	Выбор чипа, x = 2, 3
NOE(= NRE)	Ы	Разрешение выхода (в памяти: разрешение чтения, NRE)
NWE	Ы	Разрешение записи
NWAIT/INT[3:2]	В	Сигнал готов/занят NAND Flash для FSMC

Таблица 249. 16-бит PC Card

Сигнал FSMC	Вв./Выв.	Функция
A[10:0]	O	Address bus
NIOS16	I	Data transfer in I/O space. It must be shorted to GND (16-bit transfer only)
NIORD	O	Output enable for I/O space
NIOWR	O	Write enable for I/O space
NREG	O	Register signal indicating if access is in Common or Attribute space
D[15:0]	I/O	Bidirectional databus
NCE4_1	O	Chip select 1
NCE4_2	O	Chip select 2 (indicates if access is 16-bit or 8-bit)
NOE	O	Output enable in Common and in Attribute space
NWE	O	Write enable in Common and in Attribute space
NWAIT	I	PC Card wait input signal to the FSMC (memory signal name IORDY)
INTR	I	PC Card interrupt to the FSMC (only for PC Cards that can generate an interrupt)
CD	I	PC Card presence detection. Active high. If an access is performed to the PC Card banks while CD is low, an AHB error is generated.

36.6.2. Поддерживаемая память и передачи NAND Flash / PC Card

В таблице ниже неразрешённые передачи указаны красным цветом.

Уст-во	Режим	Ч/З	Размер данных AHB	Размер данных памяти	Можно	Комментарии
NAND 8-бит	Асинхр.	Ч	8	8	Д	
	Асинхр.	З	8	8	Д	
	Асинхр.	Ч	16	8	Д	За два обращения FSMC
	Асинхр.	З	16	8	Д	За два обращения FSMC
	Асинхр.	Ч	32	8	Д	За 4 обращения FSMC
	Асинхр.	З	32	8	Д	За 4 обращения FSMC
NAND 16-бит	Асинхр.	Ч	8	16	Д	
	Асинхр.	З	8	16	Н	
	Асинхр.	Ч	16	16	Д	
	Асинхр.	З	16	16	Д	
	Асинхр.	Ч	32	16	Д	За два обращения FSMC
	Асинхр.	З	32	16	Д	За два обращения FSMC

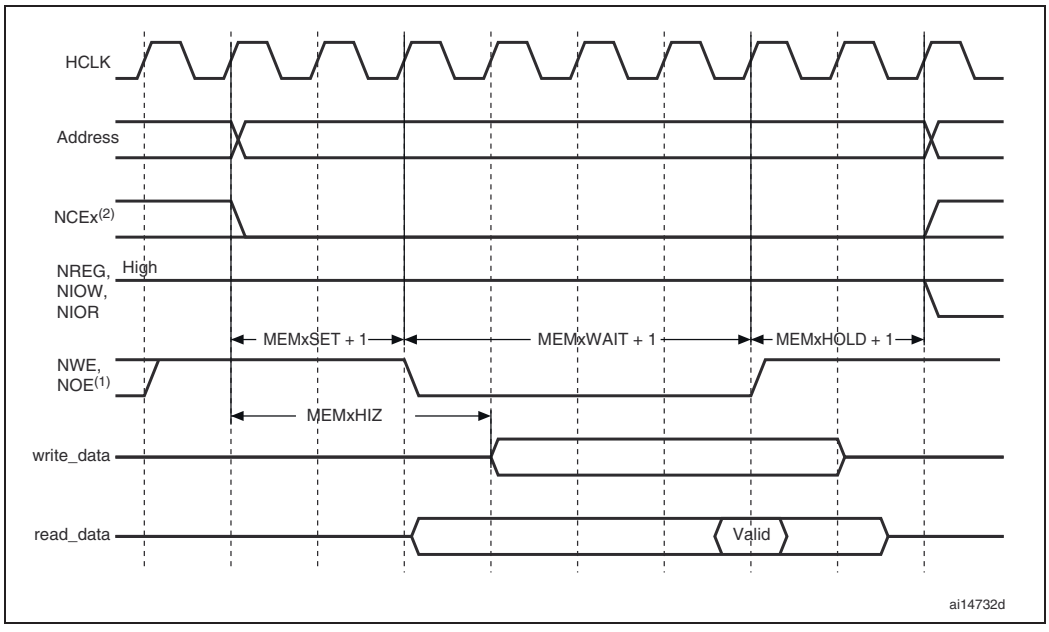
36.6.3. Временные диаграммы NAND Flash / PC Card

Банки памяти PC Card/CompactFlash и NAND Flash управляются регистрами:

- Управления: `FSMC_PCRx`
- Состояния прерываний: `FSMC_SRx`
- Коррекции ошибок (ECC): `FSMC_ECCRx`
- Времянки Общей памяти: `FSMC_PMEMx`
- Времянки памяти Атрибутов: `FSMC_PATTx`
- Времянки памяти Вв./Выв.: `FSMC_PIOx`

Регистры времянки содержат три параметра, определяющих число тактов HCLK для трёх фаз доступа к PC Card/CompactFlash NAND Flash, плюс ещё один, задающий время начала управления шиной данных при записи. Ниже показаны параметры доступа к Общей памяти, так как для памяти Атрибутов и Вв./Выв. (только для PC Card) они такие же.

Времянки доступа к Общей памяти NAND/PC Card.



1. NOE остаётся высоким (неактивным) при записи. NWE остаётся высоким (неактивным) при чтении.
2. NCEx ставится низким при запросе доступа к NAND и держится до обращения к другому банку.

36.6.4. Операции NAND Flash

Сигналы разрешения защёлок команды (CLE) и адреса (ALE) у NAND управляются сигналами адреса FSMC. То есть, CPU пишет команды или адреса в определённые области NAND Flash памяти.

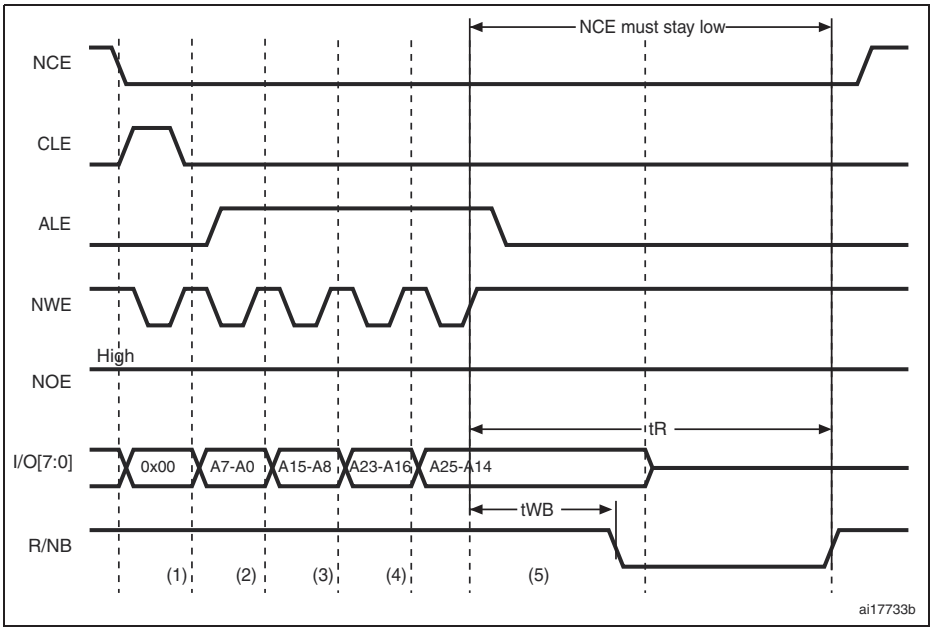
Вот типичное чтение из NAND Flash:

1. Регистрами FSMC_PCRx и FSMC_PMEMx (иногда и FSMC_PATTx) выбрать банк NAND Flash в соответствии с его характеристиками (битами PWID ширину шины данных, и как нужно PTYP = 1, PWAITEN = 0 или 1).
2. CPU пишет один байт команды в общую память (например 0x00 для Samsung NAND Flash). Во время строба записи (низкий импульс на NWE) сигнал на входе CLE активен, и байт рассматривается как команда. Теперь при последующих чтениях страницы команду подавать не нужно.
3. Теперь CPU может послать адрес начала чтения (STARTAD) записью байтов (например, STARTAD[7:0], STARTAD[15:8], STARTAD[23:16] и, наконец, STARTAD[25:24] для 64 Mb x 8 бит NAND Flash) в общую память или пространство атрибутов. Во время строба записи (низкий импульс на NWE) сигнал на входе ALE активен, и байты рассматриваются как начальный адрес чтения. Использование памяти атрибутов позволяет FSMC включать разные времянки вроде предожидания в некоторых NAND Flash.
4. Контроллер ждёт готовности NAND Flash (высокий сигнал R/NB) перед новым обращением (в тот же или другой банк). При этом контроллер держит сигнал NCE активным (низким).
5. Теперь CPU может байт за байтом читать из общей памяти страницу NAND Flash (поле данных + резервное поле).
6. Следующую страницу NAND Flash можно читать без записи команд и адреса тремя способами:
   - просто выполняя операцию шага 5
   - писать новый адрес, начиная с шага 3
   - послать новую команду, начиная с шага 2.

36.6.5. Предожидание NAND Flash

Некоторые устройства NAND Flash требуют чтобы после записи последней части адреса контроллер ждал падения сигнала R/NB.

Доступ к не ‘CE не волнует’ NAND-Flash.



1. CPU записал байт 0x00 по адресу 0x7001 0000.
2. CPU записал байт A7-A0 по адресу 0x7002 0000.
3. CPU записал байт A15-A8 по адресу 0x7002 0000.
4. CPU записал байт A23-A16 по адресу 0x7002 0000.
5. CPU записал байт A25-A24 по адресу 0x7802 0000: FSMC выполняет запись используя определение FSMC_PATT2, где ATTHOLD ≥ 7 (обеспечивая (7+1) × HCLK = 112 ns > t_{WB} max). так гарантируется что NCE остаётся низким вплоть до падения и подъёма R/NB (требуется для NAND Flash, безразличным к NCE).

При надобности это достигается установкой MEMHOLD для соответствия t_{WB}. Но у чтения NAND Flash задержка удержания равна (MEMHOLD + 2) x HCLK тактов, а запись имеет задержку удержания (MEMHOLD) x HCLK тактов.

Чтобы обойти это можно использовать память атрибутов установив ATTHOLD для получения времянки t_{WB}, и оставив MEMHOLD минимальной. Тогда при всех обменах с NAND Flash надо использовать общую память, но последний байт адреса нужно писать а память атрибутов.

36.6.6. Вычисление кода коррекции ошибки (ECC) NAND Flash

Контроллер FSMC PC-Card имеет два блока аппаратного вычисления коррекции ошибок, по одному на блоки памяти 2 и 3. С блоком 4 аппаратное вычисление ECC невозможно. Оба регистра идентичны.

Использованный алгоритм позволяет исправлять ошибку в 1 бите и обнаруживать ошибки 2 битов в 256, 512, 1 024, 2 048, 4 096 и 8 192 передачах. Он основан на кодах Хэмминга с вычислением чётности строк и колонок.

Модули ECC отслеживают шину данных и сигналы NCE и NWE активного банка памяти NAND Flash.

Они работают так:

- При доступе к банкам 2 и 3 NAND Flash перехваченные данные на шине D[15:0] запоминаются для последующих вычислений.
- При доступе по любым другим адресам логика ECC отдыхает, так что команды и сами адреса в вычислениях не участвуют.

После передачи желаемого числа байт из регистров FSMC_ECCR2/3 читается результат вычисления и они сбрасываются обнулением бита ECCEN. Для вычисления нового блока бит ECCEN в регистре FSMC_PCR2/3 нужно поставить.

Вычисляем ECC:

1. Ставим бит ECCEN в регистре FSMC_PCR2/3.
2. Пишем данные в страницу памяти NAND Flash. Блок ECC в это время вычисляет ECC.
3. Сохраняем значение ECC из FSMC_ECCR2/3 в переменной.
4. Снимаем и снова ставим бит ECCEN в регистре FSMC_PCR2/3 для обратного чтения данных. Блок ECC в это время вычисляет ECC.
5. Читаем новое значение ECC из регистра FSMC_ECCR2/3.
6. Если оба значения ECC равны, то можно спокойно вздохнуть, иначе программа исправления ошибок скажет вам, может ли она это сделать.

36.6.7. Операции PC Card/CompactFlash

Адресные пространства и доступ к памяти

FSMC поддерживает Compact Flash и PC Cards в режимах Памяти и I/O (но не True IDE).

Compact Flash и PC Card имеют 3 пространства памяти:

- Общее
- Атрибутов
- Память I/O

Ножки nCE2 и nCE1 (FSMC_NCE4_2 и FSMC_NCE4_1) выбирают карту и словный или байтовый доступ: nCE2 выбирает нечётный байт на D15-8 и nCE1 выбирает чётный байт на D7-0 при A0=0 или нечётный байт на D7-0 при A0=1. Полное слово на D15-0 выбирается при обоих низких nCE2 и nCE1.

Пространство памяти выбирается низким nOE при чтении или низким nWE при записи, в сочетании с низкими nCE2/nCE1 и nREG.

- Если при доступе к памяти nREG=1, то используется общая память
- Если при доступе к памяти nREG=0, то используется память атрибутов

Пространство I/O выбирается низким nIORD при чтении или низким nIOWR при записи [вместо nOE/nWE для пространства памяти], в сочетании с nCE2/nCE1. При этом nREG должен быть низким.

Для 16-бит PC Card есть три вида доступа:

- К общей памяти: 8-бит по чётным адресам или 16-бит AHB.

  8-бит доступ по нечётным адресам не поддерживается и не ведёт к выставлению низкого nCE2.

  32-бит запросы AHB транслируются в два 16-бит обращения к памяти.
- К памяти атрибутов с конфигурацией PC Card по 8-бит запросу AHB по чётным адресам.

  16-бит доступ AHB преобразуется в одно 8-бит обращение к памяти: nCE1 будет низким, а nCE2 высоким и достоверным будет только байт на D7–0. 32-бит доступ AHB преобразуется в два 8-bit обращения к памяти по чётным адресам: nCE1 будет низким, а nCE2 будет высоким и достоверным будет только чётный байт.
- Доступ к пространству I/O выполняется 8- или 16-бит запросами AHB.

Таблица 251. Сигналы и типы доступа 16-бит PC Card.

nCE2	nCE1	nREG	nOE/nWE	nIORD /nIOWR	A10	A9	A7-1	A0	Space	Access Type	Allowed/not Allowed
1	0	1	0	1	X	X	X-X	X	Common Memory Space	Read/Write byte on D7-D0	YES
0	1	1	0	1	X	X	X-X	X		Read/Write byte on D15-D8	Not supported
0	0	1	0	1	X	X	X-X	0		Read/Write word on D15-D0	YES
X	0	0	0	1	0	1	X-X	0	Attribute Space	Read or Write Configuration Registers	YES
X	0	0	0	1	0	0	X-X	0		Read or Write CIS (Card Information Structure)	YES
1	0	0	0	1	X	X	X-X	1	Attribute Space	Invalid Read or Write (odd address)	YES
0	1	0	0	1	X	X	X-X	x		Invalid Read or Write (odd address)	YES
1	0	0	1	0	X	X	X-X	0	I/O space	Read Even Byte on D7-0	YES
1	0	0	1	0	X	X	X-X	1		Read Odd Byte on D7-0	YES
1	0	0	1	0	X	X	X-X	0		Write Even Byte on D7-0	YES
1	0	0	1	0	X	X	X-X	1		Write Odd Byte on D7-0	YES
0	0	0	1	0	X	X	X-X	0		Read Word on D15-0	YES
0	0	0	1	0	X	X	X-X	0		Write word on D15-0	YES
0	1	0	1	0	X	X	X-X	X		Read Odd Byte on D15-8	Not supported
0	1	0	1	0	X	X	X-X	X		Write Odd Byte on D15-8	Not supported

Банк 4 FSMC позволяет доступ к этим 3 пространствам. См. *Секцию 21.4.2* и *Таблицу 102*.

Ожидание

Если ожидание разрешено битом PWAITEN в регистре FSMC_PCRx, то CompactFlash и PC Card могут запросить у FSMC расширение фазы доступа, указанное битами у MEMWAITx/ATTWAITx/IOWAITx, выставляя сигнал nWAIT после активации nOE/nWE или nIORD/nIOWR. Для правильного определения nWAIT биты MEMWAITx/ATTWAITx/IOWAITx нужно определять так:

```
xxWAITx >= 4 + max_wait_assertion_time/HCLK
```

где max_wait_assertion_time максимальное время падения nWAIT после падения nOE/nWE или nIORD/nIOWR.

После снятия nWAIT FSMC Расширяет фазу ожидания на 4 такта HCLK.

36.6.8. Управляющие регистры NAND Flash/PC Card

Они доступны словами (32 бита).

Регистры управления 2..4 NAND Flash/PC Card (FSMC_PCR2..4).

Смещение адреса: 0xA0000000 + 0x40 + 0x20 * (x – 1), x = 2..4

По сбросу: 0x0000 0018

31–20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	ECCPS[2:0]			TAR[2:0]				TCLR[2:0]				Res.		ECCEN	PWID[1:0]		PTYP	PBKEN	PWAITEN	Reserved
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	

— Биты 31:20 Резерв, не трогать.

— Биты 19:17 **ECCPS[2:0]:** Размер страницы расширенного ECC.

000: 256 байт
001: 512 байт
010: 1024 байта
011: 2048 байт
100: 4096 байт
101: 8192 байта

— Биты 16:13 **TAR[2:0]:** Задержка от ALE до RE в тактах AHB (HCLK).

Время: t_ar = (TAR + SET + 4) × THCLK где THCLK это период такта HCLK

0000: 1 так HCLK (по умолчанию)
1111: 16 тактов HCLK

NB: SET это MEMSET или ATTSET в зависимости от пространства адресов.

— Биты 12:9 **TCLR[2:0]:** Задержка от CLE до RE в тактах AHB (HCLK).

Время: t_clr = (TCLR + SET + 4) × THCLK где THCLK это период такта HCLK

0000: 1 так HCLK (по умолчанию)
1111: 16 тактов HCLK

NB: SET это MEMSET или ATTSET в зависимости от пространства адресов.

— Биты 8:7 Резерв, не трогать.

— Бит 6 **ECCEN**: Разрешение работы ECC

0: Логика ECC отключена и сброшена (по умолчанию после сброса),
1: Логика ECC включена.

— Биты 5:4 **PWID[1:0]:** Ширина шины данных внешней памяти.

00: 8 бит
01: 16 бит (по умолчанию после сброса). Для PC Card обязательно.
10: Резерв, не трогать.
11: Резерв, не трогать.

— Бит 3 **PTYP**: Тип памяти банка.

0: PC Card, CompactFlash, CF+ или PCMCIA
1: NAND Flash (по умолчанию после сброса)

— Бит 2 **PBKEN**: Включение банка памяти PC Card/NAND Flash.

Обращение к отключённому банку вызывает ERROR на шине AHB.

0: Выключен (по умолчанию после сброса)
1: Включён

— Бит 1 **PWAITEN**: Разрешение ожидания.

0: Нельзя
1: Можно

NB: Для PC Card биты MEMWAITx/ATTWAITx/IOWAITx должны ставиться так:

xxWAITx ≥ 4 + max_wait_assertion_time/HCLK

где max_wait_assertion_time это максимальное время между падением nWAIT и падением nOE/nWE или nIORD/nIOWR.

— Бит 0 Резерв, не трогать.

Регистр 2..4 состояния и прерываний FIFO (FSMC_SR2..4)

Смещение адреса: 0xA000 0000 + 0x44 + 0x20 * (x-1), x = 2..4

По сбросу: 0x0000 0040

FSMC использует FIFO при записи в память для хранения до 16 слов данных от AHB. При этом AHB быстро освобождается для передач другой периферии пока FSMC сливает FIFO в память. При вычислении ECC надо использовать один бит состояния FIFO.

ECC вычисляется при записи в память, так что достоверное значение получается при пустом FIFO.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																									FEMPT	IFEN	ILEN	IREN	IFS	ILS	IRS
																									r	rw	rw	rw	rw	rw	rw

— Биты 31:20 Резерв, не трогать.

— Бит 6 **FEMPT:** FIFO пуст.

Только чтение.

0: Что-то есть

1: FIFO пуст

— Бит 5 **IFEN:** Разрешение определения заднего фронта прерывания

0: Нельзя

1: Можно

— Бит 4 **ILEN:** Разрешение определения высокого уровня прерывания

0: Нельзя

1: Можно

— Бит 3 **IREN:** Разрешение определения переднего фронта прерывания

0: Нельзя

1: Можно

— Бит 2 **IFS:** Флаг заднего фронта прерывания

Ставится аппаратно, снимается программно.

0: Не было

1: Случилось

NB: Ставится программно (ЛАЖА КАКАЯ-ТО, но там так и написано). Может снимается записью 1?

— Бит 1 **ILS:** Флаг высокого уровня прерывания

Ставится аппаратно, снимается программно.

0: Не было

1: Случилось

— Бит 0 **IRS:** Флаг переднего фронта прерывания

Ставится аппаратно, снимается программно.

0: Не было

1: Случилось

NB: Ставится программно (ОПЯТЬ ЛАЖА, но там ИМЕННО ТАК). Может снимается записью 1?

Регистр 2..4 времянки общей памяти (FSMC_PMEM2..4)

Смещение адреса: 0xA000 0000 + 0x48 + 0x20 * (x – 1), x = 2..4

По сбросу: 0xFCFC FCFC

Каждый регистр чтения/записи `FSMC_PMEMx` (x = 2..4) содержит определение времянки банка X PC Card или NAND Flash, используемой при доступе к общей памяти 16-бит PC Card/CompactFlash, или при записи команд и адреса NAND Flash и при чтении/записи данных.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MEMHIZ[7:0]								MEMHOLD[7:0]								MEMWAIT[7:0]								MEMSET[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:24 **MEMHIZx[7:0]:** Время состояния HiZ шины данных общей памяти X.

Число тактов HCLK (+1 только для NAND) удержания состояния HiZ на шине данных после старта записи в общее пространство (сокет X) PC Card/NAND Flash. Работает только при записи:

0000 0000: 1 такт HCLK

1111 1110: 255 тактов HCLK

1111 1111: Резерв

— Биты 23:16 **MEMHOLDx[7:0]**: Время удержания общей памяти X.

При чтении общей памяти NAND Flash определяют число тактов (HCLK+2) удержания адреса после снятия команды (NWE, NOE).

При записи общей памяти NAND Flash определяют число тактов (HCLK) удержания данных после снятия команды (NWE, NOE).

0000 0000: Резерв.

0000 0001: 1 такт HCLK при записи, 3 такта HCLK при чтении.

1111 1110: 254 такта HCLK при записи, 256 тактов HCLK при чтении.

1111 1111: Резерв.

— Биты 15:9 **MEMWAITx[7:0]:** Время ожидания общей памяти X.

При чтении и записи общей памяти PC Card/NAND Flash определяют минимальное число тактов HCLK (+1) выдачи команд (NWE, NOE) для сокета X. Длительность команд может расширяться сигналом ожидания (NWAIT) после заданного значения HCLK:

0000 0000: Резерв

0000 0001: 2 такта HCLK (+ такт ожидания, вставленный снятием NWAIT)

1111 1110: 255 тактов HCLK (+ такт ожидания, вставленный снятием NWAIT)

1111 1111: Резерв.

— Биты 8:0 **MEMSETx[7:0]:** Время установки общей памяти X.

При чтении и записи общей памяти PC Card/NAND Flash определяют число тактов HCLK (+1 для PC Card, +2 для NAND) установки адреса перед подачей команд (NWE, NOE) для сокета X:

0000 0000: 1 такт HCLK

1111 1110: 255 тактов HCLK

1111 1111: Резерв

Регистр 2..4 времянки памяти атрибутов (FSMC_PATT2..4)

Смещение адреса: 0xA000 0000 + 0x4C + 0x20 * (x – 1), x = 2..4

По сбросу: 0xFCFC FCFC

Каждый регистр чтения/записи `FSMC_PATTx` (x = 2..4) содержит определение времянки банка X PC Card/CompactFlash или NAND Flash, используемой при 8-бит доступе к памяти атрибутов PC Card/CompactFlash, или при записи последней части адреса NAND Flash, если при этом времянка должна отличаться от предыдущих стадий.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ATTHIZ[7:0]								ATTHOLD[7:0]								ATTWAIT[7:0]								ATTSET[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:24 **ATTHIZx[7:0]:** Время состояния HiZ шины данных памяти атрибутов X.

Число тактов HCLK (+1 только для NAND) удержания состояния HiZ на шине данных после старта записи в пространство атрибутов (сокет X) PC Card/NAND Flash. Работает только при записи:

0000 0000: 0 тактов HCLK

1111 1110: 255 тактов HCLK

1111 1111: Резерв

— Биты 23:16 **ATTHOLDx[7:0]**: Время удержания памяти атрибутов X.

При чтении памяти атрибутов PC Card/NAND Flash определяют число тактов (HCLK+2) удержания адреса после снятия команды (NWE, NOE).

При записи памяти атрибутов PC Card/NAND Flash определяют число тактов (HCLK) удержания данных после снятия команды (NWE, NOE).

0000 0000: Резерв.

0000 0001: 1 такт HCLK при записи, 3 такта HCLK при чтении.

1111 1110: 254 такта HCLK при записи, 256 тактов HCLK при чтении.

1111 1111: Резерв.

— Биты 15:9 **ATTWAITx[7:0]:** Время ожидания памяти атрибутов X.

При чтении и записи памяти атрибутов PC Card/NAND Flash определяют минимальное число тактов HCLK (+1) выдачи команд (NWE, NOE) для сокета X. Длительность команд может расширяться сигналом ожидания (NWAIT) после заданного значения HCLK:

0000 0000: Резерв

0000 0001: 2 такта HCLK (+ такт ожидания, вставленный снятием NWAIT)

1111 1110: 255 тактов HCLK (+ такт ожидания, вставленный снятием NWAIT)

1111 1111: Резерв.

— Биты 8:0 **ATTSETx[7:0]:** Время установки памяти атрибутов X.

При чтении и записи памяти атрибутов PC Card/NAND Flash определяют число тактов HCLK (+1) установки адреса перед подачей команд (NWE, NOE) для сокета X:

0000 0000: 1 такт HCLK

1111 1110: 255 тактов HCLK

1111 1111: Резерв

Регистр 4 времянки пространства Вв./Выв. (FSMC_PIO4)

Смещение адреса: 0xA000 0000 + 0xB0

По сбросу: 0xFCFCFCFC

Регистр чтения/записи FSMC_PIO4 содержит данные времянки доступа к пространству I/O 16-бит PC Card/CompactFlash.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IOHIZ[7:0]								IOHOLD[7:0]								IOWAIT[7:0]								IOSET[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:24 **IOHIZx[7:0]:** Время состояния HiZ шины данных I/O X.

Число тактов HCLK удержания состояния HiZ на шине данных после старта записи в пространство I/O (сокет X) PC Card. Работает только при записи:

0000 0000: 0 тактов HCLK

1111 1111: 255 тактов HCLK (по умолчанию после сброса)

— Биты 23:16 **IOHOLDx[7:0]**: Время удержания I/O X.

При чтении/записи памяти I/O PC Card определяют число тактов (HCLK+2) удержания адреса (и данных при записи) после снятия команды (NWE, NOE).

0000 0000: Резерв.

0000 0001: 1 такт HCLK.

1111 1111: 255 тактов HCLK (по умолчанию после сброса).

— Биты 15:9 **IOWAITx[7:0]:** Время ожидания I/O X.

При чтении и записи памяти I/O PC Card определяют минимальное число тактов HCLK (+1) выдачи команд (SMNWE, SMNOE) для сокета X. Длительность команд может расширяться сигналом ожидания (NWAIT) после заданного значения HCLK:

0000 0000: Резерв

0000 0001: 2 такта HCLK (+ такт ожидания, вставленный снятием NWAIT)

1111 1111: 255 тактов HCLK (+ такт ожидания, вставленный Картой снятием NWAIT)

— Биты 8:0 **IOSETx[7:0]:** Время установки I/O X.

При чтении и записи памяти I/O PC Card определяют число тактов HCLK (+1) установки адреса перед подачей команд (NWE, NOE) для сокета X:

0000 0000: 1 такт HCLK

1111 1111: 256 тактов HCLK (по умолчанию после сброса)

Регистры 2/3 результата ECC (FSMC_ECCR2/3)

Смещение адреса: 0xA000 0000 + 0x54 + 0x20 * (x – 1), x = 2 или 3

По сбросу: 0x0000 0000

Они содержат текущий код коррекции ошибки от модулей ECC. При чтении/записи данных из страницы NAND Flash по нормальному адресу они автоматически обрабатываются модулем этого банку. По концу чтения X байтов (в соответствии с полем ECCPS в регистре FSMC_PCRx) нужно прочитать число из регистра FSMC_ECCx и сравнить его с ранее сохранённым числом для тех же данных чтобы узнать о наличии ошибки и возможном её исправлении. Читать регистры FSMC_ECCRx нужно перед их очисткой снятием бита ECCEN. Для вычисления нового блока нужно установить бит ECCEN снова.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ECCx[31:0]																															
r																															

— Биты 31:24 **ECCx[31:0]**: Результат ECC

Таблица 135. Релевантные биты результата ECC.

ECCPS[2:0]	Размер страницы в байтах	Биты ECC
000	256	ECC[21:0]
001	512	ECC[23:0]
010	1024	ECC[25:0]
011	2048	ECC[27:0]
100	4096	ECC[29:0]
101	8192	ECC[31:0]

36.6.9. Карта регистров FSMC

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0000	FSMC_BCR1	Reserved												CBURSTRW	CPSIZE[2:0]			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	WRAPMOD	WAITPOL	BURSTEN	Reserved	FACCEN	MWID[1:0]		MTYP[0:1]		MUXEN	MBKEN
0008	FSMC_BCR2	Reserved												CBURSTRW	CPSIZE[2:0]			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	WRAPMOD	WAITPOL	BURSTEN	Reserved	FACCEN	MWID[1:0]		MTYP[0:1]		MUXEN	MBKEN
0010	FSMC_BCR3	Reserved												CBURSTRW	CPSIZE[2:0]			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	WRAPMOD	WAITPOL	BURSTEN	Reserved	FACCEN	MWID[1:0]		MTYP[0:1]		MUXEN	MBKEN
0018	FSMC_BCR4	Reserved												CBURSTRW	CPSIZE[2:0]			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	WRAPMOD	WAITPOL	BURSTEN	Reserved	FACCEN	MWID[1:0]		MTYP[0:1]		MUXEN	MBKEN
0004	FSMC_BTR1	Res.		ACCMOD[1:0]		DATLAT[3:0]				CLKDIV[3:0]				BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
000C	FSMC_BTR2	Res.		ACCMOD[1:0]		DATLAT[3:0]				CLKDIV[3:0]				BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
0014	FSMC_BTR3	Res.		ACCMOD[1:0]		DATLAT[3:0]				CLKDIV[3:0]				BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
001C	FSMC_BTR4	Res.		ACCMOD[1:0]		DATLAT[3:0]				CLKDIV[3:0]				BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
0104	FSMC_BWTR 1	Res.		ACC MOD [1:0]		Res.								BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
010C	FSMC_BWTR 2	Res.		ACC MOD [1:0]		Res.								BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0114	FSMC_BWTR 3	Res.		ACC MOD [1:0]		Res.								BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
011C	FSMC_BWTR 4	Res.		ACC MOD [1:0]		Res.								BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
0xA000 0060	FSMC_PCR2	Reserved												ECCPS[2:0]				TAR[2:0]				TCLR[2:0]			Res.	ECCEN		PWID[1:0]		PTYP	PBKEN	PWAITEN	Reserved
0xA000 0080	FSMC_PCR3	Reserved												ECCPS[2:0]				TAR[2:0]				TCLR[2:0]			Res.	ECCEN		PWID[1:0]		PTYP	PBKEN	PWAITEN	Reserved
0xA000 00A0	FSMC_PCR4	Reserved												ECCPS[2:0]				TAR[2:0]				TCLR[2:0]			Res.	ECCEN		PWID[1:0]		PTYP	PBKEN	PWAITEN	Reserved
0xA000 0064	FSMC_SR2	Reserved																								FEMPT	IFEN		ILEN	IREN	IFS	ILS	IRS
0xA000 0084	FSMC_SR3	Reserved																								FEMPT	IFEN		ILEN	IREN	IFS	ILS	IRS
0xA000 00A4	FSMC_SR4	Reserved																								FEMPT	IFEN		ILEN	IREN	IFS	ILS	IRS
0xA000 0068	FSMC_PMEM 2	MEMHIZ[7:0]								MEMHOLD[7:0]								MEMWAIT[7:0]								MEMSET[7:0]							
0xA000 0088	FSMC_PMEM 3	MEMHIZ[7:0]								MEMHOLD[7:0]								MEMWAIT[7:0]								MEMSET[7:0]							
0xA000 00A8	FSMC_PMEM 4	MEMHIZ[7:0]								MEMHOLD[7:0]								MEMWAIT[7:0]								MEMSET[7:0]							
0xA000 006C	FSMC_PATT2	ATTHIZ[7:0]								ATTHOLD[7:0]								ATTWAIT[7:0]								ATTSET[7:0]							
0xA000 008C	FSMC_PATT3	ATTHIZ[7:0]								ATTHOLD[7:0]								ATTWAIT[7:0]								ATTSET[7:0]							
0xA000 00AC	FSMC_PATT4	ATTHIZ[7:0]								ATTHOLD[7:0]								ATTWAIT[7:0]								ATTSET[7:0]							
0xA000 00B0	FSMC_PIO4	IOHIZ[7:0]								IOHOLD[7:0]								IOWAIT[7:0]								IOSET[7:0]							
0xA000 0074	FSMC_ECCR2	ECC[31:0]																															
0xA000 0094	FSMC_ECCR3	ECC[31:0]																															

37. Гибкий контроллер памяти (FMC)

FMC имеет три контроллера:

- NOR/PSRAM
- NAND/PC Card
- Синхронной DRAM (SDRAM/Mobile LPSDR SDRAM)

Только для STM32F42xxx и STM32F43xxx only.

37.1. Основные свойства FMC

Это интерфейс с асинхронной и синхронной статической памятью типа SDRAM и 16-бит PC Card. Он согласует передачи и времянку между AHB и внешней памятью по общим линиям адресов, данных и управления. Устройства адресуют через уникальный Chip Select. FMC использует только одно обращение к памяти за раз.

Главные свойства:

- Включает устройства:
  - Статическая память(SRAM)
  - NOR Flash/OneNAND Flash
  - PSRAM (4 банка)
  - 16-бит PC Card совместимые
  - Два банка NAND Flash памяти с аппаратной проверкой ECC до 8 Кбайт данных
- Интерфейс с синхронной памятью DRAM (SDRAM/Mobile LPSDR SDRAM)
- Пакетный режим доступа к синхронной памяти вроде NOR Flash, PSRAM и SDRAM
- Программируемые постоянные такты для синхронного и асинхронного доступа
- Шина данных 8-,16- или 32-бит ширины
- Независимый Chip Select на каждый банк памяти
- Независимая конфигурация каждого банка
- Выходы разрешения записи и выбора линий байтов для PSRAM, SRAM и SDRAM
- Внешнее асинхронное управление ожиданием
- FIFO записи данных 16 x33-бит глубины
- FIFO адреса 16x30-бит глубины
- Кешируемый FIFO чтения 6 x32-бит глубины (6 x14-бит теги адреса) для контроллера SDRAM.

У FMC есть 2 FIFO записи: FIFO Данных 16x33-бит глубины и FIFO Адреса 16x30-бит глубины.

- FIFO Данных хранит данные AHB для записи в память (до 32 бит) + 1 бит для передачи AHB (пакетный или нет последовательный режим)
- FIFO Адреса хранит адрес AHB (до 28 бит) плюс размер данных AHB (до 2 бит). В пакетном режиме хранится только адрес начала за исключением пересечения границ страницы (для PSRAM и SDRAM). В этом случае пакет AHB разбивается на две строки FIFO.

При запуске ножки FMC надо ставить программно. Свободные ножки FMC I/O свободны.

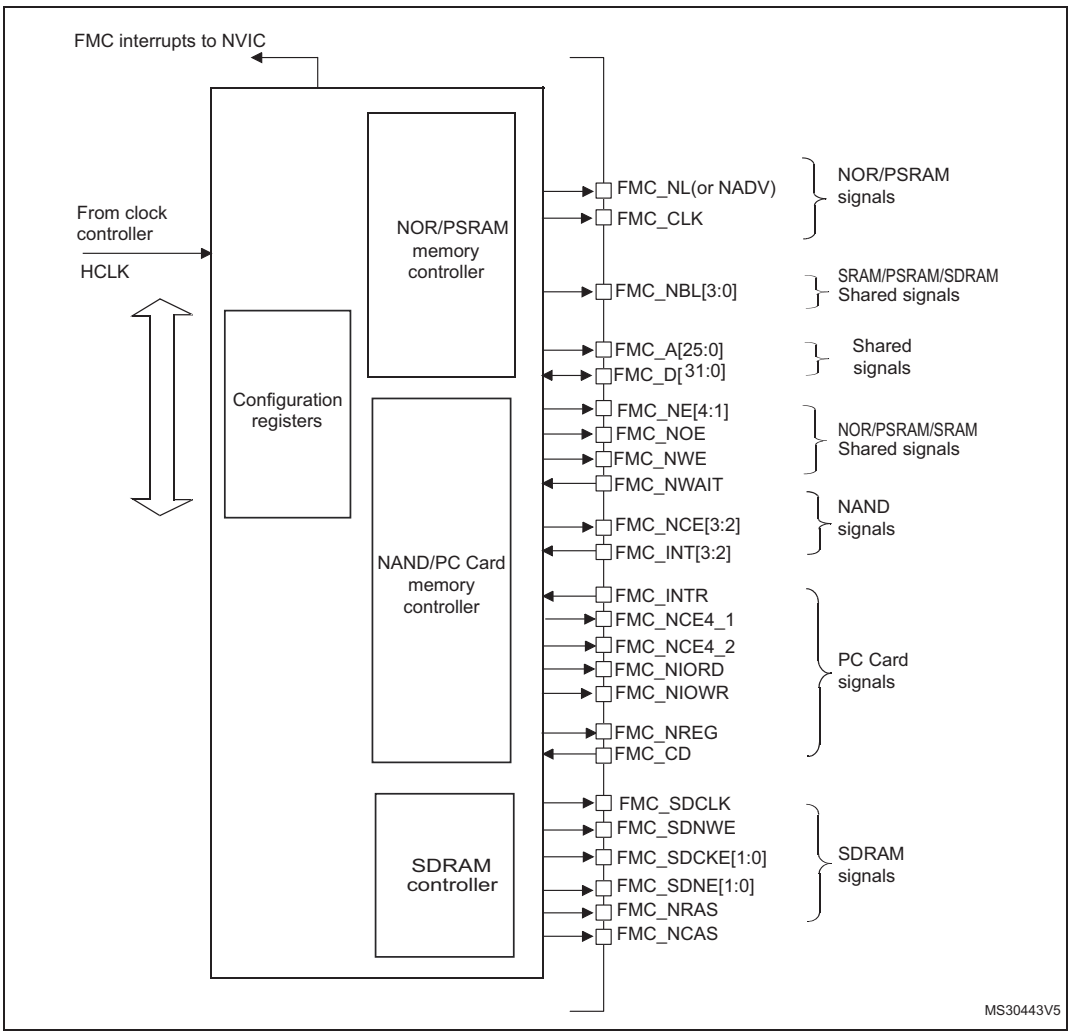
Регистры FMC, отвечающие за внешнее устройство, обычно ставят при загрузке и больше не меняют, хоть это и можно всегда.

37.2. Блок-схема FMC

FMC состоит из пяти главных блоков:

- Интерфейс AHB (включая регистры конфигурации FMC)
- Контроллер NOR Flash/PSRAM/SRAM
- Контроллер NAND Flash/PC Card
- Контроллер SDRAM
- Внешний интерфейс устройства

Рис. 456. Блок-схема FMC



37.3. Интерфейс AHB

AHB работает ведомым. Передачи AHB транслируются в протокол внешнего устройства. В частности, 32-бит передачи AHB могут разбиваться на последовательные 16- или 8-битовые. Между последовательными обращениями, FMC Chip Select (`FMC_NEx`) не переключается, за исключением доступа в расширенном режиме D.

FMC выдаёт ошибку AHB когда пробуется:

- Чтение или запись не разрешённого банка FMC (1 - 4).
- Чтение или запись банка NOR Flash при снятом бите `FMC_BCRx`/`FACCEN`.
- Чтение или запись банка PC Card при низкой входной ножке FMC_CD (Card Presence Detection).
- Запись в защищённый банк SDRAM (стоит бит `SDRAM_SDCRx`/`WP`).
- Запись в резервные адреса SDRAM

Реакция на ошибку AHB зависит от ведущего на шине:

- Если это CPU Cortex®-M4 с FPU, то выдаётся прерывание тяжёлого сбоя.
- Если это контроллер DMA, то выдаётся ошибка передачи DMA и этот канал выключается.

Тактируется FMC от тактов AHB (HCLK).

37.3.1. Поддерживаемая память и передачи

Общие правила

Если:

- Размер передачи AHB и данных в памяти равны: Всё ништяк.
- Размер передачи AHB больше данных в памяти:

FMC разбивает передачу AHB на более мелкие. FMC Chip Select (FMC_NEx) между частями не переключается.

- Размер передачи AHB меньше данных в памяти:

  Всё зависит от типа внешнего устройства:

- С выбором номера байта (SRAM, ROM, PSRAM, SDRAM)

  FMC передаёт записываемый байт по линиям байта BL[3:0].

  Он адресуется по NBL[3:0].

  При чтении получают всё (NBL[3:0] низкие), бесполезные байты выбрасываются.

- Без выбора номера байта (16-бит NOR и NAND Flash)

  Пишут полусловами, при чтении ненужный байт выбрасывается.

Регистры конфигурации

Всё есть ниже.

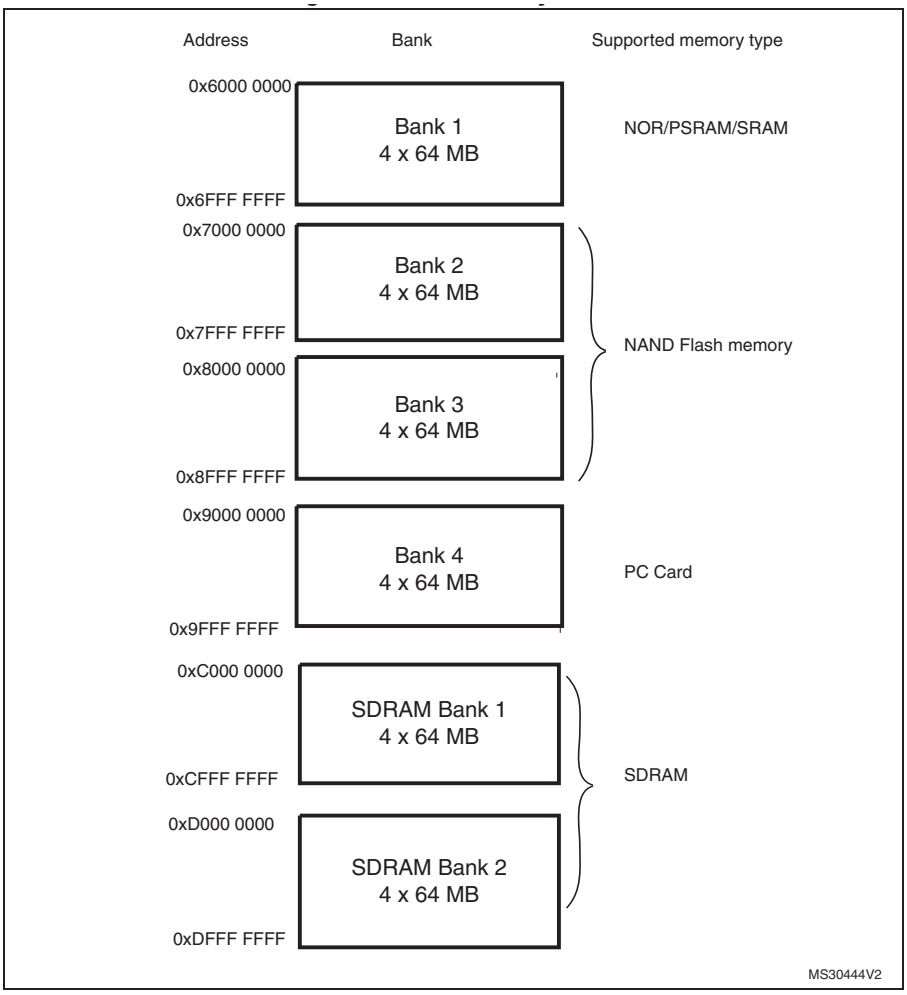
37.4. Размещение адресов внешних устройств

С точки зрения FMC внешняя память разделена на 6 банков по 256 Мбайт:

- Банк 1 адресует до 4 NOR Flash или PSRAM устройств. Он разбит на 4 NOR/PSRAM участка с 4 своими Chip Select:
  - Банк 1 - NOR/PSRAM 1
  - Банк 1 - NOR/PSRAM 2
  - Банк 1 - NOR/PSRAM 3
  - Банк 1 - NOR/PSRAM 4
- Банки 2 и 3 адресуют NAND Flash (1 устройство на банк)
- Банк 4 адресует PC Card
- Банки 5 и 6 адресуют SDRAM (1 устройство на банк).

Тип памяти каждого банка задают в регистре конфигурации.

Банки памяти FMC



37.4.1. Адреса памяти NOR/PSRAM

Используются биты HADDR[27:26].

Таблица 254. Выбор банка NOR/PSRAM

HADDR[27:26](1)	Selected bank
0	Bank 1 - NOR/PSRAM 1
1	Bank 1 - NOR/PSRAM 2
10	Bank 1 - NOR/PSRAM 3
11	Bank 1 - NOR/PSRAM 4

1. HADDR are internal AHB address lines that are translated to external memory.

Биты HADDR[25:0] содержат адрес байтовый внешней памяти, а память адресуется словами. Так что реальный адрес обращения меняется.

Таблица 255. Адреса внешней памяти NOR/PSRAM

Ширина памяти(1)	Адрес данных для памяти	Макс. ёмкость памяти (бит)
8-bit	HADDR[25:0]	64 Mbyte x 8 = 512 Mbit
16-bit	HADDR[25:1] >> 1	64 Mbyte/2 x 16 = 512 Mbit
32-bit	HADDR[25:2] >> 2	64 Mbyte/4 x 32 = 512 Mbit

1. При ширине памяти 16-бит FMC сам из HADDR[25:1] делает внешние адреса FMC_A[24:0]. При ширине памяти 32-бит FMC сам из HADDR[25:2] делает внешние адреса.

FMC_A[0] должен быть подключён а адресу внешней памяти A[0] независимо от ширины памяти.

Поддержка заворота для NOR Flash/PSRAM

Групповой режим с заворотом у синхронной памяти не поддерживается. Она должна стоять в линейном групповом режиме неопределённой длины.

37.4.2. Адреса памяти NAND Flash/PC Card

Тут есть три разделённых на области банка.

Таблица 256. Карта памяти NAND/PC Card и регистры времянки

Арес начала	Адрес конца	Банк FMC	Пространство	Регистр
0x9C00 0000	0x9FFF FFFF	Bank 4 - PC card	I/O	FMC_PIO4 (0xB0)
0x9800 0000	0x9BFF FFFF		Attribute	FMC_PATT4 (0xAC)
0x9000 0000	0x93FF FFFF		Common	FMC_PMEM4 (0xA8)
0x8800 0000	0x8BFF FFFF	Bank 3 - NAND Flash	Attribute	FMC_PATT3 (0x8C)
0x8000 0000	0x83FF FFFF		Common	FMC_PMEM3 (0x88)
0x7800 0000	0x7BFF FFFF	Bank 2- NAND Flash	Attribute	FMC_PATT2 (0x6C)
0x7000 0000	0x73FF FFFF		Common	FMC_PMEM2 (0x68)

У NAND Flash пространство атрибутов и общее подразделяются на три секции, лежащих в нижних 256 Кбайтах:

- Данных (первые 64 Кбайт)
- Команд (вторые 64 Кбайт)
- Адресов (следующие 128 Кбайт)

Таблица 257. Выбор банка NAND

Секция	HADDR[17:16]	Адреса
Адресов	1X	0x020000-0x03FFFF
Команд	1	0x010000-0x01FFFF
Данных	0	0x000000-0x0FFFF

При доступе к NAND Flash используются 3 секции:

- **Команду** пишут в любую область секции Команд.
- **Адрес доступа** пишут в любую область секции Адресов. Для записи полного адреса может понадобиться несколько обращений.
- **Данные** читают или пишут по любому адресу секции Данных.

Адрес секции данных инкрементируется сам после обращения.

37.4.3. Адреса памяти SDRAM

Один из двух банков выбирают битом HADDR[28].

Таблица 258. Выбор банка SDRAM

HADDR[28]	Банк	Регистр управления	Регистр времянки
0	SDRAM Bank1	FMC_SDCR1	FMC_SDTR1
1	SDRAM Bank2	FMC_SDCR2	FMC_SDTR2

Таблица 259. Карта адресов SDRAM с 4 банками, 13-бит строками и 11-бит колонками

Ширина памяти(1)	Внутренний банк	Адрес строки	Адрес колонки(2)	Макс. ёмкость (Mbyte)
8-bit	HADDR[25:24]	HADDR[23:11]	HADDR[10:0]	64 Mbyte: 4 x 8K x 2K
16-bit	HADDR[26:25]	HADDR[24:12]	HADDR[11:1]	128 Mbyte: 4 x 8K x 2K x 2
32-bit	HADDR[27:26]	HADDR[25:13]	HADDR[12:2]	256 Mbyte: 4 x 8K x 2K x 4

1. Для 16-бит памяти FMC делает внешний адрес из линий HADDR[11:1] AHB. Для 32-бит памяти используются линии HADDR[12:2]. FMC_A[0] должен быть подключён а адресу внешней памяти A[0] независимо от ширины памяти.
2. АвтоПредзаряд не поддерживается. FMC_A[10] должен быть подключён к A[10] но всегда будет низким.

Биты HADDR[27:0] транслируются на внешнюю SDRAM в зависимости от конфигурации её контроллера:

- Размер данных:8, 16 или 32 бита
- Размер строки:11, 12 или 13 бит
- Размер колонки: 8, 9, 10 или 11 бит
- Число внутренних банков: два или четыре

Таблица 260. Адреса SDRAM с 8-бит шиной данных(1)(2)

Row size configuration	HADDR(AHB Internal Address Lines)																										
	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
11-bit row size configuration	Res.							Bank [1:0]	Row[10:0]												Column[7:0]							
	Res.							Bank [1:0]	Row[10:0]												Column[8:0]							
	Res.							Bank [1:0]	Row[10:0]												Column[9:0]							
	Res.							Bank [1:0]	Row[10:0]												Column[10:0]							
12-bit row size configuration	Res.							Bank [1:0]	Row[11:0]												Column[7:0]							
	Res.							Bank [1:0]	Row[11:0]												Column[8:0]							
	Res.							Bank [1:0]	Row[11:0]												Column[9:0]							
	Res.							Bank [1:0]	Row[11:0]												Column[10:0]							
13-bit row size configuration	Res.							Bank [1:0]	Row[12:0]												Column[7:0]							
	Res.							Bank [1:0]	Row[12:0]												Column[8:0]							
	Res.							Bank [1:0]	Row[12:0]												Column[9:0]							
	Res.							Bank [1:0]	Row[12:0]												Column[10:0]							

1. BANK[1:0] это Адрес банка BA[1:0]. При работе только 2 банков BA1 всегда ставят в ‘0’.
2. Обращение к Резерву (Res.) выдаёт ошибку AHB.

Таблица 261. Адреса SDRAM с 16-бит шиной данных(1)(2)

Row size Configuration	HADDR(AHB address Lines)																										
	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
11-bit row size configuration	Res.						Bank [1:0]		Row[10:0]											Column[7:0]								BM0(3)
	Res.						Bank [1:0]		Row[10:0]											Column[8:0]								BM0
	Res.						Bank [1:0]		Row[10:0]											Column[9:0]								BM0
	Res.						Bank [1:0]		Row[10:0]											Column[10:0]								BM0
12-bit row size configuration	Res.						Bank [1:0]		Row[11:0]											Column[7:0]								BM0
	Res.						Bank [1:0]		Row[11:0]											Column[8:0]								BM0
	Res.						Bank [1:0]		Row[11:0]											Column[9:0]								BM0
	Res.						Bank [1:0]		Row[11:0]											Column[10:0]								BM0
13-bit row size configuration	Res.						Bank [1:0]		Row[12:0]											Column[7:0]								BM0
	Res.						Bank [1:0]		Row[12:0]											Column[8:0]								BM0
	Res.						Bank [1:0]		Row[12:0]											Column[9:0]								BM0
	Res.						Bank [1:0]		Row[12:0]											Column[10:0]								BM0

1. BANK[1:0] это Адрес банка BA[1:0]. При работе только 2 банков BA1 всегда ставят в ‘0’.
2. Обращение к Резерву (Res.) выдаёт ошибку AHB.
3. BM0 это маска байта для 16-бит доступа.

Таблица 262. Адреса SDRAM с 32-бит шиной данных(1)(2)

Row size configuration	HADDR(AHB address Lines)																										
	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
11-bit row size configuration	Res.						Bank [1:0]		Row[10:0]											Column[7:0]							BM[1:0](3)	
	Res.						Bank [1:0]		Row[10:0]											Column[8:0]							BM[1:0	
	Res.						Bank [1:0]		Row[10:0]											Column[9:0]							BM[1:0	
	Res.						Bank [1:0]		Row[10:0]											Column[10:0]							BM[1:0	
12-bit row size configuration	Res.						Bank [1:0]		Row[11:0]											Column[7:0]							BM[1:0	
	Res.						Bank [1:0]		Row[11:0]											Column[8:0]							BM[1:0	
	Res.						Bank [1:0]		Row[11:0]											Column[9:0]							BM[1:0	
	Res.						Bank [1:0]		Row[11:0]											Column[10:0]							BM[1:0	
13-bit row size configuration	Res.						Bank [1:0]		Row[12:0]											Column[7:0]							BM[1:0	
	Res.						Bank [1:0]		Row[12:0]											Column[8:0]							BM[1:0	
	Res.						Bank [1:0]		Row[12:0]											Column[9:0]							BM[1:0	
	Bank [1:0]						Row[12:0]													Column[10:0]							BM[1:0	

1. BANK[1:0] это Адрес банка BA[1:0]. При работе только 2 банков BA1 всегда ставят в ‘0’.
2. Обращение к Резерву (Res.) выдаёт ошибку AHB.
3. BM[1:0] это маска байта для 32-бит доступа.

37.5. Контроллер NOR Flash/PSRAM

FMC управляет времянкой сигналов для:

- Асинхронных SRAM и ROM
  - 8 бит
  - 16 бит
  - 32 бита
- PSRAM (Cellular RAM)
  - Асинхронный режим
  - Пакетный синхронный режим
  - Мультиплексированный и нет
- NOR Flash
  - Асинхронный режим
  - Пакетный синхронный режим
  - Мультиплексированный и нет

Сигналы Chip Select, NE[4:1], по одному на банк. Остальные сигналы общие.

Программируемая времянка:

- Состояния ожидания (до 15)
- Такты обратной связи шины (до 15)
- Задержка разрешения выхода и разрешения записи (до 15)
- Независимые протокол и времянки для разных типов памяти
- Непрерывный вывод тактов (FMC_CLK).

Такты FMC_CLK кратны тактам HCLK. Бит `FMC_BCR1`/`CCKEN` разрешает подачу их на внешнюю память при синхронном или синхронном и асинхронном доступе:

- Если `CCKEN` снят, то только при синхронном Чтении/записи.
- Если `CCKEN` стоит, то при обоих типах доступа. Для выдачи непрерывных тактов FMC_CLK Банк 1 надо ставить в синхронный режим. При этом ширина данных памяти (MWID) должна быть равна ширине данных AHB, иначе частота FMC_CLK изменится в соответствии с передачами данных AHB.

Каждый банк имеет по 64 Мбайт и свои регистры.

Таблица 263. Программируемые параметры доступа к NOR/PSRAM

Параметр	Функция	Доступ	Такты	Min.	Max.
Подача адреса	Длительность этой фазы	Асинхронный	AHB (HCLK)	0	15
Удержание адреса	Длительность этой фазы	Асинхронный, мультиплекс I/O	AHB (HCLK)	1	15
Подача данных	Длительность этой фазы	Асинхронный	AHB (HCLK)	1	256
Возврат шины	Длительность этой фазы	Асинхронный и синхронный	AHB (HCLK)	0	15
Делитель тактов	Число тактов HCLK на такт CLK	Синхронный	AHB (HCLK)	2	16
Задержка данных	Число тактов до подачи первого данного пакета	Синхронный	Памяти (CLK)	2	17

37.5.1. Сигналы интерфейса внешней памяти

Префикс “N” говорит, что сигнал активен низким. Максимальная ёмкость 512 Мбит.

Таблица 264. Не-мультиплексированные I/O NOR Flash

Имя сигнала	I/O	Функция
CLK	O	Такты (синхронный доступ)
A[25:0]	O	Шина адреса
D[31:0]	I/O	Двунаправленная шина данных
NE[x]	O	Chip Select, x = 1..4
NOE	O	Разрешение выхода
NWE	O	Разрешение записи
NL(=NADV)	O	Разрешение считывания (он же адрес верен, NADV)
NWAIT	I	NOR Flash ждёт сигнала

Таблица 265. 16-бит мультиплексированные I/O NOR Flash

Имя сигнала	I/O	Функция
CLK	O	Такты (синхронный доступ)
A[25:16]	O	Шина адреса
AD[15:0]	I/O	16-бит мультиплексированнай двунаправленная шина адреса/данных A[15:0]/D[15:0]
NE[x]	O	Chip Select, x = 1..4
NOE	O	Разрешение выхода
NWE	O	Разрешение записи
NL(=NADV)	O	Разрешение считывания (он же адрес верен, NADV)
NWAIT	I	NOR Flash ждёт сигнала

Таблица 266. Не-мультиплексированные I/O PSRAM/SRAM

Имя сигнала	I/O	Функция
CLK	O	Такты (синхронный доступ)
A[25:0]	O	Шина адреса
D[31:0]	I/O	Двунаправленная шина данных
NE[x]	O	Chip Select, x = 1..4 (called NCE by PSRAM (Cellular RAM i.e. CRAM))
NOE	O	Разрешение выхода
NWE	O	Разрешение записи
NL(= NADV)	O	Адрес верен только для входа PSRAM (сигнал памяти: NADV)
NWAIT	I	NOR Flash ждёт сигнала
NBL[3]	O	Разрешение байта 3 (сигнал памяти: NUB)
NBL[2]	O	Разрешение байта 2 (сигнал памяти: NLB)
NBL[1]	O	Разрешение байта 1 (сигнал памяти: NLB)
NBL[0]	O	Разрешение байта 0 (сигнал памяти: NLB)

Таблица 267. Мультиплексированные 16-бит I/O PSRAM/SRAM

Имя сигнала	I/O	Функция
CLK	O	Такты (синхронный доступ)
A[25:16]	O	Шина адреса
AD[15:0]	I/O	16-бит мультиплексированнай двунаправленная шина адреса/данных A[15:0]/D[15:0]
NE[x]	O	Chip Select, x = 1..4
NOE	O	Разрешение выхода
NWE	O	Разрешение записи
NL(=NADV)	O	Разрешение считывания (он же адрес верен, NADV)
NWAIT	I	NOR Flash ждёт сигнала
NBL[1]	O	Разрешение байта 1 (сигнал памяти: NLB)
NBL[0]	O	Разрешение байта 0 (сигнал памяти: NLB)

37.5.2. Поддерживаемая память и передачи

Неподдерживаемые передачи выделены красным.

Таблица 268. NOR Flash/PSRAM:

Уст-во	Режим	R/W	Ширина AHB	Ширина данных	Можно?	Коммент
NOR Flash (мультиплекс. и не-мультиплекс. I/O)	Асинхр.	R	8	16	Y	
	Асинхр.	W	8	16	N	
	Асинхр.	R	16	16	Y	
	Асинхр.	W	16	16	Y	
	Асинхр.	R	32	16	Y	Два обращения FMC
	Асинхр.	W	32	16	Y	Два обращения FMC
	Асинхр. стр.	R	-	16	N	Не поддерживается.
	Синхр.	R	8	16	N	
	Синхр.	R	16	16	Y	
	Синхр.	R	32	16	Y	

PSRAM (мультиплекс. и не-мультиплекс. I/O)	Асинхр.	R	8	16	Y	
	Асинхр.	W	8	16	Y	Линии NBL[1:0]
	Асинхр.	R	16	16	Y	
	Асинхр.	W	16	16	Y	
	Асинхр.	R	32	16	Y	Два обращения FMC
	Асинхр.	W	32	16	Y	Два обращения FMC
	Асинхр. стр.	R	-	16	N	Не поддерживается.
	Синхр.	R	8	16	N	
	Синхр.	R	16	16	Y	
	Синхр.	R	32	16	Y	
	Синхр.	W	8	16	Y	Линии NBL[1:0]
	Синхр.	W	16/32	16	Y	
SRAM и ROM	Асинхр.	R	8 / 16	16	Y	
	Асинхр.	W	8 / 16	16	Y	Линии NBL[1:0]
	Асинхр.	R	32	16	Y	Два обращения FMC
	Асинхр.	W	32	16	Y	Два обращения FMC. Линии NBL[1:0]

37.5.3. Общие правила времянки

Синхронизация сигналов

- Все выходные сигналы контроллера меняются по переднему фронту HCLK
- В синхронном режиме все выходные сигналы контроллера меняются по переднему фронту HCLK. Независимо от CLKDIV выходы меняются так:
  - NOEL/NWEL/ NEL/NADVL/ NADVH /NBLL/ Допустимого адреса по заднему фронту FMC_CLK.
  - NOEH/ NWEH / NEH/ NOEH/NBLH/ Недопустимого адреса по переднему фронту FMC_CLK.

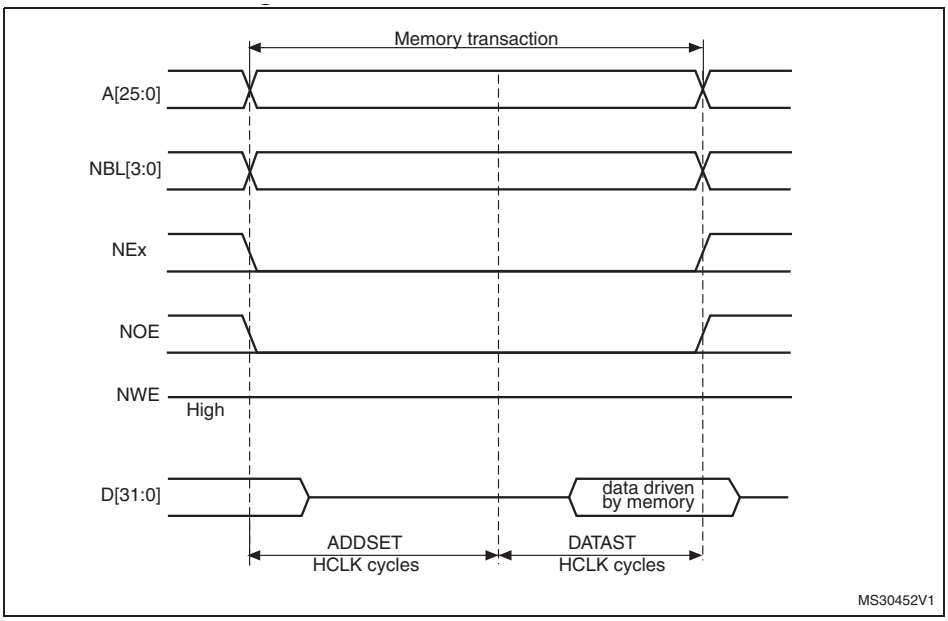
37.5.4. Асинхронные передачи контроллера NOR Flash/PSRAM

Асинхронная статическая память (NOR Flash, PSRAM, SRAM)

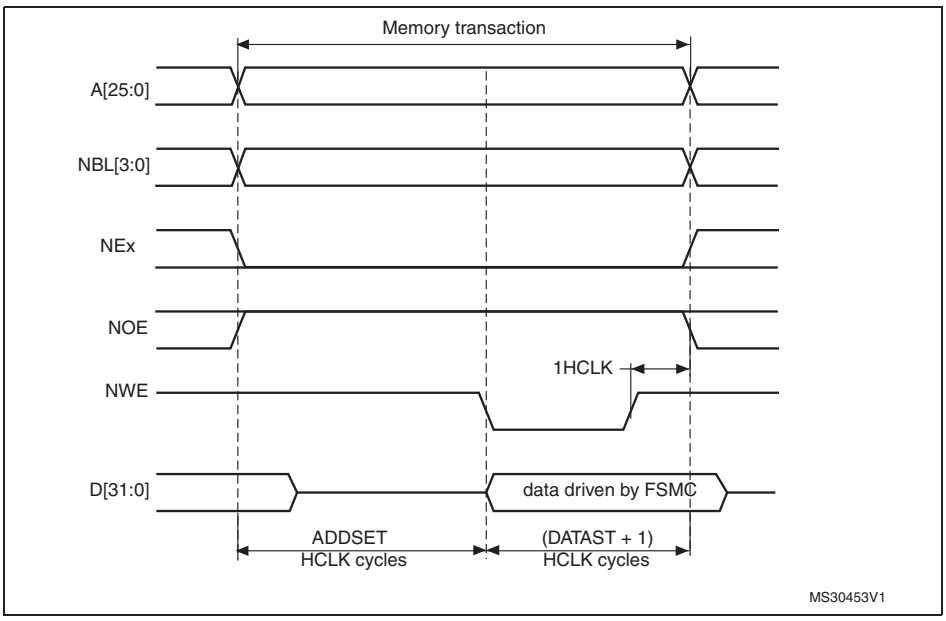
- Сигналы синхронизируются внутренними тактами HCLK. На память они не идут.
- FMC читает данные до снятия сигнала NOE.
- Расширенные режимы (A, B, C и D) включаются битом FMC_BCRx/EXTMOD. Их можно смешивать, например, чтение в A и запись в B.
- При выключении расширенных режимов FMC может работать в Mode 1 и Mode 2:
  - Mode 1 по умолчанию для SRAM/PSRAM (FMC_BCRx/MTYP[1:0] = 0x0 или 0x01)
  - Mode 2 по умолчанию для NOR ((FMC_BCRx/MTYP[1:0] = 0x10).

Mode 1 - SRAM/PSRAM (CRAM)

Чтение в Mode 1



Запись в Mode 1



Один такт HCLK после записи гарантирует удержание адреса и данных за передним фронтом NWE. Из-за этого значение DATAST должно быть больше 0.

Таблица 269. Поля FMC_BCRx

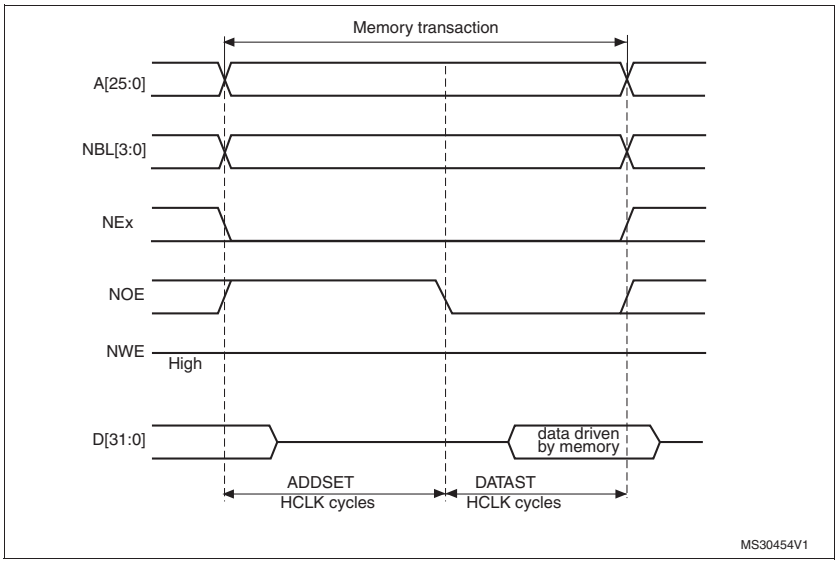
Номер бита	Имя его	Что писать
31-21	Резерв	0x000
20	CCLKEN	То, что надо
19	CBURSTRW	0x0 (в асинхронном режиме эффекта нет)
18:16	CPSIZE	0x0 (в асинхронном режиме эффекта нет)
15	ASYNCWAIT	1 - если память поддерживает. Иначе - 0.
14	EXTMOD	0x0
13	WAITEN	0x0 (в асинхронном режиме эффекта нет)
12	WREN	То, что надо
11	WAITCFG	Хоть что-нибудь, если очень уж хочется.
10	WRAPMOD	0x0
9	WAITPOL	Осмыслено только при бите 15 = 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	А-а-а, всё равно.
5-4	MWID	То, что надо
3-2	MTYP[1:0]	То, что надо, кроме 0x2 (NOR Flash)
1	MUXE	0x0
0	MBKEN	0x1

Таблица 270. Поля FMC_BTRx

Номер бита	Имя его	Что писать
31:30	Резерв	0x0
29-28	ACCMOD	А-а-а, всё равно.
27-24	DATLAT	А-а-а, всё равно.
23-20	CLKDIV	А-а-а, всё равно.
19-16	BUSTURN	Время от высокого NEx до низкого NEx (BUSTURN HCLK)
15-8	DATAST	Время 2-й фазы доступа (DATAST+1 такт HCLK для записи, DATAST тактов HCLK - чтение).
7-4	ADDHLD	А-а-а, всё равно.
3-0	ADDSET	Время 1-й фазы доступа (ADDSET тактов HCLK). Минимум ADDSET = 0.

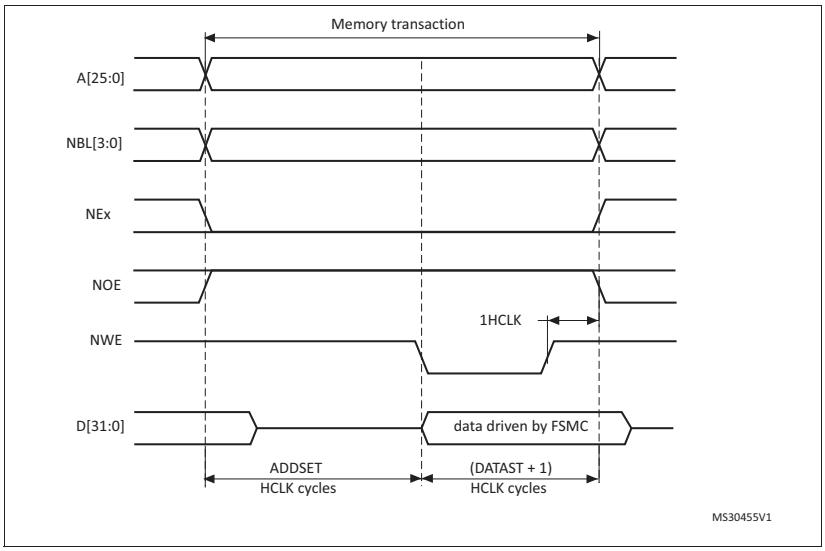
Mode A - SRAM/PSRAM (CRAM), переключение OE

Чтение - Mode A



1. NBL[3:0] при чтении держат низким

Запись - Mode A



От режима 1 отличается переключением NOE и независимым временем чтения и записи.

Таблица 271. Поля FMC_BCRx

Номер бита	Имя его	Что писать
31-21	Reserved	0x000
20	CCLKEN	Как надо
19	CBURSTRW	0x0 (в асинхронном режиме эффекта нет)
18:16	CPSIZE	0x0 (в асинхронном режиме эффекта нет)
15	ASYNCWAIT	1 - если память поддерживает. Иначе - 0.
14	EXTMOD	0x1
13	WAITEN	0x0 (в асинхронном режиме эффекта нет)
12	WREN	Как надо
11	WAITCFG	Всё равно
10	WRAPMOD	0x0
9	WAITPOL	Осмыслено только при бите 15 = 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	Всё равно
5-4	MWID	Как надо
3-2	MTYP[1:0]	Как надо, исключая 0x2 (NOR Flash)
1	MUXEN	0x0
0	MBKEN	0x1

Таблица 272. Поля FMC_BTRx

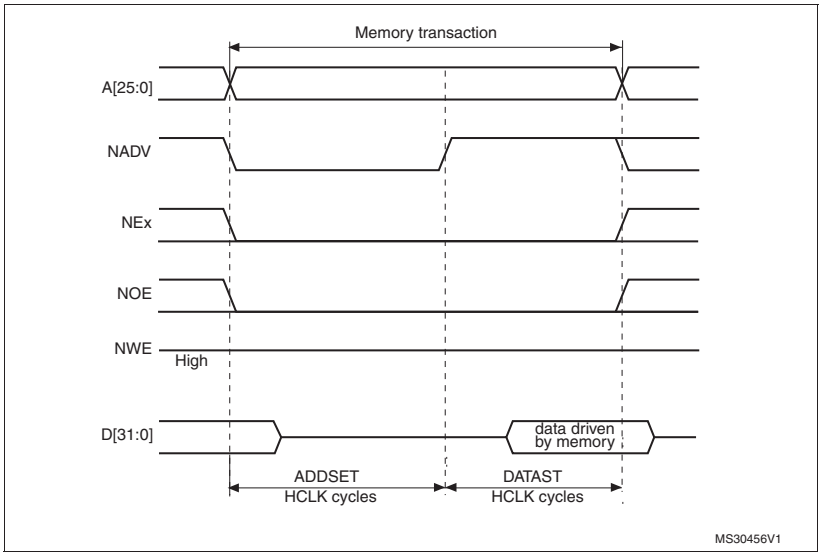
Номер бита	Имя его	Что писать
31:30	Резерв	0x0
29-28	ACCMOD	0x0
27-24	DATLAT	A-a-a, всё равно.
23-20	CLKDIV	A-a-a, всё равно.
19-16	BUSTURN	Время от высокого NEx до низкого NEx (BUSTURN HCLK)
15-8	DATAST	Время 2-й фазы доступа (DATAST такт HCLK для записи, DATAST тактов HCLK - чтение).
7-4	ADDHLD	A-a-a, всё равно.
3-0	ADDSET	Время 1-й фазы доступа (ADDSET тактов HCLK). Минимум ADDSET = 0.

Таблица 273. Поля FMC_BWTRx

Номер бита	Имя его	Что писать
31:30	Резерв	0x0
29-28	ACCMOD	0x0
27-24	DATLAT	A-a-a, всё равно.
23-20	CLKDIV	A-a-a, всё равно.
19-16	BUSTURN	Время от высокого NEx до низкого NEx (BUSTURN HCLK)
15-8	DATAST	Время 2-й фазы доступа (DATAST такт HCLK для записи, DATAST тактов HCLK - чтение).
7-4	ADDHLD	A-a-a, всё равно.
3-0	ADDSET	Время 1-й фазы доступа (ADDSET тактов HCLK). Минимум ADDSET = 0.

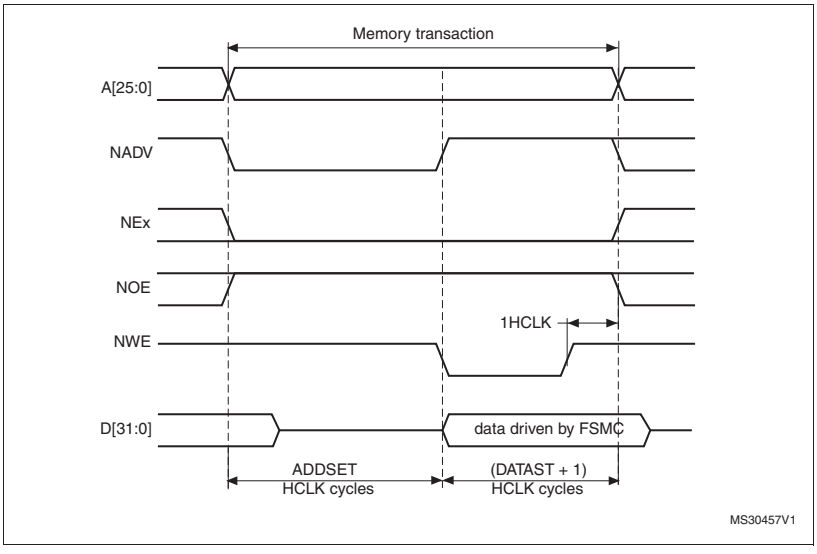
Mode 2/B - NOR Flash

Mode 2 и Mode B - Чтение

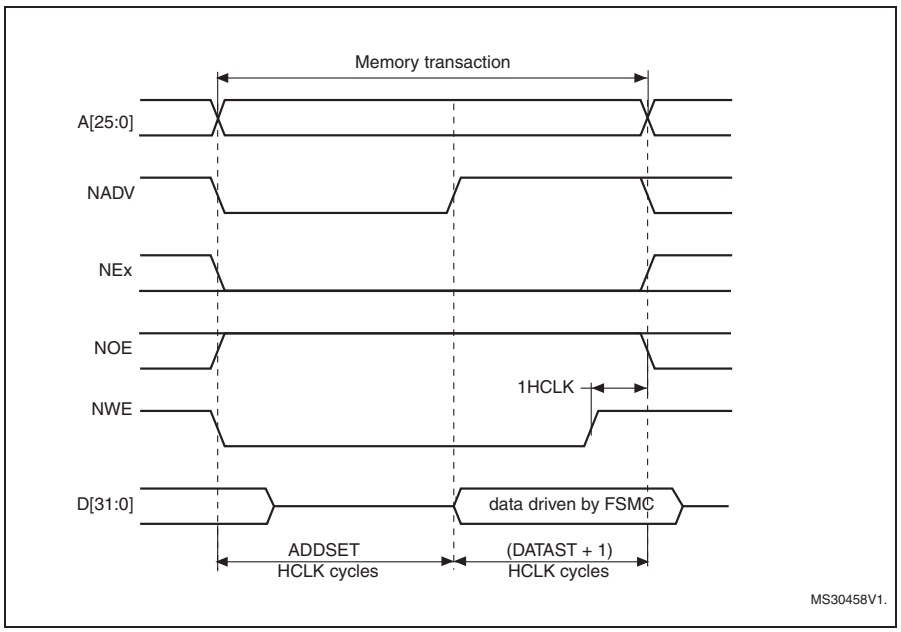


1. NBL[3:0] при чтении держат низким

Mode 2 - Запись



Mode B - Запись



От режима 1 отличается переключением NWE независимым временем чтения и записи а режиме B.

Таблица 274. Поля FMC_BCRx

Номер бита	Имя его	Что писать
31-21	Reserved	0x000
20	CCLKEN	Как надо
19	CBURSTRW	0x0 (в асинхронном режиме эффекта нет)
18:16	CPSIZE	0x0 (в асинхронном режиме эффекта нет)
15	ASYNCWAIT	1 - если память поддерживает. Иначе - 0.
14	EXTMOD	0x1 для mode B, 0x0 для mode 2
13	WAITEN	0x0 (в асинхронном режиме эффекта нет)
12	WREN	Как надо
11	WAITCFG	Всё равно
10	WRAPMOD	0x0
9	WAITPOL	Осмыслено только при бит 15 = 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	0x1
5-4	MWID	Как надо
3-2	MTYP[1:0]	0x2 (NOR Flash)
1	MUXEN	0x0
0	MBKEN	0x1

Таблица 275. Поля FMC_BTRx

Номер бита	Имя его	Что писать
31:30	Резерв	0x0
29-28	ACCMOD	0x1 в расширенном режиме
27-24	DATLAT	A-a-a, всё равно.
23-20	CLKDIV	A-a-a, всё равно.
19-16	BUSTURN	Время от высокого NEx до низкого NEx (BUSTURN HCLK)
15-8	DATAST	Время 2-й фазы доступа (DATAST такт HCLK для записи, DATAST тактов HCLK - чтение).
7-4	ADDHLD	A-a-a, всё равно.
3-0	ADDSET	Время 1-й фазы доступа (ADDSET тактов HCLK). Минимум ADDSET = 0.

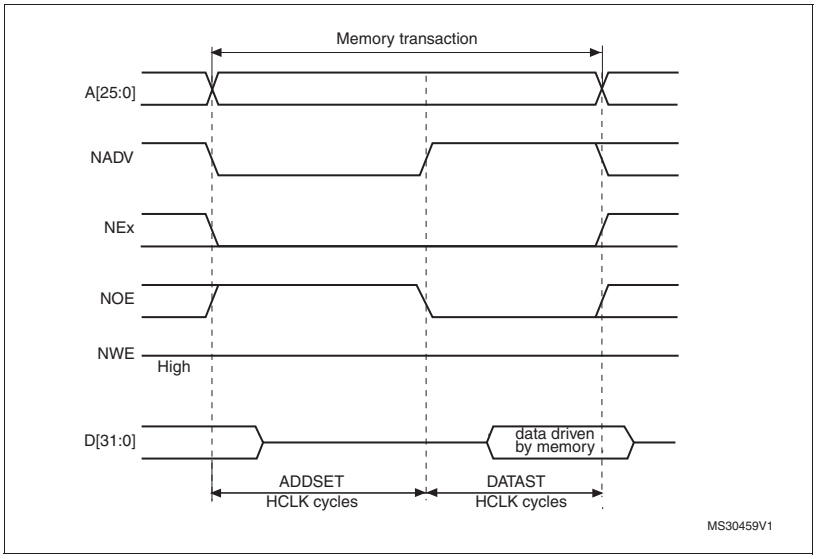
Таблица 276. Поля FMC_BWTRx

Номер бита	Имя его	Что писать
31:30	Резерв	0x0
29-28	ACCMOD	0x0
27-24	DATLAT	А-а-а, всё равно.
23-20	CLKDIV	А-а-а, всё равно.
19-16	BUSTURN	Время от высокого NEx до низкого NEx (BUSTURN HCLK)
15-8	DATAST	Время 2-й фазы доступа (DATAST такт HCLK для записи, DATAST тактов HCLK - чтение).
7-4	ADDHLD	А-а-а, всё равно.
3-0	ADDSET	Время 1-й фазы доступа (ADDSET тактов HCLK). Минимум ADDSET = 0.

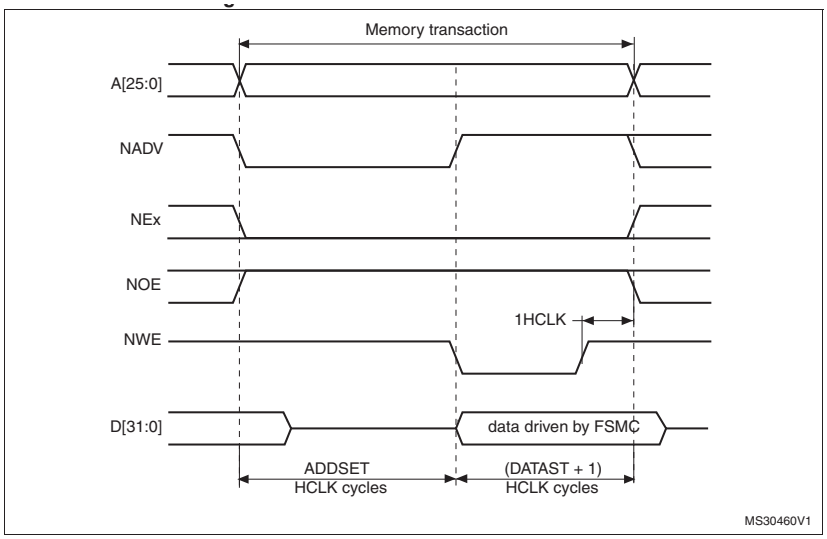
Регистр FMC_BWTRx действует только в режиме mode B, иначе там лажа.

Mode C - NOR Flash - переключение OE

Mode C - чтение



Mode C - запись



От режима 1 отличается переключением NOE и независимым временем чтения и записи.

Таблица 277. Поля FMC_BCRx

Номер бита	Имя его	Что писать
31-21	Reserved	0x000
20	CCLKEN	Как надо
19	CBURSTRW	0x0 (в асинхронном режиме эффекта нет)
18:16	CPSIZE	0x0 (в асинхронном режиме эффекта нет)
15	ASYNCWAIT	1 - если память поддерживает. Иначе - 0.
14	EXTMOD	0x1
13	WAITEN	0x0 (в асинхронном режиме эффекта нет)

12	WREN	Как надо
11	WAITCFG	Всё равно
10	WRAPMOD	0x0
9	WAITPOL	Осмыслено только при бит 15 = 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	0x1
5-4	MWID	Как надо
3-2	MTYP[1:0]	0x2 (NOR Flash)
1	MUXEN	0x0
0	MBKEN	0x1

Таблица 278. Поля FMC_BTRx

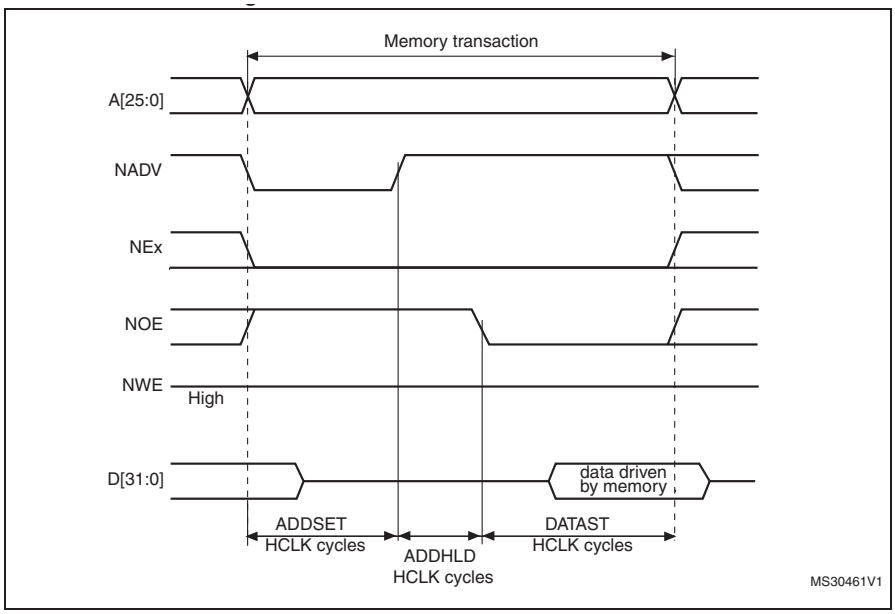
Номер бита	Имя его	Что писать
31:30	Резерв	0x0
29-28	ACCMOD	0x2
27-24	DATLAT	0x0
23-20	CLKDIV	0x0
19-16	BUSTURN	Время от высокого NEx до низкого NEx (BUSTURN HCLK)
15-8	DATAST	Время 2-й фазы доступа (DATAST такт HCLK для записи, DATAST тактов HCLK - чтение).
7-4	ADDHLD	A-a-a, всё равно.
3-0	ADDSET	Время 1-й фазы доступа (ADDSET тактов HCLK). Минимум ADDSET = 0.

Таблица 279. Поля FMC_BWTRx

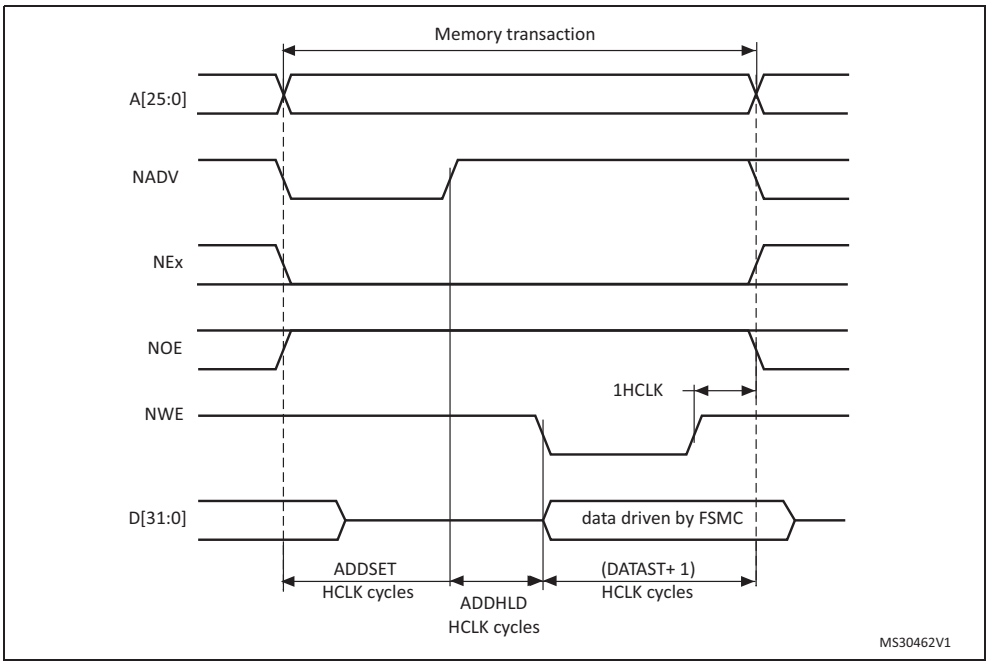
Номер бита	Имя его	Что писать
31:30	Резерв	0x0
29-28	ACCMOD	0x2
27-24	DATLAT	A-a-a, всё равно.
23-20	CLKDIV	A-a-a, всё равно.
19-16	BUSTURN	Время от высокого NEx до низкого NEx (BUSTURN HCLK)
15-8	DATAST	Время 2-й фазы доступа (DATAST такт HCLK для записи, DATAST тактов HCLK - чтение).
7-4	ADDHLD	A-a-a, всё равно.
3-0	ADDSET	Время 1-й фазы доступа (ADDSET тактов HCLK). Минимум ADDSET = 0.

Mode D - Асинхронный доступ с расширенным адресом

Mode D - чтение



Mode D - запись



От режима 1 отличается переключением NOE после изменения NADV и независимым временем чтения и записи.

Таблица 280. Поля FMC_BCRx

Номер бита	Имя его	Что писать
31-21	Reserved	0x000
20	CCLKEN	Как надо
19	CBURSTRW	0x0 (в асинхронном режиме эффекта нет)
18:16	CPSIZE	0x0 (в асинхронном режиме эффекта нет)
15	ASYNCWAIT	1 - если память поддерживает. Иначе - 0.
14	EXTMOD	0x1
13	WAITEN	0x0 (в асинхронном режиме эффекта нет)
12	WREN	Как надо
11	WAITCFG	Всё равно
10	WRAPMOD	0x0
9	WAITPOL	Осмыслено только при бит 15 = 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	В соответствии с поддержкой памяти
5-4	MWID	Как надо
3-2	MTYP[1:0]	Как надо
1	MUXEN	0x0
0	MBKEN	0x1

Таблица 281. Поля FMC_BTRx

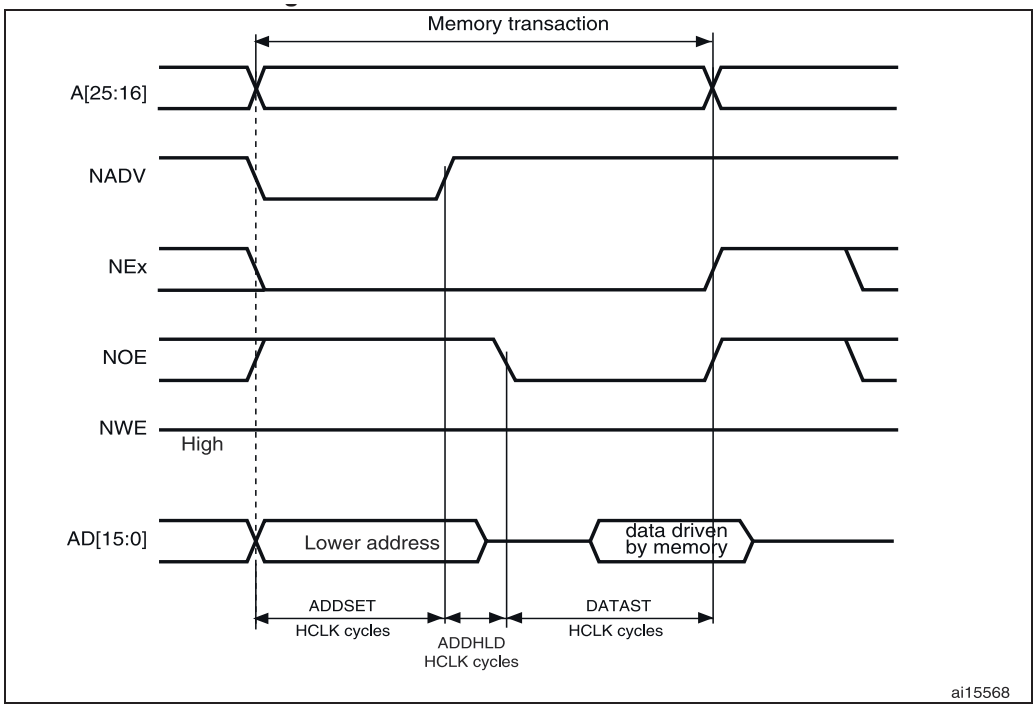
Номер бита	Имя его	Что писать
31:30	Резерв	0x0
29-28	ACCMOD	0x3
27-24	DATLAT	Всё равно
23-20	CLKDIV	Всё равно
19-16	BUSTURN	Время от высокого NEx до низкого NEx (BUSTURN HCLK)
15-8	DATAST	Время 2-й фазы доступа (DATAST такт HCLK для записи, DATAST тактов HCLK - чтение).
7-4	ADDHLD	Время средней фазы чтения (ADDHLD тактов HCLK)
3-0	ADDSET	Время 1-й фазы доступа (ADDSET тактов HCLK). Минимум ADDSET = 1.

Таблица 282. Поля FMC_BWTRx

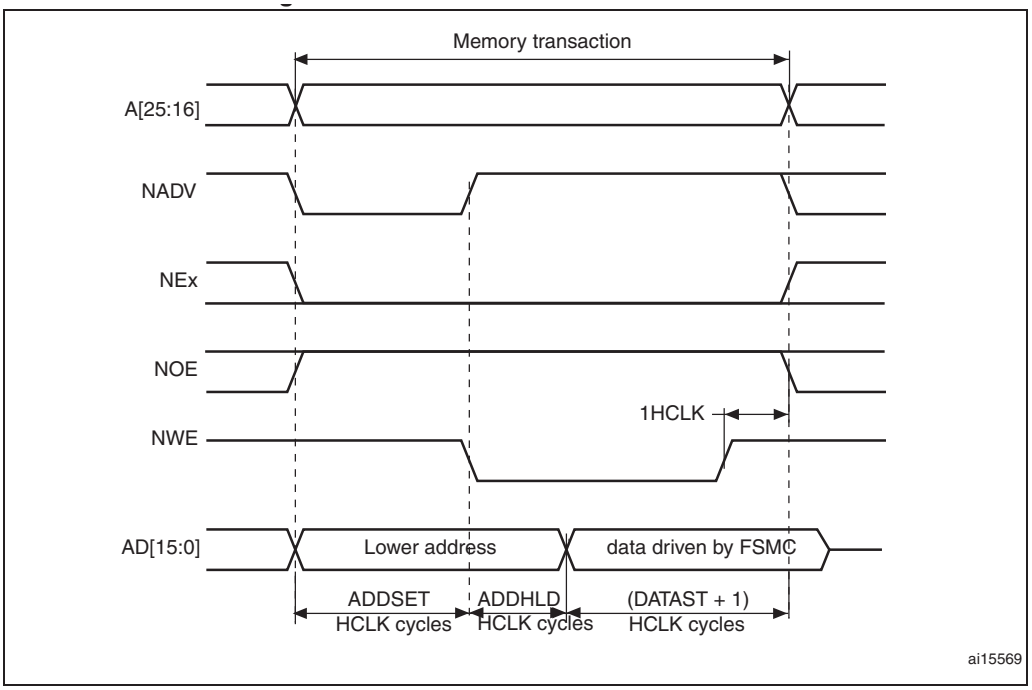
Номер бита	Имя его	Что писать
31:30	Резерв	0x0
29-28	ACCMOD	0x3
27-24	DATLAT	A-a-a, всё равно.
23-20	CLKDIV	A-a-a, всё равно.
19-16	BUSTURN	Время от высокого NEx до низкого NEx (BUSTURN HCLK)
15-8	DATAST	Время 2-й фазы доступа (DATAST такт HCLK для записи, DATAST тактов HCLK - чтение).
7-4	ADDHLD	Время средней фазы чтения (ADDHLD тактов HCLK)
3-0	ADDSET	Время 1-й фазы доступа (ADDSET тактов HCLK). Минимум ADDSET = 1.

Мультиплексный режим - асинхронный доступ к NOR Flash

Мультиплексное чтение



Мультиплексная запись



От режима D отличается выдачей младшего байта адреса на шину данных.

Таблица 283. Поля FMC_BCRx

Номер бита	Имя его	Что писать
31-21	Reserved	0x000
20	CCLKEN	Как надо
19	CBURSTRW	0x0 (в асинхронном режиме эффекта нет)
18:16	CPSIZE	0x0 (в асинхронном режиме эффекта нет)
15	ASYNCWAIT	1 - если память поддерживает. Иначе - 0.
14	EXTMOD	0x0
13	WAITEN	0x0 (в асинхронном режиме эффекта нет)
12	WREN	Как надо
11	WAITCFG	Всё равно
10	WRAPMOD	0x0
9	WAITPOL	Осмыслено только при бит 15 = 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	0x1
5-4	MWID	Как надо
3-2	MTYP[1:0]	0x02 (NOR Flash)
1	MUXEN	0x1
0	MBKEN	0x1

Таблица 284. Поля FMC_BTRx

Номер бита	Имя его	Что писать
31:30	Резерв	0x0
29-28	ACCMOD	0x0
27-24	DATLAT	Всё равно
23-20	CLKDIV	Всё равно
19-16	BUSTURN	Время от высокого NEx до низкого NEx (BUSTURN HCLK)
15-8	DATAST	Время 2-й фазы доступа (DATAST+1 такт HCLK для записи, DATAST тактов HCLK - чтение).
7-4	ADDHLD	Время средней фазы чтения (ADDHLD тактов HCLK)
3-0	ADDSET	Время 1-й фазы доступа (ADDSET тактов HCLK). Минимум ADDSET = 1.

Управление WAIT при асинхронном доступе

При выдаче памятью сигнала WAIT о своей неготовности ставят бит FMC_BCRx/ASYNCWAIT t.

При активном сигнале WAIT (высокий или низкий в зависимости от бита WAITPOL), вторая фаза доступа (Установки Данных), записанная в битах DATAST, расширяется до снятия сигнала WAIT. Фазы Установки адреса и Удержания адреса, записанные в бита ADDSET[3:0] и ADDHLD, на сигнал WAIT не реагируют.

Фазу Установки Данных надо писать так, чтобы обнаруживать сигнал WAIT за 4 такта HCLK до конца передачи памяти. Надо учесть:

1. Память выдаёт WAIT, выравненный с переключением NOE/NWE:

   DATAST ≥ (4 × HCLK) + max_wait_assertion_time

2. Память выдаёт WAIT, выравненный с NEx (или NOE/NWE не переключаются):

Если

max_wait_assertion_time > address_phase + hold_phase

DATAST ≥ (4 × HCLK) + (max_wait_assertion_time – address_phase – hold_phase)

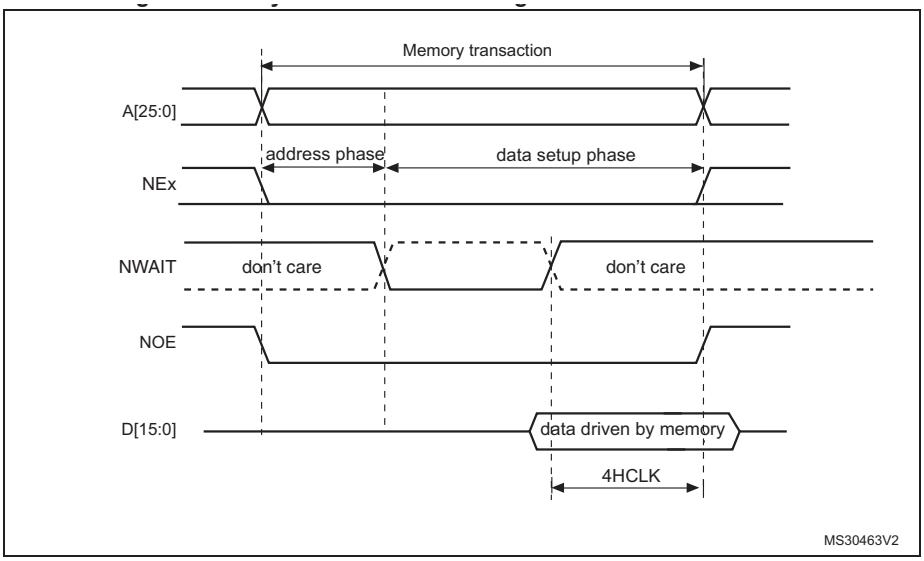
иначе

DATAST ≥ 4 × HCLK

где max_wait_assertion_time это максимальное время выставления WAIT при низких NEx/NOE/NWE.

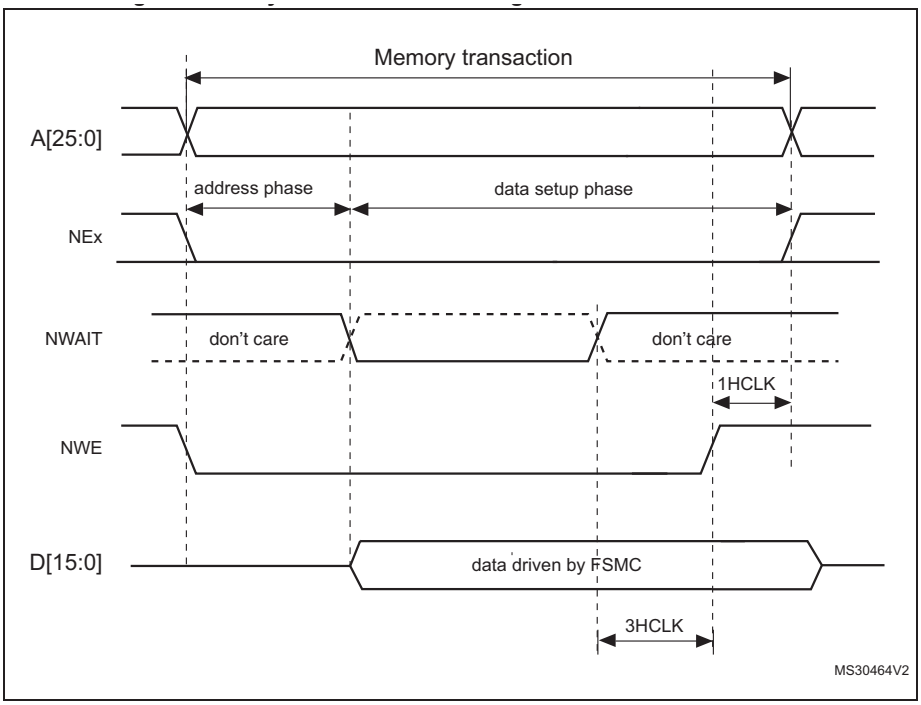
Ниже показано число тактов HCLK, добавляемых к доступу в память после снятия ею сигнала WAIT (независимо от описанных случаев).

Асинхронное ожидание при чтении



1. Полярность NWAIT зависит от бита FMC_BCRx/WAITPOL.

Асинхронное ожидание при записи



1. Полярность NWAIT зависит от бита FMC_BCRx/WAITPOL.

37.5.5. Синхронные передачи

Такты памяти FMC_CLK получают из HCLK по формуле:

FMC_CLK divider ratio = max(CLKDIV + 1,MWID(AHB data size))

Если `MWID` равен 16 или 8 бит, то делитель FMC_CLK всегда определяют в `CLKDIV`.

Если `MWID` равен 32 бит, то делитель FMC_CLK также зависит от ширины данных AHB.

Пример:

- При `CLKDIV`=1, `MWID`=32 бит, ширина AHB=8 бит, FMC_CLK=HCLK/4.
- При `CLKDIV`=1, `MWID`=16 бит, ширина AHB=8 бит, FMC_CLK=HCLK/2.

У NOR Flash определено минимальное время от выдачи NADV до высокого CLK. Поэтому во время первого цикла синхронного доступа (до выдачи NADV) FMC но подаёт памяти такта. Так передний фронт такта попадает но середину низкого NADV.

Задержки данных и памяти NOR

Задержка данных это число тактов ожидания перед считыванием данных. Значение DATLAT должно соответствовать задержке из регистра конфигурации NOR Flash. FMC не включает в число тактов задержки данных цикл при низком NADV.

Некоторые NOR Flash это учитывают, так что точное значение задержки может быть:

- NOR Flash latency = (DATLAT + 2) CLK clock cycles
- NOR Flash latency = (DATLAT + 3) CLK clock cycles

Некоторые современные устройства памяти в фазе задержки выставляют NWAIT. В этом случае DATLAT может быть минимальным. У памяти без NWAIT задержку надо соблюсти.

Передача одним пакетом

Если банк стоит в синхронном пакетном режиме, то FMC выполняет пакетную передачу с длиной, кратной ширине AHB и после строба последнего данного снимает Chip Select.

Переход границ страницы для Cellular RAM 1.5

Cellular RAM 1.5 не позволяет пакетного доступа при пересечении границ страниц (или страниц границ?). Контроллер FMC автоматически разбивает пакетный доступ на основании битов размера страницы FMC_BCR1/CPSIZE.

Управление ожиданием

У синхронной NOR Flash сигнал NWAIT смотрят после задержки в (DATLAT+2) тактов CLK.

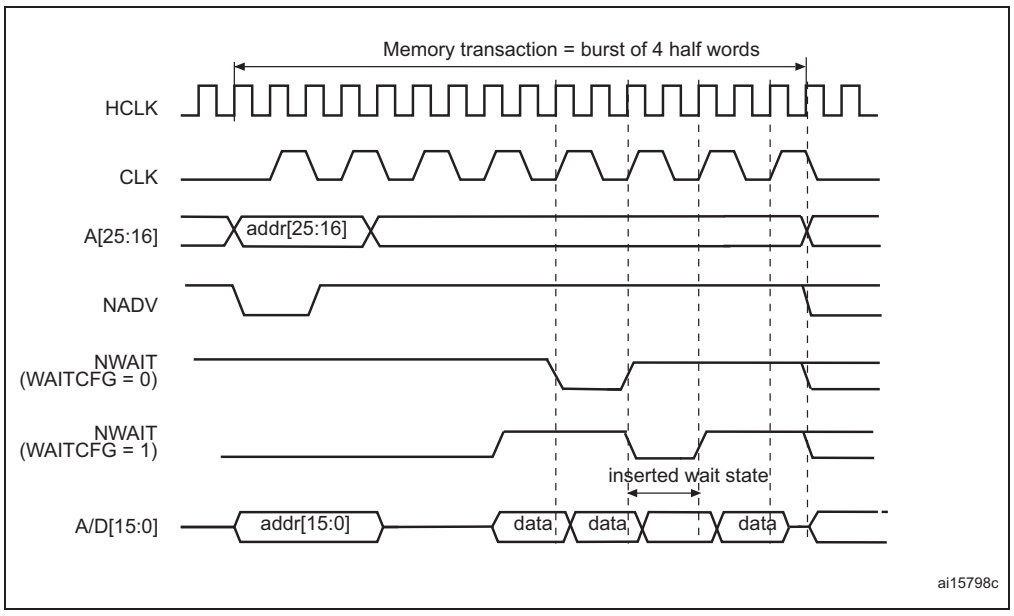
При активном NWAIT все ждут его снятия и читают данные сразу (бит WAITCFG = 1) или последующему фронту такта (бит WAITCFG = 0).

При активном NWAIT контроллер FMC также шлёт памяти такты, держит Chip Select и разрешение вывода. В пакетном режиме есть две конфигурации сигнала NWAIT:

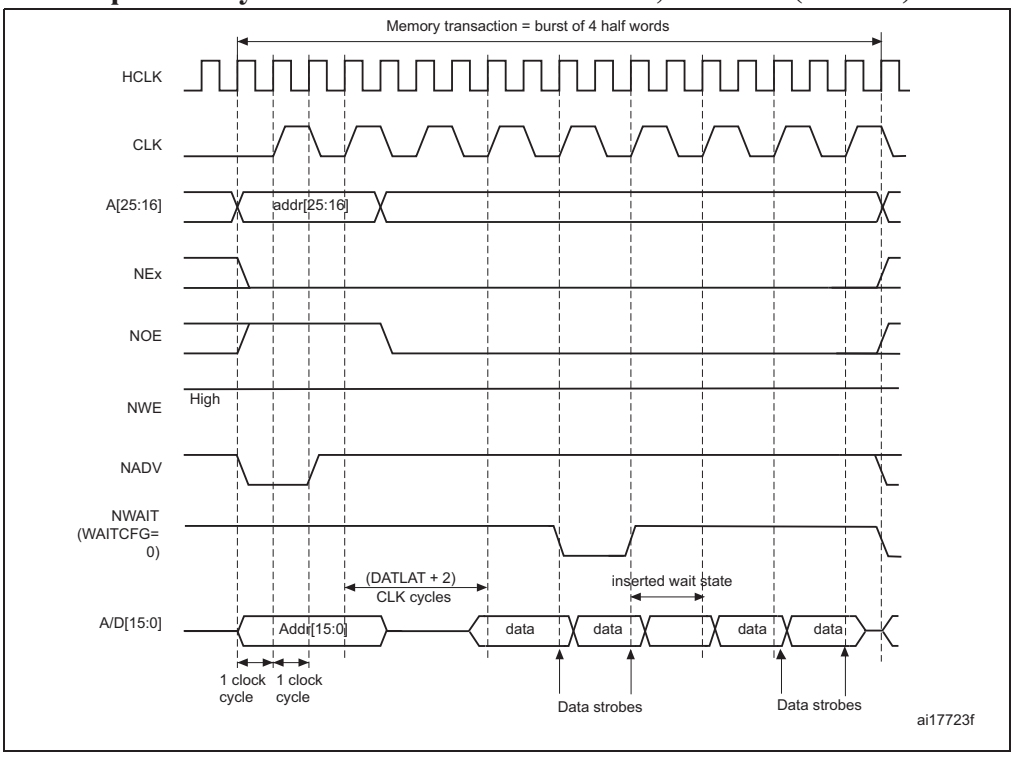
- Память ставит NWAIT за цикл данных до состояния ожидания (по умолчанию после сброса).
- NWAIT встаёт во время ожидания

Сие задают битом FMC_BCRx/WAITCFG.

Конигурация ожидания



Синхронное мультиплексное чтение - NOR, PSRAM (CRAM)



1. Линии байта BL не показаны; NOR доступа они высокие

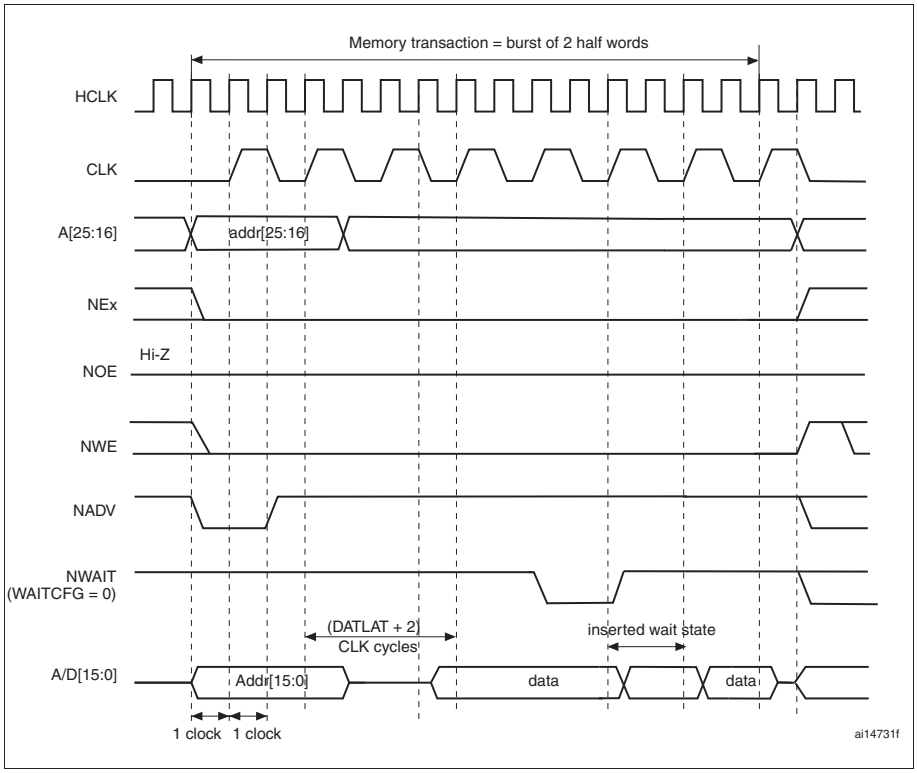
Таблица 285. Поля FMC_BCRx

Номер бита	Имя его	Что писать
31-21	Reserved	0x000
20	CCLKEN	Как надо
19	CBURSTRW	0x0 (в асинхронном режиме эффекта нет)
18:16	CPSIZE	Как надо (0x1 для CRAM 1.5)
15	ASYNCWAIT	0x0
14	EXTMOD	0x0
13	WAITEN	0x1 если память поддерживает, иначе 0
12	WREN	В синхронном режиме эффекта нет
11	WAITCFG	Соответственно памяти
10	WRAPMOD	0x0
9	WAITPOL	Соответственно памяти
8	BURSTEN	0x1
7	Reserved	0x1
6	FACCEN	Соответственно памяти
5-4	MWID	Как надо
3-2	MTYP[1:0]	0x01 или 0x02
1	MUXEN	Как надо
0	MBKEN	0x1

Таблица 286. Поля FMC_BTRx

Номер бита	Имя его	Что писать
31:30	Резерв	0x0
29-28	ACCMOD	0x0
27-24	DATLAT	Задержка данных
23-20	CLKDIV	Делитель для HCLK
19-16	BUSTURN	Время от высокого NEx до низкого NEx (BUSTURN HCLK)
15-8	DATAST	Всё равно
7-4	ADDHLD	Всё равно
3-0	ADDSET	Всё равно

Синхронная мультиплексная запись - PSRAM (CRAM)



1. Память должна ставить NWAIT за такт вперёд и WAITCFG b= 0.
2. Выходы байтов (NBL) не показаны, их держат низкими при активном NEx.

Таблица 285. Поля FMC_BCRx

Номер бита	Имя его	Что писать
31-21	Reserved	0x000
20	CCLKEN	Как надо
19	CBURSTRW	0x1
18:16	CPSIZE	Как надо (0x1 для CRAM 1.5)
15	ASYNCWAIT	0x0
14	EXTMOD	0x0
13	WAITEN	0x1 если память поддерживает, иначе 0
12	WREN	0x1
11	WAITCFG	0x0
10	WRAPMOD	0x0
9	WAITPOL	Соответственно памяти
8	BURSTEN	В синхронной записи эффекта нет
7	Reserved	0x1
6	FACCEN	Соответственно памяти
5-4	MWID	Как надо
3-2	MTYP[1:0]	0x01
1	MUXEN	Как надо
0	MBKEN	0x1

Таблица 286. Поля FMC_BTRx

Номер бита	Имя его	Что писать
31:30	Резерв	0x0
29-28	ACCMOD	0x0
27-24	DATLAT	Задержка данных
23-20	CLKDIV	Делитель для HCLK (0 - нельзя)
19-16	BUSTURN	Время от высокого NEx до низкого NEx (BUSTURN HCLK)
15-8	DATAST	Всё равно
7-4	ADDHLD	Всё равно
3-0	ADDSET	Всё равно

37.5.6. Регистры контроллера NOR/PSRAM

Регистры 1..4 управления SRAM/NOR-Flash (FMC_BCR1..4)

Смещение адреса: 8 * (x – 1), x = 1…4

По сбросу: 0x0000 30DB для Bank 1 и 0x0000 30D2 для Bank 2 - 4

31–21	20	19	18–16	15	14	13	12	11	10	9	8	7	6	5–4	3–2	1	0
Reserved	CCLKEN	CBURSTRW	CPSIZE[2:0]	ASCYCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	WRAPMOD	WAITPOL	BURSTEN	Reserved	FACCEN	MWID[1:0]	MTYP[1:0]	MUXEN	MBKEN
	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

— Биты 31:21 Резерв, не трогать

— Бит 20 **CCLKEN:** Разрешение постоянной выдачи FMC_CLK на память.

0: FMC_CLK выдаётся только при синхронном доступе.
Частота FMC_CLK задаётся в FMC_BCRx/CLKDIV.

1: FMC_CLK выдаётся постоянно после установки CCLKEN и Bank 1 в синхронном режиме.

NB: Биты CCLKEN регистров FMC_BCR2..4 не волнуют, включает только регистр FMC_BCR1.

NB: При CCLKEN=1 используется CLKDIV регистра FMC_BTR1. CLKDIV в FMC_BWTR1 не влияет.

NB: В синхронном режиме при CCLKEN=1 все подключённые синхронные банки получают те же такты

— Бит 19 **CBURSTRW:** Разрешение синхронной пакетной записи в PSRAM (CRAM)

Синхронное чтение разрешает бит FMC_BCRx/BURSTEN.

0: Запись асинхронная

1: Запись синхронная.

— Биты 18:16 **CPSIZE[2:0]:** Размер страницы Cellular RAM 1.5.

При заданном размере контроллер FMC автоматически разбивает пакеты при достижении границ страниц.

000: Не разбивает

001: 128 байт

010: 256 байт

011: 512 байт

100: 1024 байта

Иное: Резерв

— Бит 15 **ASYNCWAIT:** Разрешение смотреть сигнал NWAIT для асинхронных передач

0: Да ну его, пусть гуляет

1: Учитываем NWAIT

— Бит 14 **EXTMOD:** Разрешение расширенного режима.

Так FMC берёт времянку не-мультиплексированной асинхронной записи из регистра FMC_BWTR, отдельно от чтения.

0: FMC_BWTR не нужен

1: FMC_BWTR используется

NB: При расширенном режиме FMC использует режимы:

- Mode 1 для SRAM/PSRAM (MTYP[1:0] =0x0 или 0x01)
- Mode 2 для NOR (MTYP[1:0] = 0x10).

— Бит 13 **WAITEN:** Разрешение ожидания по NWAIT в синхронном режиме.

0: Не надо

1: Надо использовать

— Бит 12 **WREN:** Разрешение записи в память.

0: Нельзя, ошибка AHB,

1: Можно.

— Бит 11 **WAITCFG:** Конфигурация активного сигнала NWAIT.

0: NWAIT активен за такт до состояния ожидания,

1: NWAIT активен во время состояния ожидания (не для PSRAM).

— Бит 10 **WRAPMOD:** Кольцевой пакетный доступ.

Контроллер может разбивать кольцевой пакет AHB на два линейных.

Только в пакетном режиме

0: Нельзя,
1: Можно.

NB: Бесполезный бит: ни CPU ни DMA кольцевых пакетов не создают.

— Бит 9 **WAITPOL:** Активный уровень сигнала NWAIT.
0: Низкий,
1: Высокий.

— Бит 8 **BURSTEN:** Разрешение пакетного синхронного чтения.
0: Нельзя. Читают асинхронно.
1:Можно.

— Бит 7 Резерв, не трогать

— Бит 6 **FACCEN:** Разрешение доступа к NOR Flash.
0: Нельзя
1: Можно

— Биты 5:4 **MWID[1:0]:** Ширина шины данных памяти.
00: 8 бит,
01: 16 бит,
10: 32 бита,
11: Резерв, не ставить.

— Биты 3:2 **MTYP[1:0]:** Тип памяти.
00: SRAM (после сброса для Bank 2…4)
01: PSRAM (CRAM)
10: NOR Flash/OneNAND Flash (после сброса для Bank 1)
11: Резерв

— Бит 1 **MUXEN:** Разрешение мультиплекса адреса/данных для NOR PSRAM:
0: Нельзя
1: Можно на шине данных

— Бит 0 **MBKEN:** Разрешение банка памяти.
После сброса Bank 1 включён, обращение к выключенному банку даёт ошибку AHB.
0: Выкл,
1: Вкл.

Регистры 1..4 времянки SRAM/NOR-Flash (FMC_BTR1..4)

Смещение адреса: 0x04+8 * (x – 1), x = 1…4

По сбросу: 0x0FFF FFFF

Добавляют задержку в конце операций чтения/записи. Это позволяет сравнивать минимальное время между последовательными операциями (t_{EHEL} от высокого NEx до низкого FMC_NEx) и максимальное время, нужное памяти для освобождения шины данных после чтения (t_{EHQZ}).

В расширенном режиме это времянка чтения, времянку записи пишут в регистр FMC_BWTRx.

31 30	29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4	3 2 1 0
Reserved	ACCMOD[1:0]	DATLAT[3:0]	CLKDIV3:0]	BUSTURN3:0]	DATAST7:0]	ADDHLD3:0]	ADDSET[3:0]
	rw rw	rw rw rw rw	rw rw rw rw	rw rw rw rw	rw rw rw rw rw rw rw rw	rw rw rw rw	rw rw rw rw

— Биты 31:30 Резерв, не трогать

— Биты 29:28 **ACCMOD[1:0]:** Режим асинхронного доступа
Работают только при FMC_BCRx/EXTMOD = 1.
00: Режим A
01: Режим B
10: Режим C
11: Режим D

— Биты 27:24 **DATLAT[3:0]:** Задержка данных для синхронной памяти
Это число тактов FMC_CLK+2 для памяти в синхронном пакетном режиме перед чтением/записью первого данного. В асинхронном режиме не работает.

— Биты 23:20 **CLKDIV[3:0]:** Делитель тактов для сигнала FMC_CLK из HCLK

0000: Reserved

0001: FMC_CLK period = 2 × HCLK periods

0010: FMC_CLK period = 3 × HCLK periods

1111: FMC_CLK period = 16 × HCLK periods (default value after reset)

В асинхронных NOR Flash, SRAM или PSRAM не работает.

— Биты 19:16 **BUSTURN[3:0]:** Длительность фазы переключения шины

Добавляют задержку (0-15 тактов HCLK) в конце передач чтение/запись. Это позволяет сравнивать минимальное время между последовательными операциями (t_{EHEL} от высокого NEx до низкого FMC_NEx) и максимальное время, нужное памяти для освобождения шины данных после чтения (t_{EHQZ}). Вставляется между асинхронным чтением/записью (мультиплекс или режим D) и другим асинхронным чтением/записью статического банка. При чтении банки могут быть разными, при записи в мультиплексе или режиме D это тот же банк.

Иногда, независимо от значения BUSTURN, время переключения шины фиксировано:

- Задержка не вставляется между двумя последовательными записями в один статический банк кроме мультиплекса и режима D.
- Задержка равна 1 такту FMC между:
  - Двумя последовательными асинхронными чтениями из одного статического банка кроме мультиплекса и режима D.
  - Асинхронным чтением и асинхронной или синхронной записью в любой статический или динамический банк кроме мультиплекса и режима D.
  - Асинхронным чтением (режимы 1, 2, A, B и C) и чтением другого статического банка.
- Задержка равна 2 тактам FMC между:
  - Двумя последовательными синхронными записями в тот же банк.
  - Синхронной записью и асинхронным чтением/записью статического банка (при чтении банк может быть другим.
  - Двумя последовательными синхронными чтениями с последующим чтением/записью другого статического банка.
- Задержка равна 3 тактам FMC между:
  - Двумя последовательными синхронными записями в разные статические банки.
  - Синхронной записью и синхронным чтением любого банка.

— Биты 15:8 **DATAST[7:0]:** Длительность фазы данных асинхронного доступа

— Биты 7:4 **ADDHLD[3:0]:** Длительность фазы удержания адреса (такты HCLK)

NB: В синхронном режиме она всегда равна 1 такту памяти.

— Биты 3:0 **ADDSET[3:0]:** Длительность фазы установки адреса (такты HCLK)

Годится для SRAM, ROM и асинхронных NOR Flash и PSRAM.

NB: В синхронном режиме не нужна. В мультиплексе и режиме D минимум ADDSET = 1.

NB: PSRAM (CRAM) имеют переменную задержку, они выдают сигнал NWAIT.
У них обязательно нужно DATLAT=0.

Регистры 1..4 времянки записи SRAM/NOR-Flash (FMC_BWTR1..4)

Смещение адреса: 0x104+8 * (x – 1), x = 1…4

По сбросу: 0x0FFF FFFF

Используется только в расширенном режиме при FMC_BCRx/EXTMOD = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		ACCMOD[1:0]		Reserved								BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
		rw	rw									rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:30 Резерв, не трогать

— Биты 29:28 **ACCMOD[1:0]:** Режим асинхронного доступа

Работают только при FMC_BCRx/EXTMOD = 1.

00: Режим A
01: Режим B
10: Режим C
11: Режим D

— Биты 27:20 Резерв, не трогать

— Биты 19:16 **BUSTURN[3:0]:** Длительность фазы переключения шины

Добавляют задержку (0-15 тактов HCLK) в конце передач чтение/запись. Вставляется между асинхронной записью и другим асинхронным чтением/записью статического банка. При чтении банки могут быть разными, при записи в мультиплексе или режиме D это тот же банк.

Иногда, независимо от значения BUSTURN, время переключения шины фиксировано:

- Задержка не вставляется между двумя последовательными записями в один статический банк кроме мультиплекса и режима D.
- Задержка равна 2 тактам FMC между:
  - Двумя последовательными синхронными записями в тот же банк.
  - Синхронной записью и асинхронным чтением/записью статического банка.
- Задержка равна 3 тактам FMC между:
  - Двумя последовательными синхронными записями в разные статические банки.
  - Синхронной записью и синхронным чтением любого банка.

— Биты 15:8 **DATAST[7:0]:** Длительность фазы данных асинхронного мультиплексного доступа

— Биты 7:4 **ADDHLD[3:0]:** Длительность фазы удержания адреса (такты HCLK)

NB: В синхронном режиме она всегда равна 1 такту памяти.

— Биты 3:0 **ADDSET[3:0]:** Длительность фазы установки адреса асинхронного досткпа (такты HCLK)

NB: В синхронном режиме не нужна. В мультиплексе и режиме D минимум ADDSET = 1.

37.6. Контроллер NAND Flash/PC Card

FMC задаёт времянки для:

- 8- и 16-бит NAND Flash
- 16-бит PC Card совместимых устройств

Контроллер NAND Flash/PC Card управляет тремя внешними банками, Bank 2, 3 и 4:

- Bank 2 и Bank 3 поддерживают NAND Flash
- Bank 4 поддерживает PC Card.

Банки конфигурируют через свои регистры.

Таблица 289. Программируемые параметры доступа NAND Flash/PC Card

Параметр	Функция	Доступ	Такты	Min.	Max.
Установки	Чисто тактов HCLK установки адреса перед выдачей команды	R/W	HCLK	1	256
Ожидание	Минимум тактов HCLK выдачи команды	R/W	HCLK	2	255
Удержание	Чисто тактов HCLK удержания адреса (и данных при записи) после снятия команды	R/W	HCLK	1	254
Hi-Z шины данных	Чисто тактов HCLK удержания шины данных в Hi-Z после начала записи	W	HCLK	1	255

37.6.1. Сигналы интерфейса внешней памяти

Префикс “N” показывает сигналы, активные низким уровнем.

8-бит NAND Flash память

Таблица 290. 8-бит NAND Flash

Сигнал FMC	I/O	Функция
A[17]	O	Разрешение защёлки адреса NAND Flash (ALE)
A[16]	O	Разрешение защёлки команды NAND Flash (CLE)
D[7:0]	I/O	8-бит мультиплекс. двунаправленная шина адреса/данных
NCE[x]	O	Chip Select, x = 2, 3
NOE(= NRE)	O	Разрешение выхода (в памяти: разрешение входа, NRE)
NWE	O	Разрешение записи
NWAIT/INT[3:2]	I	Готовность NAND Flash

16-бит NAND Flash память

Таблица 291. 16-бит NAND Flash

Сигнал FMC	I/O	Функция
A[17]	O	Разрешение защёлки адреса NAND Flash (ALE)
A[16]	O	Разрешение защёлки команды NAND Flash (CLE)
D[15:0]	I/O	8-бит мультиплекс. двунаправленная шина адреса/данных
NCE[x]	O	Chip Select, x = 2, 3
NOE(= NRE)	O	Разрешение выхода (в памяти: разрешение входа, NRE)
NWE	O	Разрешение записи
NWAIT/INT[3:2]	I	Готовность NAND Flash

Таблица 292. 16-бит PC Card

Сигнал FMC	I/O	Функция
A[10:0]	O	Шина адреса
NIORD	O	Разрешение выхода пространства I/O
NIOWR	O	Разрешение записи пространства I/O
NREG	O	Доступ к пространству Атрибутов или Общему (O/A)
D[15:0]	I/O	Двунаправленная шина данных
NCE4_1	O	Chip Select 1
NCE4_2	O	Chip Select 2 (доступ 16- или 8-бит)
NOE	O	Разрешение выхода пространства O/A
NWE	O	Разрешение записи пространства O/A
NWAIT	I	PC Card ждёт сигнала от FMC (в памяти IORDY)
INTR	I	Прерывание от PC Card (только если может)
CD	I	Присутствие PC Card. Активен высоким. При низком - ошибка AHB.

37.6.2. Поддерживаемая память и передачи NAND Flash / PC Card

Неподдерживаемые передачи выделены красным.

Таблица 293. Поддерживаемая память и передачи

Уст-во	Режим	R/W	Ширина AHB	Ширина данных	Можно?	Коммент
NAND 8-bit	Асинхр.	R	8	8	Y	-
	Асинхр.	W	8	8	Y	-
	Асинхр.	R	16	8	Y	Два обращения FMC
	Асинхр.	W	16	8	Y	Два обращения FMC
	Асинхр.	R	32	8	Y	Два обращения FMC
	Асинхр.	W	32	8	Y	Два обращения FMC
NAND 16-bit	Асинхр.	R	8	16	Y	-
	Асинхр.	W	8	16	N	-
	Асинхр.	R	16	16	Y	-
	Асинхр.	W	16	16	Y	-
	Асинхр.	R	32	16	Y	Два обращения FMC
	Асинхр.	W	32	16	Y	Два обращения FMC

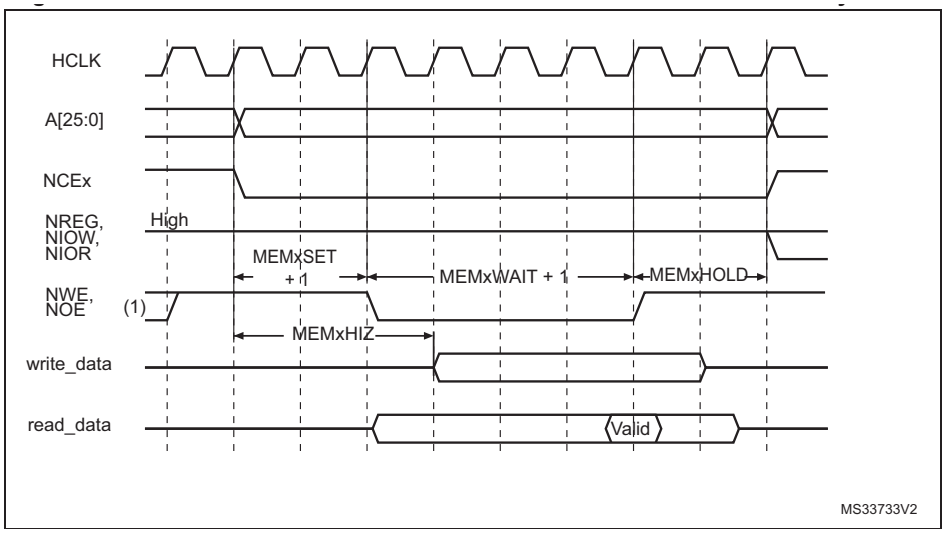
37.6.3. Времянки NAND Flash / PC Card

PC Card/CompactFlash и NAND Flash управляются через регистры:

- Управления: FMC_PCRx
- Статуса прерываний: FMC_SRx
- ECC: FMC_ECCRx
- Времянки для Общей памяти: FMC_PMEMx
- Времянки для памяти Атрибутов: FMC_PATTx
- Времянки для пространства I/O: FMC_PIOx

Три параметра регистра конфигурации задают число тактов HCLK для трёх фаз доступа к PC Card/CompactFlash или NAND Flash плюс один параметр начальной времянки шины данных при записи. Времянки доступа к всем пространствам: Общему, Атрибутов и I/O (только для PC Card) аналогичны.

Времянка доступа к Общему пространству NAND Flash/PC Card



1. При записи NOE стоит высоким (неактивным). NWE - при чтении.
2. Задержка удержания при записи равна (MEMHOLD) x HCLK, при чтении - (MEMHOLD + 2) x HCLK.

37.6.4. Операции NAND Flash

Сигналы разрешения защёлок NAND Flash (CLE и ALE) это линии адреса контроллера FMC. То есть CPU просто пишет передаваемую команду или адрес по особому адресу своей памяти.

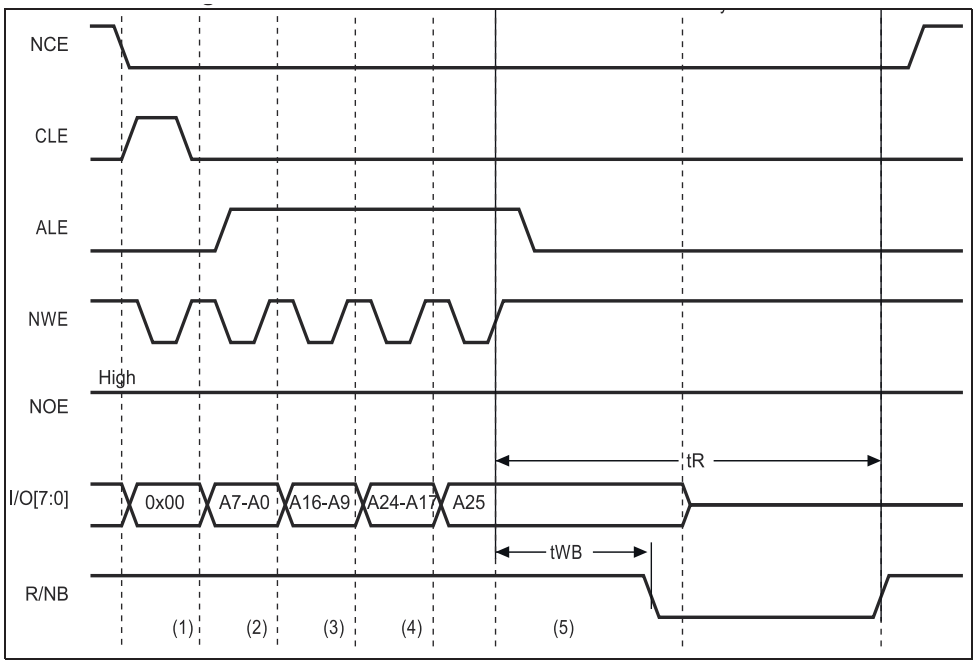
Типичное чтение страницы из NAND Flash требует следующих шагов:

1. В регистрах FMC_PCRx, FMC_PMEMx и, иногда FMC_PATTx конфигурируем и включает нужный банк памяти.
2. CPU пишет байт команды в общее пространство. Активный уровень CLE говорит, что это команда. При дальнейшем чтении страницы памяти её повторять не надо.
3. The CPU может послать адрес начала чтения (STARTAD) в четырёх или трёх байтах STARTAD[7:0], STARTAD[16:9], STARTAD[24:17] и в конце STARTAD[25] в общую память или пространство атрибутов. Активный уровень ALE говорит, что это адрес.
4. Контроллер ждёт сигнала готовности от NAND Flash (высокий R/NB), удерживая сигнал NCE активным (низким).
5. CPU может читать страницу NAND Flash чтением общего пространства (поле данных + парное поле) байт за байтом.
6. Следующую страницу NAND Flash можно читать даже без записи команд или адреса:
   - операцией пункта 5
   - задав новый адрес как в пункте 3
   - задав новую команду, начав с пункта 2

37.6.5. Предожидание NAND Flash

Некоторые устройства NAND Flash требуют ожидания сигнала готовности после записи последней части адреса.

Доступ к не ‘CE не волнует’ NAND-Flash



1. CPU записал байт 0x00 по адресу 0x7001 0000.
2. CPU записал байт A7~A0 по адресу 0x7002 0000.
3. CPU записал байт A16~A9 по адресу 0x7002 0000.
4. CPU записал байт A24~A17 по адресу 0x7002 0000.
5. CPU записал байт A25 по адресу 0x7802 0000: FMC пишет с времянкой из регистра FMC_PATT2, где ATTHOLD ≥ 7 (то есть (7+1) × HCLK = 112 ns > t_{WB} max). Это гарантирует низкий NCE до переключения R/NB в низкий и назад (что требует память NAND Flash, где NCE не волнует).

Такое дело задают через MEMHOLD для получения времянки t_{WB}. Чтение CPU из NAND Flash имеет задержку (MEMHOLD + 2) x HCLK тактов, а запись — (MEMHOLD) x HCLK тактов.

Во избежание этих ограничений можно использовать пространство с его времянкой из регистра ATTHOLD и держать MEMHOLD минимальным. И CPU должен использовать общее пространство для обмена с NAND Flash, кроме записи последнего байта адреса, его надо писать в память атрибутов.

37.6.6. Код коррекции ошибок (ECC) в NAND Flash

Тут есть два одинаковых блока вычисления ECC, по одному на банк памяти 2 и 3.

Алгоритм вычисления ECC основан на кодах Хэмминга и позволяет исправлять 1 и обнаруживать 2 ошибки в битах при обмене с NAND Flash памятью по 256, 512, 1 024, 2 048, 4 096 и 8 192 байт.

Модули ECC отслеживают шину данных и сигналы (NCE и NWE) работающего банка памяти.

ECC работает так:

- При доступе к банкам 2 и 3 NAND Flash данные на шине D[15:0] считываются и идут в ECC.
- При обращении по другим адресам (команды и атрибуты) данные не запоминаются.

После передачи желанного числа байтов из регистров FMC_ECCR2/3 можно извлечь вычисленное значение ECC и очистить регистр снятием бита включения ECCEN. Вычисления начинаются после установки бита ECCEN в регистрах FMC_EPCR2/3.

Вычисляем ECC:

1. Ставим бит ECCEN регистра FMC_EPCR2/3.
2. Пишем данные в страницу NAND Flash. ECC вычисляется.
3. Читаем ECC из регистра FMC_ECCR2/3 и запоминаем его.
4. Снимаем и снова ставим бит ECCEN в FMC_EPCR2/3. Читаем записанный блок данных.
5. Читаем новый ECC и сравниваем со старым. Теперь мы знаем, есть ли ошибка.

37.6.7. Операции PC Card/CompactFlash

Адресные пространства и доступ к памяти

FMC поддерживает у CompactFlash и PC Card режимы Памяти и I/O (True IDE не поддерживает).

Их сделали из 3 пространств памяти: Общего, Атрибутов и I/O.

Ножки nCE2 и nCE1 (FMC_NCE4_2 FMC_NCE4_1) выбирают по одному байту с шины D15-0. Какой именно байт выбрать указывает линия A0. Всё слово выбирается при низких nCE2 и nCE1.

Пространство памяти выбирается сигналом nOE для чтения или nWE для записи при участии низких nCE2/nCE1 и nREG.

- При nREG=1 выбирается Общая память
- При nREG=0 выбирается память Атрибутов

Пространство I/O выбирают сигналами nIORD для чтения или nIOWR для записи [nOE/nWE для пространства памяти] в сочетании с nCE2/nCE1 и участием nREG для пространства I/O.

Есть три типа доступа к 16-бит PC Card:

- К Обшей памяти для данных: 8-бит по чётным адресам или 16-бит доступ AHB.
  8-бит доступ по нечётным адресам невозможен. 32-бит запрос AHB это два 16-бит обращения.
- К пространству Атрибутов с конфигурацией PC Card: 8-бит по чётным адресам.
  16-бит доступ AHB это одна 8-бит передача: nCE1 будет низкой, nCE2 - высокой и только чётный байт D7-D0 достоверен.
  32-бит доступ AHB это две 8-бит передачи по чётным адресам.
- К пространству I/O допустим 8-бит или 16 бит доступ AHB.

Таблица 294. Сигналы и типы доступа к 16-бит PC-Card

nCE2	nCE1	nREG	nOE/nWE	nIORD	A10	A9	A7-1	A0	Space	Access type	Allowed/not Allowed
1	0	1	0	1	X	X	X-X	X	Common Memory Space	Read/Write byte on D7-D0	YES
0	1	1	0	1	X	X	X-X	X		Read/Write byte on D15-D8	Not supported
0	0	1	0	1	X	X	X-X	0		Read/Write word on D15-D0	YES
X	0	0	0	1	0	1	X-X	0	Attribute Space	Read or Write Configuration Registers	YES
X	0	0	0	1	0	0	X-X	0		Read or Write CIS (Card Information Structure)	YES
1	0	0	0	1	X	X	X-X	1	Attribute Space	Invalid Read or Write (odd address)	YES
0	1	0	0	1	X	X	X-X	x		Invalid Read or Write (odd address)	YES
1	0	0	1	0	X	X	X-X	0	I/O space	Read Even Byte on D7-0	YES
1	0	0	1	0	X	X	X-X	1		Read Odd Byte on D7-0	YES
1	0	0	1	0	X	X	X-X	0		Write Even Byte on D7-0	YES
1	0	0	1	0	X	X	X-X	1		Write Odd Byte on D7-0	YES
0	0	0	1	0	X	X	X-X	0		Read Word on D15-0	YES
0	0	0	1	0	X	X	X-X	0		Write word on D15-0	YES
0	1	0	1	0	X	X	X-X	X		Read Odd Byte on D15-8	Not supported
0	1	0	1	0	X	X	X-X	X		Write Odd Byte on D15-8	Not supported

Ожидание

При стоящем бите разрешения FMC_PCRx/PWAITEN фазы доступа можно расширить на число из битов MEMWAITx/ATTWAITx/IOWAITx, выдавая сигнал nWAIT после nOE/nWE или nIORD/nIOWR. Эти числа находят так:

$$xxWAITx \geq 4 + \frac{max\_wait\_assertion\_time}{HCLK}$$

где **max_wait_assertion_time** это длительность низкого NWAIT при низком nOE/nWE или nIORD/nIOWR.

После снятия WAIT, FMC расширяет фазу WAIT на 4 такта HCLK.

37.6.8. Регистры контроллера NAND Flash/PC Card

Регистры 2..4 управления NAND Flash/PC Card (FMC_PCR2..4)

Смещение адреса: 0x40 + 0x20 * (x – 1), x = 2…4

По сбросу: 0x0000 0018

31–20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	ECCPS[2:0]			TAR[3:0]				TCLR[3:0]				Reserved		ECCEN	PWID[1:0]		PTYP	PBKEN	PWAITEN	Reserved
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	

— Биты 31:20 Резерв, не трогать

— Биты 19:17 **ECCPS[2:0]:** Размер страницы ECC.

000: 256 байт
001: 512 байт
010: 1024 байта
011: 2048 байт
100: 4096 байт
101: 8192 байта

— Биты 16:13 **TAR[3:0]:** Задержка от ALE до RE в тактах HCLK.

Время: t_ar = (TAR + SET + 2) × THCLK где THCLK это период HCLK

0000: 1 HCLK cycle (default)
1111: 16 HCLK cycles

NB: SET это MEMSET или ATTSET в соответствии с пространством.

— Биты 12:9 **TCLR[3:0]:** Задержка от CLE до RE в тактах HCLK.

Время: t_clr = (TCLR + SET + 2) × THCLK где THCLK это период HCLK

0000: 1 HCLK cycle (default)
1111: 16 HCLK cycles

NB: SET это MEMSET или ATTSET в соответствии с пространством.

— Биты 8:7 Резерв, не трогать

— Бит 6 **ECCEN:** Включение ECC

0: Выкл и сброс (default after reset),
1: Вкл.

— Биты 5:4 **PWID[1:0]:** Ширина шины данных.

00: 8 бит
01: 16 бит (default after reset). Обязательно для PC Card.
10: Резерв
11: Резерв

— Бит 3 **PTYP:** Тип памяти.

0: PC Card, CompactFlash, CF+ or PCMCIA
1: NAND Flash (default after reset)

— Бит 2 **PBKEN:** Включение банка PCCard/NANDFlash.

Доступ к выключенному банку даёт ошибку шины AHB

0: Выкл. (default after reset)
1: Вкл.

— Бит 1 **PWAITEN:** Разрешение ожидания.

0: disabled
1: enabled

NB: Для PC Card при включённом ожидании MEMWAITx/ATTWAITx/IOWAITx надо ставить так:
xxWAITx ≥ 4 + max_wait_assertion_time/HCLK
где **max_wait_assertion_time** это длительность NWAIT при низких nOE/nWE или nIORD/nIOWR.

— Бит 0 Резерв, не трогать

Регистры 2..4 статуса FIFO и прерываний (FMC_SR2..4)

Смещение адреса: 0x44 + 0x20 * (x – 1), x = 2…4

По сбросу: 0x0000 0040

При записи FMC использует FIFO длиной 16 слов данных от AHB. Для чтения верного ECC надо дождаться опустения FIFO.

31 … 7	6	5	4	3	2	1	0
Reserved	FEMPT	IFEN	ILEN	IREN	IFS	ILS	IRS
	r	rw	rw	rw	rw	rw	rw

Значение битов:

0: Нельзя или Не было

1: Можно или Было

— Биты 31:7 Резерв, не трогать

— Бит 6 **FEMPT:** FIFO пуст.

0: FIFO не пуст

1: FIFO пуст

— Бит 5 **IFEN:** Разрешение прерывания обнаружения заднего фронта

— Бит 4 **ILEN:** Разрешение прерывания обнаружения высокого уровня

— Бит 3 **IREN:** Разрешение прерывания обнаружения переднего фронта

— Бит 2 **IFS:** Статус прерывания обнаружения заднего фронта

— Бит 1 **ILS:** Статус прерывания обнаружения высокого уровня

— Бит 0 **IRS:** Статус прерывания обнаружения переднего фронта

Регистры 2..4 времянки Общей памяти (FMC_PMEM2..4)

Смещение адреса: 0x48 + 0x20 * (x – 1), x = 2…4

По сбросу: 0xFCFC FCFC

31 … 24	23 … 16	15 … 8	7 … 0
MEMHIZ[7:0]	MEMHOLD[7:0]	MEMWAIT[7:0]	MEMSET[7:0]
rw … rw	rw … rw	rw … rw	rw … rw

— Биты 31:24 **MEMHIZ[7:0]:** Время Hi-Z шины данных Общей памяти после начала записи в HCLK

0000 0000: 1 HCLK

1111 1110: 255 HCLK

1111 1111: Резерв.

— Биты 23:16 **MEMHOLD[7:0]:** Время удержания Общей памяти

При чтении это число тактов (HCLK+2) удержания адреса после снятия команды (NWE, NOE).

При записи это число тактов HCLK удержания данных после снятия команды (NWE, NOE).

0000 0000: Резерв

0000 0001: 1 HCLK при записи, 3 HCLK при чтении

1111 1110: 254 HCLK при записи, 256 HCLK при чтении

1111 1111: Резерв.

— Биты 15:8 **MEMWAIT[7:0]:** Время ожидания Общей памяти

Это число тактов HCLK (+1) удержания команды (NWE, NOE) при чтении и записи. Время может расширяться если по его истечению стоит сигнал NWAIT.

0000 0000: Резерв

0000 0001: 2 HCLK (+ 1 такт после снятия NWAIT)

1111 1110: 255 HCLK (+ 1 такт после снятия NWAIT)

1111 1111: Резерв

— Биты 7:0 **MEMSET[7:0]:** Время установки адреса Общей памяти

Это число тактов HCLK (+1) установки адреса перед подачей команды (NWE, NOE):

0000 0000: 1 HCLK

1111 1110: 255 HCLK

1111 1111: Резерв.

Регистры 2..4 времянки памяти Атрибутов (FMC_PATT2..4)

Смещение адреса: 0x4C + 0x20 * (x – 1), x = 2…4

По сбросу: 0xFCFC FCFC

Используется при 8-бит доступе к памяти Атрибутов или записи последнего адреса, если это время должно отличаться от предыдущих обращений.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ATTHIZ[7:0]								ATTHOLD[7:0]								ATTWAIT[7:0]								ATTSET[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:24 **ATTHIZ[7:0]:** Время Hi-Z шины данных памяти Атрибутов после начала записи в HCLK

0000 0000: 0 HCLK

1111 1110: 255 HCLK

1111 1111: Резерв.

— Биты 23:16 **ATTHOLD[7:0]:** Время удержания памяти Атрибутов

При чтении это число тактов (HCLK+2) удержания адреса после снятия команды (NWE, NOE).

При записи это число тактов HCLK удержания данных после снятия команды (NWE, NOE).

0000 0000: Резерв

0000 0001: 1 HCLK при записи, 3 HCLK при чтении

1111 1110: 254 HCLK при записи, 256 HCLK при чтении

1111 1111: Резерв.

— Биты 15:8 **ATTWAIT[7:0]:** Время ожидания памяти Атрибутов

Это число тактов HCLK (+1) удержания команды (NWE, NOE) при чтении и записи. Время может расширяться если по его истечению стоит сигнал NWAIT.

0000 0000: Резерв

0000 0001: 2 HCLK (+ 1 такт после снятия NWAIT)

1111 1110: 255 HCLK (+ 1 такт после снятия NWAIT)

1111 1111: Резерв

— Биты 7:0 **ATTSET[7:0]:** Время установки адреса памяти Атрибутов

Это число тактов HCLK (+1) установки адреса перед подачей команды (NWE, NOE):

0000 0000: 1 HCLK

1111 1110: 255 HCLK

1111 1111: Резерв.

Регистр 4 времянки пространства I/O (FMC_PIO4)

Смещение адреса: 0xB0

По сбросу: 0xFCFC FCFC

Используется при доступе к пространству I/O 16-бит PC Card/CompactFlash.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IOHIZ[7:0]								IOHOLD[7:0]								IOWAIT[7:0]								IOSET[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:24 **IOTHIZ[7:0]:** Время Hi-Z шины данных пространства I/O после начала записи в HCLK

0000 0000: 0 HCLK

1111 1110: 255 HCLK

1111 1111: Резерв.

— Биты 23:16 **IOHOLD[7:0]:** Время удержания пространства I/O

При чтении это число тактов HCLK удержания адреса (и данных при записи) после снятия команды (NWE, NOE).

0000 0000: Резерв

0000 0001: 1 HCLK

1111 1111: 255 HCLK

— Биты 15:8 **IOWAIT[7:0]:** Время ожидания пространства I/O

Это число тактов HCLK (+1) удержания команды (SMNWE, SMNOE) при чтении и записи. Время может расширяться если по его истечению стоит сигнал NWAIT.

0000 0000: Резерв

0000 0001: 2 HCLK (+ 1 такт после снятия NWAIT)
1111 1111: 255 HCLK (+ 1 такт после снятия NWAIT)

— Биты 7:0 **IOSET[7:0]:** Время установки адреса пространства I/O

Это число тактов HCLK (+1) установки адреса перед подачей команды (NWE, NOE):
0000 0000: 1 HCLK
1111 1111: 256 HCLK

Регистры 2/3 результата ECC (FMC_ECCR2/3)

Смещение адреса: 0x54 + 0x20 * (x – 1), x = 2…3

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ECC[31:0]																															
r																															

— Биты 31:24 **ECC[31:0]:** Результат вычисления ECC

Таблица 295. Релевантные биты ECC

ECCPS[2:0]	Размер страницы в байтах	Биты ECC
0	256	ECC[21:0]
1	512	ECC[23:0]
10	1024	ECC[25:0]
11	2048	ECC[27:0]
100	4096	ECC[29:0]
101	8192	ECC[31:0]

37.7. Контроллер SDRAM

37.7.1. Основные свойства контроллера SDRAM

Вот они:

- Два банка SDRAM с независимой конфигурацией
- Ширина шины 8, 16, 32 бита
- 13-бит Адрес Строки, 11-бит Адрес Колонки, 4 внутренних банка: 4x16Mx32bit (256 MB), 4x16Mx16bit (128 MB), 4x16Mx8bit (64 MB)
- Словный, полусловный, байтовый доступ
- Такты SDRAM могут быть HCLK/2 или HCLK/3
- Автоматическое управление границами строк и банков
- Мультибанковый пинг-понг
- Программируемая времянка
- Автоматическая регенерация с программируемой частотой
- Режим саморегенерации
- Режим выключения питания
- Программная инициализация SDRAM по включению питания
- Задержка CAS от 1,2,3
- Кешируемый Read FIFO глубиной 6 строк x 32-бита (6 x 14-бит тег адреса)

37.7.2. Сигналы интерфейса SDRAM

При запуске, ножки I/O SDRAM для связи с контроллером FMC SDRAM конфигурируют программно. Неиспользованные ножки можно пустить на другие цели.

Таблица 296. Сигналы SDRAM

Сигнал SDRAM	I/O	Описание	Альтерн. функция
SDCLK	O	Такты SDRAM	
SDCKE[1:0]	O	SDCKE0: Разрешение тактов SDRAM Bank 1 SDCKE1: Разрешение тактов SDRAM Bank 2	
SDNE[1:0]	O	SDNE0: Разрешение м/с SDRAM Bank 1 SDNE1: Разрешение м/с SDRAM Bank 2	
A[12:0]	O	Адрес	FMC_A[12:0]
D[31:0]	I/O	Двунаправленная шина данных	FMC_D[31:0]
BA[1:0]	O	Адрес банка	FMC_A[15:14]
NRAS	O	Строб адреса строки (RAS)	
NCAS	O	Строб адреса колонки (CAS)	
SDNWE	O	Разрешение записи	
NBL[3:0]	O	Маска выходного байта для записи (в памяти: DQM[3:0])	FMC_NBL[3:0]

37.7.3. Функциональное описание контроллера SDRAM

Все выходы контроллера SDRAM (сигналы, адрес и данные) изменяются по заднему фронту тактов памяти (FMC_SDCLK).

Инициализация SDRAM

Всё делается программой. При использовании двух банков их надо инициализировать одновременно, установив биты CTB1 и CTB2 в регистре FMC_SDCMR:

1. Пишем параметры устройства в регистр FMC_SDCRx. Частоту тактов SDRAM, RBURST и RPIPE пишем в регистр FMC_SDCR1.
2. Пишем времянку устройства в регистр FMC_SDTRx. TRP и TRC пишут в регистр FMC_SDTR1.
3. Ставим MODE = ‘001’ и битами CTB1 и/или CTB2 регистра FMC_SDCMR пускает такты на память (SDCKE идёт высоким).
4. Ждём предписанный период. Обычно это около 100 µs.
5. Ставим MODE = ‘010’ и битами CTB1 и/или CTB2 в FMC_SDCMR выдаём команду “Precharge All” ("Предзаряд всего").
6. Ставим MODE = ‘011’ и пишем биты CTB1 и/или CTB2 вместе с числом последовательных команд Авто-регенерации (NRFS) регистр FMC_SDCMR. Обычно их 8.
7. В поле MRD ставим ваше устройство SDRAM, пишем MODE= '100' и битами CTB1 и/или CTB2 в FMC_SDCMR выдаём команду "Load Mode Register" ("Загрузить регистр режима") для программирования SDRAM. В частности:
   a) Задержку CAS надо ставить по регистрам FMC_SDCR1/2
   b) Длину пакета (BL) = 1 надо выбрать битами M[2:0] = 000 в регистре режима. Если у каждого банка свой регистр режима, то операцию надо сделать дважды, выбирая нужный банк.
8. В регистре FMC_SDRTR задаём частоту регенерации (это задержка между двумя циклами).
9. У мобильных устройств SDRAM писать регистр расширенного режима надо после инициализации SDRAM: Сначала пустым чтением при BA1=1 и BA=0 выбираем регистр расширенного режима и пишем в него нужное значение.

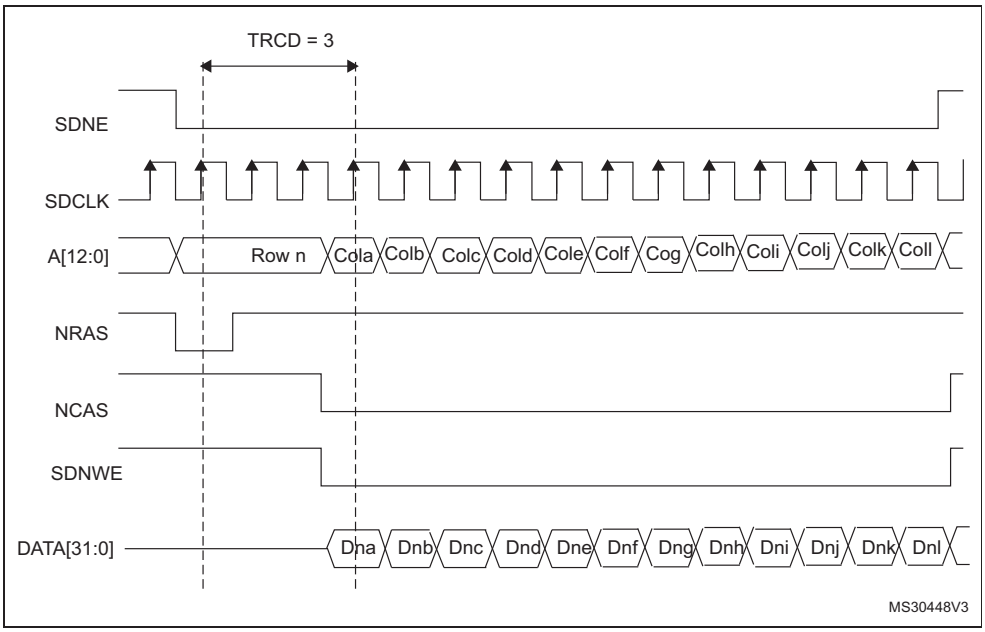
Теперь SDRAM готов принимать команды. Системный сброс может появиться когда шина данных занята устройством. Так что после сброса SDRAM надо снова инициализировать до обращения к контроллеру NOR Flash/PSRAM/SRAM или NAND Flash/PC Card.

NB: Если к FMC подключены два SDRAM, то одновременный доступ к ним делается по времянке Банка 1 (TMRD, TRAS и TXSR) регистра FMC_SDTR1.

Цикл записи контроллера SDRAM

Контроллер SDRAM транслирует все пакетные запросы записи в одинарные обращения к памяти. Он отслеживает активные строки каждого банка чтобы выполнять последовательную запись в каждый банк (Мультибанковый пинг-понг). Защиту записи выключают снятием бита WP в регистре FMC_SDCRx.

Пакетная запись в SDRAM



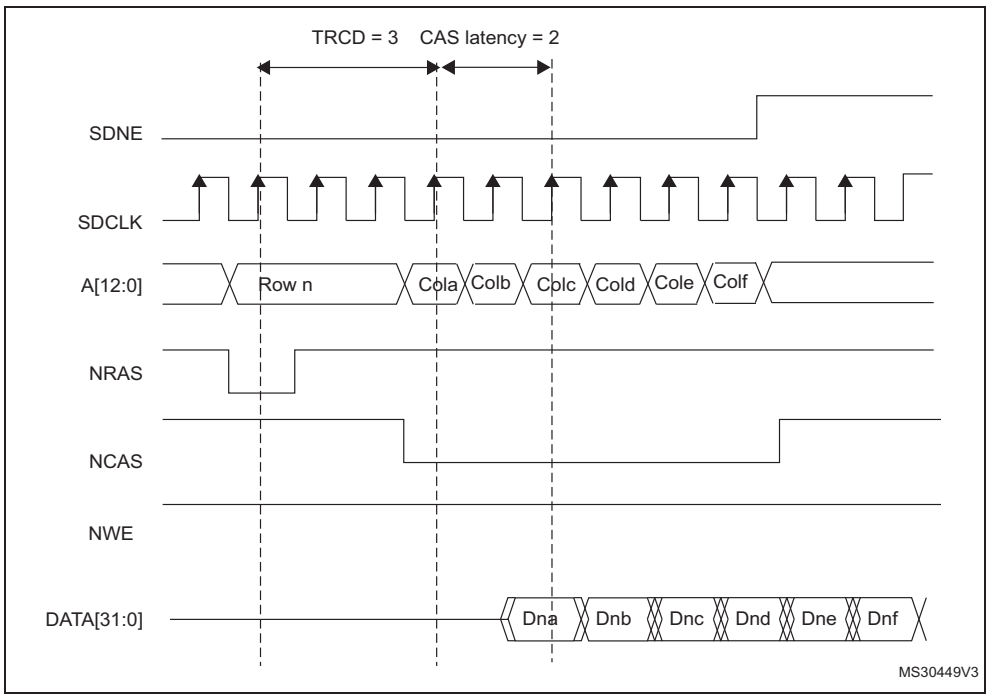
Контроллер SDRAM всегда проверяет следующий доступ.

- Если он в ту же строку или другую активную строку, то выполняется запись,
- Если он в другую строку, то контроллер выдаёт команду предзаряда, активирует новую строку и запускает команду записи.

Цикл чтения контроллера SDRAM

Контроллер SDRAM транслирует все пакетные запросы чтения в одинарные обращения к памяти. Он отслеживает активные строки каждого банка чтобы выполнять последовательное чтение из каждого банка (Мультибанковый пинг-понг).

Пакетное чтение SDRAM



У контроллера FMC SDRAM есть кешируемый FIFO чтения (6 строк x 32 бита). Туда заранее читают данные во время задержек CAS и RPIPE. Вот формула:

Число ранних данных = задержка CAS + 1 + (задержка RPIPE)/2

The `FMC_SDCR1/RBURST` должен стоять.

Пример:

- Задержка CAS = 3, задержка RPIPE = 0: в FIFO идут 4 данных.
- Задержка CAS = 3, задержка RPIPE = 2: в FIFO идут 5 данных.

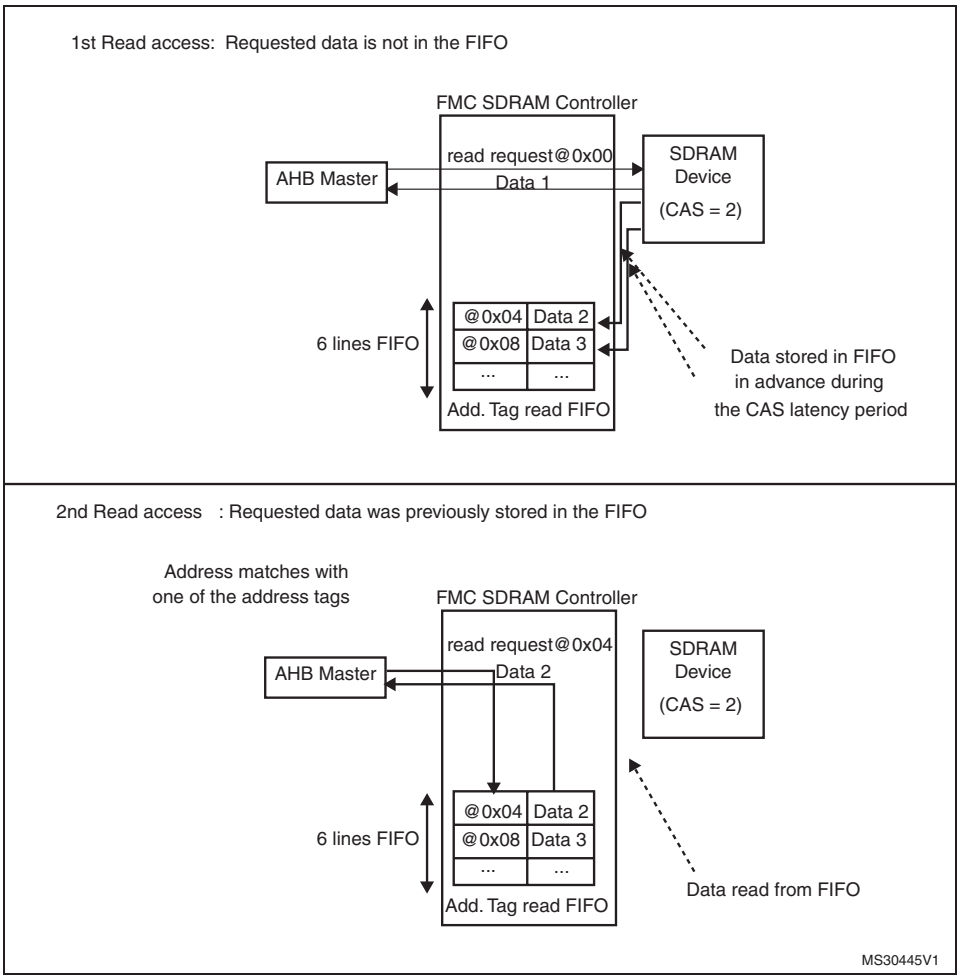
У FIFO чтения есть 14-бит тег адреса: 11 бит адреса колонки, 2 бита выбора внутреннего банка и активной строки, и 1 бит устройства SDRAM

Если при предварительном чтении пакетной передачи AHB достигнут конец строки, то такие данные в FIFO не попадают. При одиночном чтении всё поступает в FIFO замечательно.

При появлении каждого запроса чтения контроллер SDRAM смотрит:

- Если адрес совпадает с адресом из тега, то данные читаются из FIFO и соответствующий адресный тег/строка очищаются и остаток строк в FIFO сжимается дабы не было пустот.
- Иначе, памяти выдаётся команда чтения и новое данное идёт в FIFO. Если он полон, то более старое данное теряется.

Чтение со стоящим RBURST (CAS=2, RPIPE=0)



Во время записи или команды Предзаряда FIFO чтения сливается и ждёт новых данных.

Если при первом запросе чтения текущий адрес не достиг границы строки, то контроллер SDRAM заранее читает по следующему адресу памяти во время задержек CAS и RPIPE (если есть)..

Надо соблюсти условия:

• Бит FMC_SDCR1/RBURST должен стоять.

Управление адресом зависит от следующего запроса AHB:

- Он последовательный (пакет AHB)
  Контроллер SDRAM инкрементирует адрес.
- Он не последовательный
  - Если читают из той же или другой активной строки, то новый адрес передаётся памяти и ведущий останавливается на период задержки CAS, ожидая нового данного из памяти.
  - Иначе, контроллер SDRAM выдаёт команду предзаряда, активирует новую строку и задаёт команду чтения.

Если RBURST снят, то FIFO не используется.

Пересечение границы строки и банка

Если при последовательном чтении/записи адрес обращения достигает границы, то контроллер SDRAM выполняет:

1. Предзаряд активной строки,
2. Активацию новой строку
3. Старт команды чтения/записи.

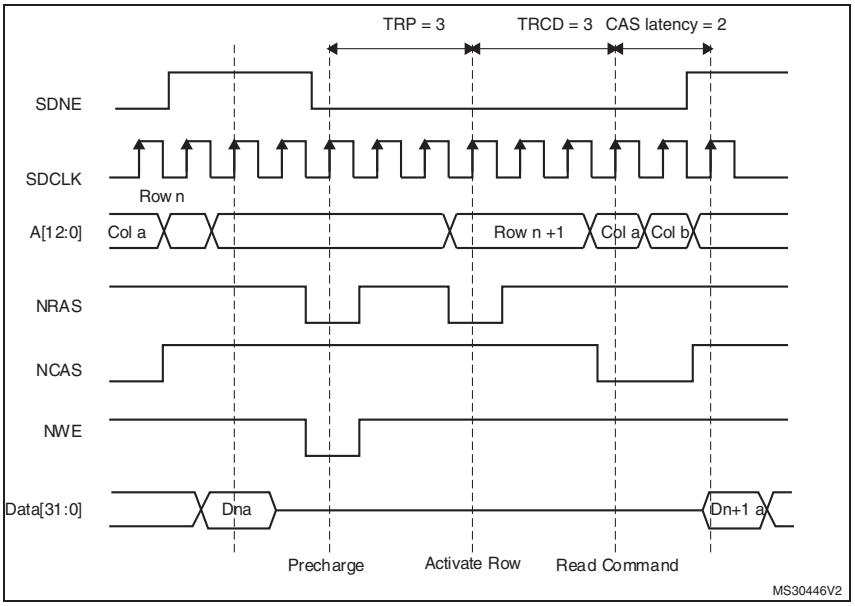
На границе строк автоматическая активация следующей строки поддерживается для всех колонок и ширины шины данных.

Если надо, то контроллер SDRAM вставляет дополнительные такты между следующими командами:

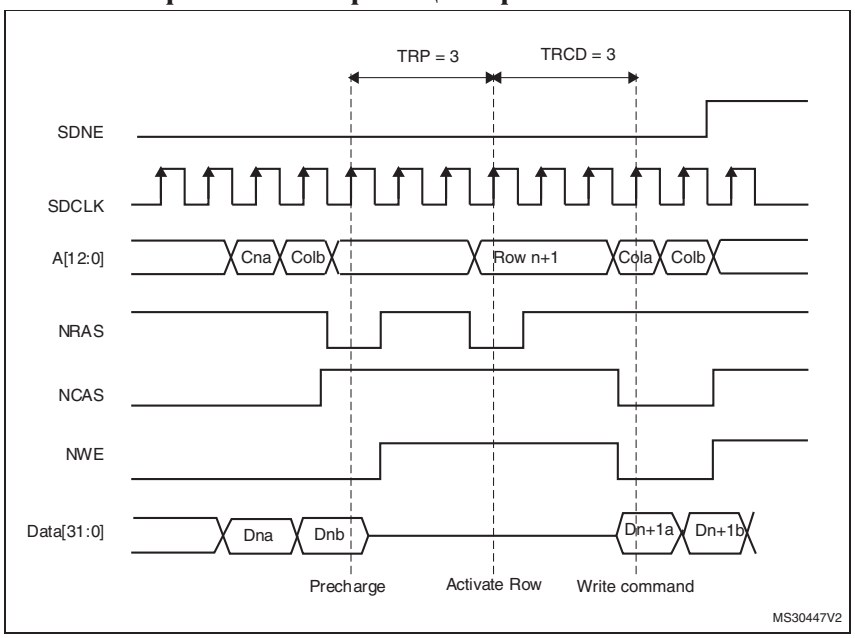
- Между командами Предзаряда и Активизации для соблюдения параметра TRP (только при обращении к другой строке того же банка),
- Между командами Активизации и Чтения для соблюдения параметра TRCD.

Их определяют в регистре FMC_SDTRx.

Чтение с пересечением границы строки



Запись с пересечением границы строки



Если следующее последовательное обращение пересекает границу банка, то контроллер SDRAM активирует первую строку следующего банка и пускает новую команду чтения/записи. Бывает так:

- Если текущий банк не последний, то новый банк надо предзарядить. На границе банка автоматическая активация следующей строки поддерживается для всех колонок и ширины шины данных.
- При 13-бит адресе строки, 11-бит адресе колонки, 4 внутренних банках и 32-бит шине памяти, если текущий банк последний и устройство SDRAM подключено к Банку 1, то контроллер SDRAM продолжает чтение/запись из второго устройства SDRAM (просто надеясь, что оно уже инициализировано):
  a) Контроллер SDRAM активирует первую строку (после предзаряда активной строки если уже есть активная строка в первом внутреннем банке) и пускает команду нового чтения/записи.
  b) Если первая строка уже активирована, то контроллер SDRAM просто отдаёт команду нового чтения/записи..
- На границе последнего банка активация следующей строки поддерживается только при 13-бит адресе строки, 11-бит адресе колонки, 4 внутренних банках и 32-бит шине памяти SDRAM. Иначе, это нарушение диапазона адресов SDRAM вызовет ошибку AHB.

Цикл регенерации контроллера SDRAM

Контроллер SDRAM периодически выдаёт памяти команду Регенерации (Auto-refresh). Интервал выдачи импульсов регенерации в тактах памяти пишут в биты FMC_SDRTR/COUNT.

При работающем доступе к памяти запрос регенерации задерживается. Но при одновременном появлении запросов доступа и регенерации, последний важнее.

Запрос доступа при работающей регенерации запоминается и выполняется после завершения регенерации.

Появление нового запроса регенерации при не завершённом предыдущем цикле вызовет ошибку регенерации (бит RE в регистре состояния). Прерывание по нему разрешается битом REIE = ‘1’.

Если линии SDRAM не в состоянии простоя (не все строки закрыты), то контроллер SDRAM перед регенерацией выдаёт команду PALL (Предзаряд Всего).

Перед выдачей команды регенерации через регистр FMC_SDCMR (MODE = '011’) надо выдавать команду PALL (MODE =’ 010’).

37.7.4. Экономные режимы

Их два:

- Само-регенерация
  Устройство SDRAM само всё делает без внешнего тактирования.
- Выключенного питания
  Циклы регенерации запускает контроллер SDRAM.

Само-регенерация

Включается битами MODE = ‘101’ и выбором банка (CTB1 и/или CTB2) в регистре FMC_SDCMR.

Такты SDRAM и внутренний таймер регенерации останавливаются после задержки TRAS если:

- Команда Регенерации выдана на оба устройства, или
- Одно устройство не активировано (банк SDRAM не инициализирован).

Перед входом в режим регенерации контроллер SDRAM автоматически выдаёт команду PALL.

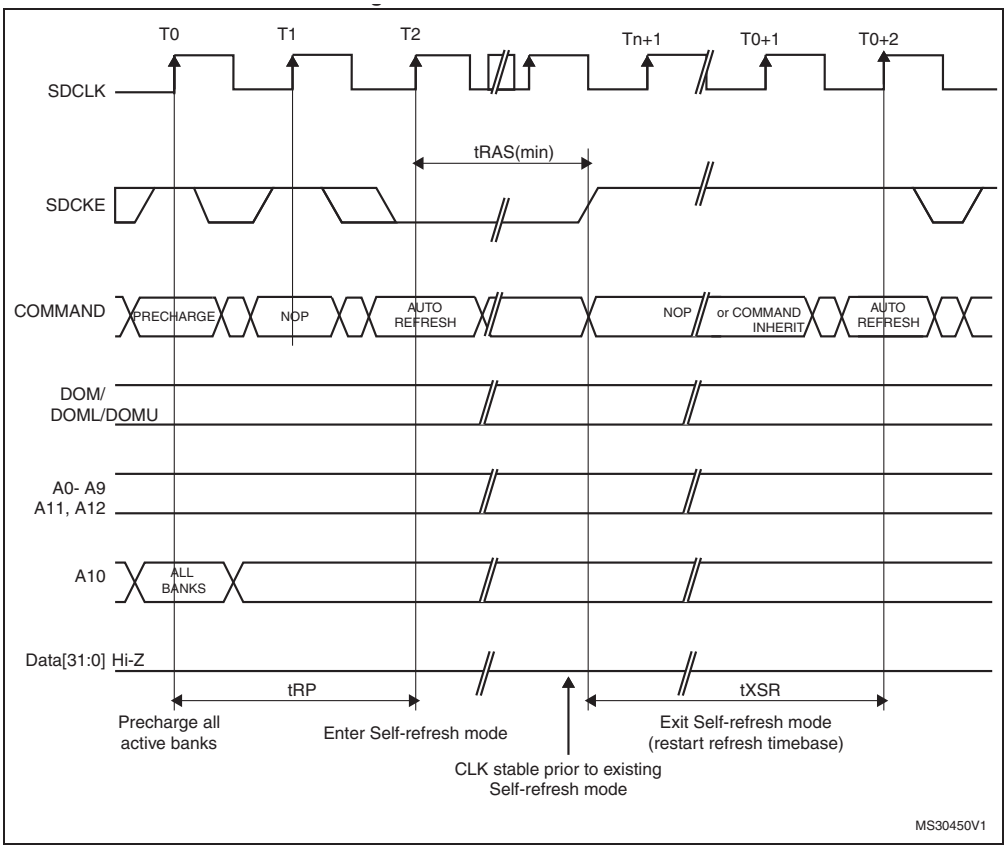
Если FIFO записи не пуст, то перед включением регенерации все данные посылаются в память, но флаг BUSY остаётся стоять до конца регенерации на время TRAS.

При регенерации все входы устройства SDRAM уходят в "Всё Равно", но SDCKE будет низким.

После выбора устройства SDRAM контролер выдаёт последовательность команд выхода из режима. После доступа к памяти выбранное устройство остается в Нормальном режиме.

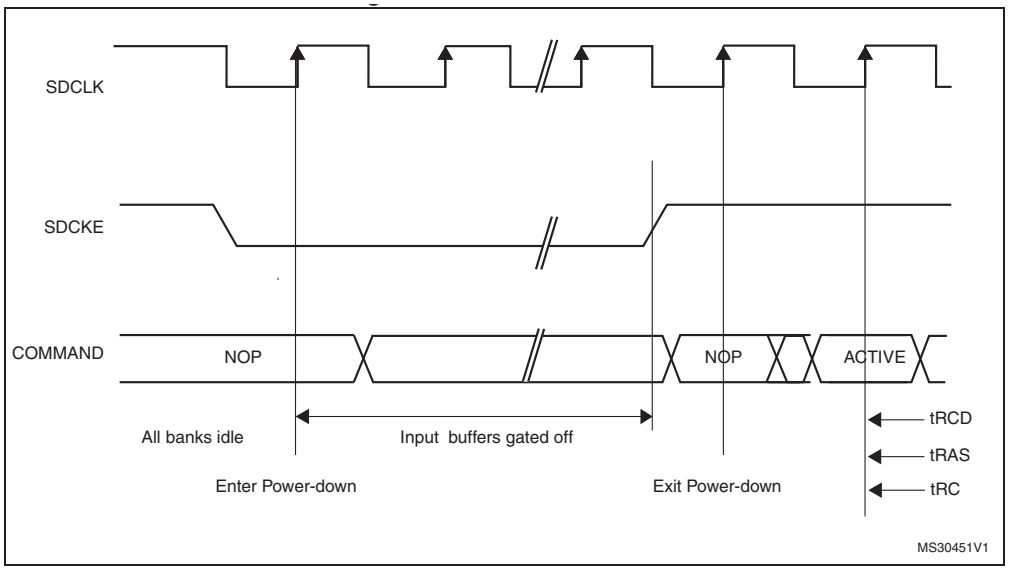
Выходят из режима Регенерации установкой битов MODE = 000 и выбора банков CTB1 и/или CTB2 в регистре FMC_SDCMR.

Режим Само-регенерации



Режим выключенного питания

Включается битами `MODE` = ‘110’ и выбором банка (`CTB1` и/или `CTB2`) в регистре `FMC_SDCMR`.



Если FIFO записи не пуст, то перед включением режима все данные посылаются в память.

После выбора устройства SDRAM контролер выдаёт последовательность команд выхода из режима. После доступа к памяти выбранное устройство остается в Нормальном режиме.

В этом режиме все входы устройства SDRAM уходят в "Всё Равно", но SDCKE будет низким.

Режим не может длиться больше периода регенерации и само-регенерация не делается. Контроллер делает её так:

1. Выход из выключения и установка высокого SDCKE
2. Выдача команды PALL если была активная строка
3. Выдача команды Само-регенерации
4. Установка низкого SDCKE для возврата в режим Выключения.

Выходят из режима Выключения установкой битов `MODE` = 000 и выбора банков `CTB1` и/или `CTB2` в регистре `FMC_SDCMR`.

37.7.5. Регистры контроллера SDRAM

Регистры управления 1,2 (FMC_SDCR1,2)

Смещение адреса: 0x140 + 4 * (x – 1), x = 1…2

По сбросу: 0x0000 02D0

31–15	14–13	12	11–10	9	8–7	6	5–4	3–2	1–0
Reserved	RPIPE[1:0]	RBURST.	SDCLK[1:0]	WP	CAS[1:0]	NB	MWID[1:0]	NR[1:0]	NC[1:0]
	rw rw	rw	rw rw	rw	rw rw	rw	rw rw	rw rw	rw rw

— Биты 31:15 Резерв, не трогать

— Биты 14:13 **RPIPE[1:0]:** Конвейер чтения

Это число тактов HCLK задержки чтения после задержки CAS.

00: нету

01: 1 такт HCLK

10: 2 такта HCLK

11: Резерв

NB: У соответствующих битов регистра FMC_SDCR2 только чтение.

— Бит 12 **RBURST:** Разрешение пакетного чтения

Контроллер SDRAM заранее читает следующие данные в FIFO чтения во время задержки CAS.

0: Одинарный запрос остаётся одинарным

1: Одинарный запрос становится как бы пакетным

NB: В соответствующих битах регистра FMC_SDCR2 ерунда.

— Биты 11:10 **SDCLK[1:0]:** Длина тактов SDCLK в тактах HCLK

Для обоих банков SDRAM. Для смены частоты такты надо выключать и после неё инициализировать.

00: Выкл.

01: Резерв

10: 2 x HCLK

11: 3 x HCLK

NB: У соответствующих битов регистра FMC_SDCR2 только чтение.

— Бит 9 **WP:** Защита записи банка

0: Пишите на здоровье.

1: Защищено.

— Биты 8:7 **CAS[1:0]:** Задержка CAS в тактах памяти

00: Резерв.

01: 1 такт

10: 2 такта

11: 3 такта

— Бит 6 **NB:** Число внутренних банков.

0: 2 банка.

1: 4 банка.

— Биты 5:4 **MWID[1:0]:** Ширина шины данных

00: 8 бит.

01: 16 бит

10: 32 бита

11: Резерв

— Биты 3:2 **NR[1:0]:** Число битов адреса строки

00: 11 бит.

01: 12 бит

10: 13 бита

11: Резерв

— Биты 1:0 **NC[1:0]:** Число битов адреса колонки

00: 8 бит.

01: 9 бит

10: 10 бит

11: 11 бит

NB: Перед изменением RBURST или RPIPE и выключением тактов SDCLK надо послать команду PALL для завершения текущих операций.

Регистры времянок 1,2 (FMC_SDTR1,2)

Смещение адреса: 0x148 + 4 * (x – 1), x = 1…2

По сбросу: 0x0FFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				TRCD[3:0]				TRP[3:0				TWR[3:0				TRC[3:0				TRAS[3:0				TXSR[3:0				TMRD[3:0			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

— Биты 31:28 Резерв, не трогать

— Биты 27:24 **TRCD[3:0]:** Задержка Строка-Колонка в тактах памяти

Задаёт задержку N+1 тактов памяти между командой Активации и командами Чтения/Записи.

— Биты 23:20 **TRP[3:0]:** Задержка предзаряда строки перед другой командой

Задаёт задержку N+1 тактов памяти между командой Предзаряда и другими командами. Надо использовать времянку самого медленного из подключённых устройств.

NB: Соответствующие биты регистра FMC_SDCR2 не волнуют.

— Биты 19:16 **TWR[3:0]:** Задержка восстановления

Задаёт задержку N+1 тактов памяти между командой Записи и командой Предзаряда.

NB: TWR должна годиться для характеристик подключённой памяти (t_{WR}), чтобы гарантировать:

TWR ≥ TRAS - TRCD и TWR ≥TRC - TRCD - TRP

Пример: TRAS= 4 такта, TRCD= 2 такта. Отсюда, TWR >= 2 такта. TWR надо писать 0x1.

Для двух устройств SDRAM в регистры и FMC_SDTR1 и FMC_SDTR2 пишут самое медленное их двух.

— Биты 15:12 **TRC[3:0]:** Задержка цикла строки

Задаёт задержку N+1 тактов памяти между командой Регенерации и Активизации или Регенерации. Для двух устройств SDRAM в регистры и FMC_SDTR1 и FMC_SDTR2 пишут самое медленное их двух.

NB: TRC должна соответствовать TRC и TRFC подключённого.

NB: Соответствующие биты регистра FMC_SDCR2 не волнуют.

— Биты 11:8 **TRAS[3:0]:** Время само-регенерации

Задаёт минимальную задержку N+1 тактов памяти между циклами регенерации.

— Биты 7:4 **TXSR[3:0]:** Задержка выхода из само-регенерации

Задаёт задержку N+1 тактов памяти между концом команды Регенерации и выдачей команды Активизации.

— Биты 3:0 **TMRD[3:0]:** Задержка от LoadModeRegister до Active

Задаёт задержку N+1 тактов памяти между командой Загрузить Регистр Режима и командой Активизации или Регенерации

NB: При подключении двух устройств SDRAM при одновременном доступе по командам Загрузить Регистр Режима и Регенерации используется времянка Банка 1 (TMRD, TRAS и TXSR) в FMC_SDTR1.

NB: TRP и TRC пишут только в регистр FMC_SDTR1. Для двух устройств надо писать времянки самого медленного.

Регистр командного режима (FMC_SDCMR)

Смещение адреса: 0x150

По сбросу: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										MRD[11:0]														NRFS[3:0]				CTB1	CTB2	MODE[2:0]		
										rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	w	w	w	w

— Биты 31:22 Резерв, не трогать

— Биты 21:9 **MRD[11:0]:** Определение регистра режима

Отсюда в регистр режима пишут командой Load Mode Register.

— Биты 8:5 **NRFS[3:0]:** Число выдаваемых команд само-регенерации при MODE = ‘011’.
0000: 1
0001: 2
….
1110: 15
1111: Резерв

— Бит 4 **CTB1:** Выбор целевого банка 1

— Биты 21:9 **CTB2:** Выбор целевого банка 2

— Биты 21:9 **MODE[2:0]:** Выдаваемая целевому банку команда
000: Нормальный режим
001: Разрешение конфигурации тактов
010: PALL (“Предзаряд Всего”)
011: Авто-регенерация
100: Загрузить регистр
101: Само-регенерация command
110: Выключение питания
111: Резерв

NB: При выдаче команды должен стоять бит целевого банка или обоих (CBT1 и/или CBT2).

Регистр таймера регенерации (FMC_SDRTR)

Смещение адреса: 0x154

По сбросу: 0x0000 0000

Это число тактов SDCLK между циклами регенерации.

COUNT = (SDRAM refresh rate × SDRAM clock frequency) – 20
SDRAM refresh rate = SDRAM refresh period / Number of rows

Пример

SDRAM refresh rate = 64 ms / (8196rows) = 7.81µs

где 64 ms это период регенерации SDRAM.

7.81µs × 60MHz = 468.6

Частоту регенерации надо увеличить на 20 тактов SDRAM (как в примере выше) чтобы получить безопасную границу когда внутренний запрос обновления появляется после получения запроса чтения. Это соответствует значению COUNT = ‘0000111000000’ (448).

Это 13-бит поле грузится в таймер, тот декрементируется тактами SDRAM и по достижении нуля выдаёт импульс регенерации и перегружается значением COUNT. Оно должно быть не меньше 41 такта SDRAM. Если оно нулевое, то регенерации нет.

Таймер начинает считать после записи регистра FMC_SDRTR. Его нельзя писать повторно, нарушится частота.

При работающем доступе к памяти запрос Само-регенерации задерживается. Но при одновременном появлении запросов Само-регенерация главнее. Опоздавший запрос обмена ждёт конца регенерации.

Регистр общий для обоих Банков SDRAM.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																	REIE	COUNT[12:0]													CRE
																	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w

— Биты 31:15 Резерв, не трогать

— Бит 14 **REIE:** Разрешение прерываний

— Биты 13:1 **COUNT[12:0]:** Перегружаемое значение счётчика

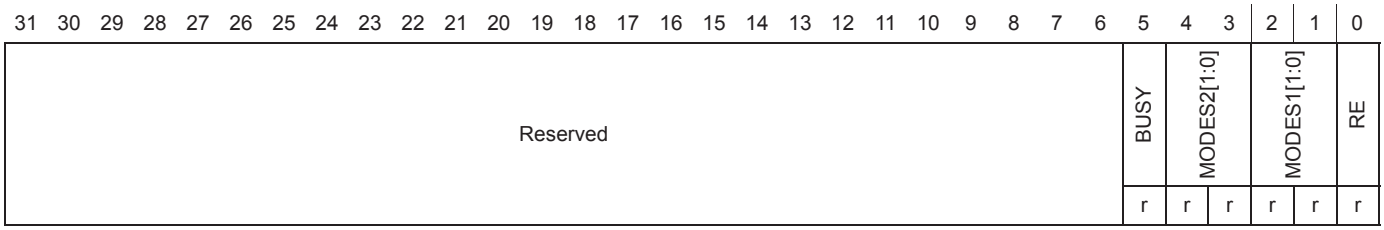
— Бит 0 **CRE:** Чистка флага ошибки регенерации

NB: Значение COUNT не должно быть равно сумме: TWR+TRP+TRC+TRCD+4 тактов памяти.

Регистр состояния (FMC_SDSR)

Смещение адреса: 0x158

По сбросу: 0x0000 0000



— Биты 31:6 Резерв, не трогать

— Бит 5 **BUSY:** Контроллер занят, обрабатывает команду

— Биты 4:3 **MODES2[1:0]:** Текущий режим Банка 2

00: Нормальный

01: Само-регенерация

10: Выключенный

— Биты 2:1 **MODES1[1:0]:** Текущий режим Банка 1

00: Нормальный

01: Само-регенерация

10: Выключенный

— Бит 0 **RE:** Флаг ошибки регенерации

37.8. Карта регистров FMC

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	FMC_BCR1	Reserved											CCLKEN	CBURSTRW	CPSIZE[2:0]			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	WRAPMOD	WAITPOL	BURSTEN	Reserved	FACCEN	MWID[1:0]		MTYP[1:0]		MUXEN	MBKEN
0x08	FMC_BCR2	Reserved												CBURSTRW	CPSIZE[2:0]			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	WRAPMOD	WAITPOL	BURSTEN	Reserved	FACCEN	MWID[1:0]		MTYP[1:0]		MUXEN	MBKEN
0x10	FMC_BCR3	Reserved												CBURSTRW	CPSIZE[2:0]			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	WRAPMOD	WAITPOL	BURSTEN	Reserved	FACCEN	MWID[1:0]		MTYP[1:0]		MUXEN	MBKEN
0x18	FMC_BCR4	Reserved												CBURSTRW	CPSIZE[2:0]			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	WRAPMOD	WAITPOL	BURSTEN	Reserved	FACCEN	MWID[1:0]		MTYP[1:0]		MUXEN	MBKEN
0x04	FMC_BTR1	Res.		ACCMOD[1:0]		DATLAT[3:0]				CLKDIV[3:0]				BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
0x0C	FMC_BTR2	Res.		ACCMOD[1:0]		DATLAT[3:0]				CLKDIV[3:0]				BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
0x14	FMC_BTR3	Res.		ACCMOD[1:0]		DATLAT[3:0]				CLKDIV[3:0]				BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
0x1C	FMC_BTR4	Res.		ACCMOD[1:0]		DATLAT[3:0]				CLKDIV[3:0]				BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
0x104	FMC_BWTR1	Res.		ACCMOD[1:0]		Res.								BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x10C	FMC_BWTR2	Res.		ACCMOD[1:0]		Res.								BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
0x104	FMC_BWTR3	Res.		ACCMOD[1:0]		Res.								BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
0x10C	FMC_BWTR4	Res.		ACCMOD[1:0]		Res.								BUSTURN[3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
0x60	FMC_PCR2	Reserved												ECCPS[2:0]			TAR[3:0]				TCLR[3:0]				Res.	ECCEN	PWID[1:0]		PTYP	PBKEN	PWAITEN	Reserved
0x80	FMC_PCR3	Reserved												ECCPS[2:0]			TAR[3:0]				TCLR[3:0]				Res.	ECCEN	PWID[1:0]		PTYP	PBKEN	PWAITEN	Reserved
0xA0	FMC_PCR4	Reserved												ECCPS[2:0]			TAR[3:0]				TCLR[3:0]				Res.	ECCEN	PWID[1:0]		PTYP	PBKEN	PWAITEN	Reserved
0x64	FMC_SR2	Reserved																								FEMPT	IFEN	ILEN	IREN	IFS	ILS	IRS	
0x84	FMC_SR3	Reserved																								FEMPT	IFEN	ILEN	IREN	IFS	ILS	IRS	
0xA4	FMC_SR4	Reserved																								FEMPT	IFEN	ILEN	IREN	IFS	ILS	IRS	
0x68	FMC_PMEM2	MEMHIZ[7:0]								MEMHOLD[7:0]								MEMWAIT[7:0]								MEMSET[7:0]							
0x88	FMC_PMEM3	MEMHIZ[7:0]								MEMHOLD[7:0]								MEMWAIT[7:0]								MEMSET[7:0]							
0xA8	FMC_PMEM4	MEMHIZ[7:0]								MEMHOLD[7:0]								MEMWAIT[7:0]								MEMSET[7:0]							
0x6C	FMC_PATT2	ATTHIZ[7:0]								ATTHOLD[7:0]								ATTWAIT[7:0]								ATTSET[7:0]							
0x8C	FMC_PATT3	ATTHIZ[7:0]								ATTHOLD[7:0]								ATTWAIT[7:0]								ATTSET[7:0]							
0xAC	FMC_PATT4	ATTHIZ[7:0]								ATTHOLD[7:0]								ATTWAIT[7:0]								ATTSET[7:0]							
0xB0	FMC_PIO4	IOHIZ[7:0]								IOHOLD[7:0]								IOWAIT[7:0]								IOSET[7:0]							
0x74	FMC_ECCR2	ECC[31:0]																															
0x94	FMC_ECCR3	ECC[31:0]																															
0x140	FMC_SDCR_1	Reserved																		RPIPE[1:0]		RBURST	CLK[1:0]		WP	CAS[1:0]		NB	MWID[1:0]		NR[1:0]		NC[1:0]	
0x144	FMC_SDCR_2	Reserved																					CLK[1:0]		WP	CAS[1:0]		NB	MWID[1:0]		NR[1:0]		NC[1:0]	
0x148	FMC_SDTR1	Reserved		TRCD[3:0]				TRP[3:0]				TWR[3:0]				TRC[3:0]						TRAS[3:0]				TXSR[3:0]				TMRD[3:0]			
0x14C	FMC_SDTR2	Reserved		TRCD[3:0]				TRP[3:0]				TWR[3:0]				TRC[3:0]						TRAS[3:0]				TXSR[3:0]				TMRD[3:0]			
0x150	FMC_SDCMR	Reserved												MRD[12:0]												NRFS[3:0]			CTB1	CTB2	MODE[2:0]		
0x154	FMC_SDRTR	Reserved																		REIE		COUNT[12:0]											CRE
0x158	FMC_SDSR	Reserved																										BUSY	MODES2[1:0]		MODES1[1:0]		RE

38. Поддержка отладки (DBG)

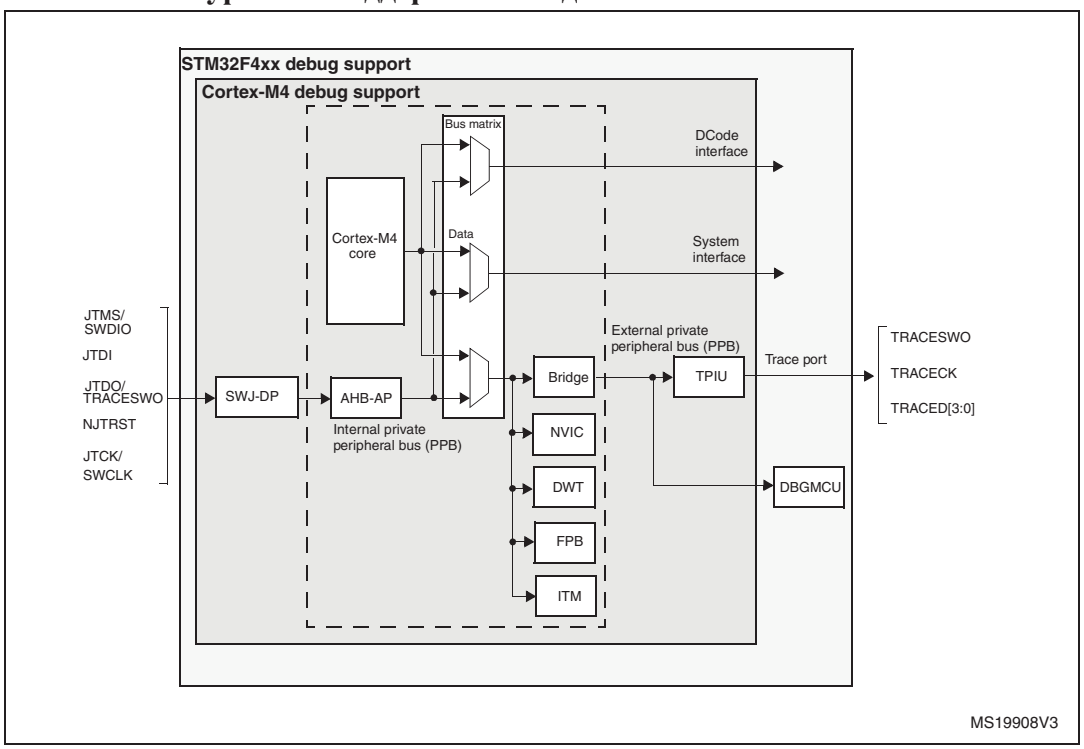
38.1. Обзор

STM32F4xx построены вокруг ядра Cortex-M4 с аппаратной поддержкой отладки. Она позволяет останавливать ядро по выборке команды (контрольная точка) или доступу к данным (точка осмотра). Состояние остановленного ядра можно разглядывать на хосте.

Есть два интерфейса:

- Последовательный провод
- Порт отладки JTAG

Блок-схема уровней поддержки отладки STM32 MCU и Cortex-M4



NB: Средства отладки Cortex-M4 входят в Arm CoreSight Design Kit.

Средства отладки Arm Cortex-M4 включают:

- SWJ-DP: Последовательный провод / Порт отладки JTAG
- AHP-AP: Порт доступа к AHB
- ITM: Макроячейка инструментальной трассировки
- FPB: Контрольная точка патча Flash
- DWT: Включение точки просмотра данных
- TPUI: Интерфейс блока порта трассировки (у корпусов с нужными ножками)
- ETM: Встроенная макроячейка трассировки (у корпусов с нужными ножками)

У отладки STM32F10xxx есть:

- Гибкое назначение точек отладки
- Блок отладки MCU (поддержка экономных режимов, управление тактами и т.д.)

38.2. Документация от фирмы ARM

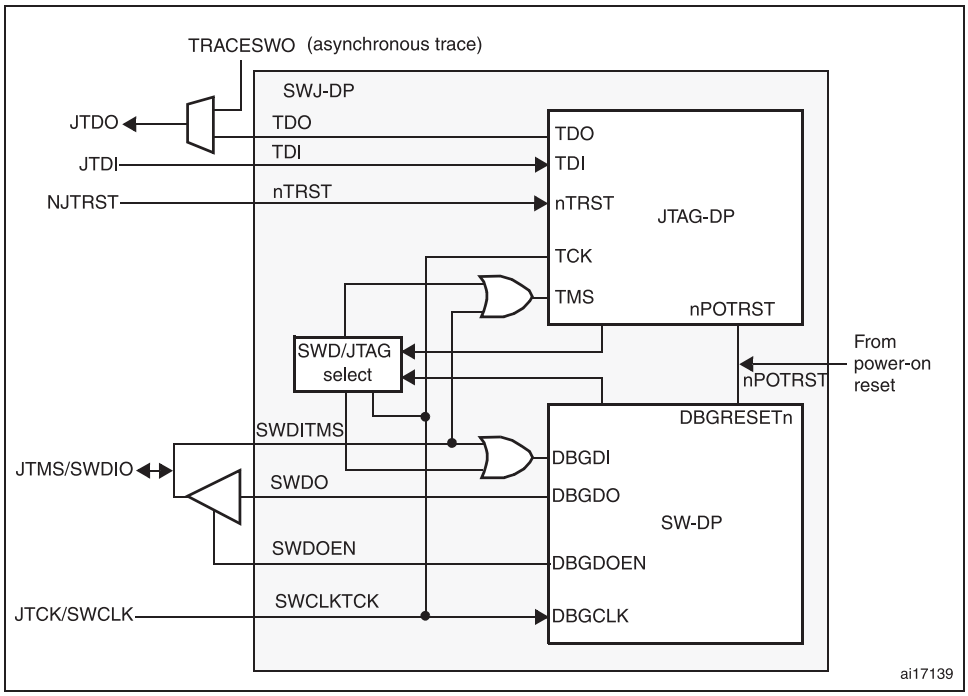
- Cortex-M4 Technical Reference Manual (TRM) по адресу: *http://infocenter.arm.com/*
- Arm Debug Interface V5
- Arm CoreSight Design Kit revision r0p1 Technical Reference Manual

38.3. Порт отладки SWJ (последовательный провод и JTAG)

- JTAG Debug Port (JTAG-DP) это 5-контактный стандартный интерфейс JTAG с портом AHP-AP.
- The Serial Wire Debug Port (SW-DP) это 2-контактный (такты + данные) интерфейс с портом AHP-AP.

В SWJ-DP две ножки JTAG SW-DP мультиплексированы с некоторыми ножками JTAG из JTAG-DP.

На рисунке ниже асинхронный выход TRACE (TRACESWO) мультиплексируется с TDO. То есть асинхронная трассировка может использоваться с SW-DP, но не с JTAG-DP.



38.3.1. Механизм выбора JTAG-DP или SW-DP

По умолчанию активен порт JTAG-Debug.

Для переключения на SW-DP надо выдать JTAG последовательность TMS/TCK (на SWDIO и SWCLK):

1. Послать более 50 тактов TCK с TMS (SWDIO) =1
2. Послать 16 бит на TMS (SWDIO) = 0111100111100111 (старший бит первым)
3. Послать более 50 тактов TCK с TMS (SWDIO) =1

38.4. Распиновка и ножки порта отладки

Варианты MCU STM32F4xx выпускаются в корпусах с разным числом ножек и некоторые функции в разных корпусах могут различаться.

38.4.1. Ножки порта SWJ

Имя ножки SWJ-DP	Порт JTAG		Порт SW		Ножка
	Тип	Описание	Тип	Назначение	
JTMS/SWDIO	I	Выбор режима теста JTAG	IO	Ввод/Вывод SW	PA13
JTCK/SWCLK	I	Такты теста JTAG	I	Такты SW	PA14
JTDI	I	Ввод данных JTAG	-	-	PA15
JTDO/ TRACESWO	O	Вывод данных JTAG	-	TRACESWO если разрешена асинхронная трассировка	PB3
NJTRST	I	nReset теста JTAG	-	-	PB4

38.4.2. Гибкое назначение ножек SWJ-DP

После RESET (SYSRESETn или PORESETn) все 5 ножек SWJ-DP сразу доступны отладчику (выходы трассировки надо назначать явно).

Распределением ножек ведает программа через регистр MCU (AFIO_MAPR).

Используются три бита, обнуляемых по сбросу.

• AFIO_MAPR (@ 0x40010004)

– READ: APB - Без состояний ожидания
– WRITE: APB - 1 состояние ожидания если буфер записи моста AHB-APB полон.

Биты 26:24= **SWJ_CFG[2:0]**

Ставятся и снимаются программно.

Это число ножек, отведённых порту SWJ, дабы освободить их для GPIO.

По умолчанию равно “000” (все от подключения JTAG-DP). Можно ставить только один бит.

Порты отладки	Ножки IO для SWJ				
	PA13 / JTMS / SWDIO	PA14 / JTCK / SWCLK	PA15 / JTDI	PB3 / JTDO	PB4 / NJTRST
Полн. SWJ (JTAG-DP + SW-DP) - Сброс	X	X	X	X	X
Полн, SWJ (JTAG-DP + SW-DP) без NJTRST	X	X	X	X	Свободно
JTAG-DP выкл. SW-DP вкл.	X	X	Свободно		
JTAG-DP выкл. SW-DP вкл.	Свободно				

NB: Если буфер записи моста APB полон, то запись в регистр `AFIO_MAPR` занимает один дополнительный такт APB из-за двух тактов деактивации ножек JTAGSW для гарантии чистого уровня входных сигналов ядра nTRST и TCK.

- Такт 1: входные сигналы JTAGSW ставятся в 1 для nTRST, TDI и TMS и в 0 для TCK
- Такт 2: контроллер GPIO занимает ножки для сигналов SWJTAG (вроде направления, подпорки/подтяжки, активации триггеров Шмитта и пр.).

38.4.3. Внутренняя подпорка и подтяжка ножек JTAG

Входы JTAG не должны быть плавающими, поскольку они напрямую подключены к триггерам. Особенно ножка SWCLK/TCK.

Входы JTAG имеют:

- NJTRST: подпорку
- JTDI: подпорку
- JTMS/SWDIO: подпорку
- TCK/SWCLK: подтяжку

Ножка JTAG IO возвращается контроллеру GPIO. По сбросу ножки GPIO получают состояние:

- NJTRST: подпорки
- JTDI: подпорки
- JTMS/SWDIO: подпорки
- JTCK/SWCLK: подтяжки
- JTDO: плавающие AF выход/вход

Это стандартные GPIO.

NB: Стандарт JTAG IEEE советует подпирать TDI, TMS и nTRST, для TCK советов нет. Но для JTCK внутренняя подтяжка нужна.

38.4.4. Последовательный провод и свободные ножки GPIO

Последовательный провод DP выделяется установкой release SWJ_CFG=010 сразу после сброса. Ножки PA15, PB3 и PB4 освобождаются.

При отладке хост:

- Под системным сбросом, назначает все ножки SWJ (JTAG-DP + SW-DP).
- Под системным сбросом, посылает последовательность переключения из JTAG-DP в SW-DP.
- Под системным сбросом, отладчик ставит контрольную точку на вектор сброса.
- Системный сброс снимается и ядро останавливается.
- Вся связь отладки идет по SW-DP, остальные ножки свободны.

NB: Освобождённые ножки отладки некоторое время остаются в состоянии подпорки-входа (nTRST, TMS, TDI), подтяжки (TCK) или в третьем состоянии выхода (TDO) вплоть до момента освобождения программой.

Если ножки отладки (JTAG or SW or TRACE) отведены отладке, то конфигурация контроллера IOPORT на них не влияет.

38.5. Подключение STM32F4xx JTAG TAP

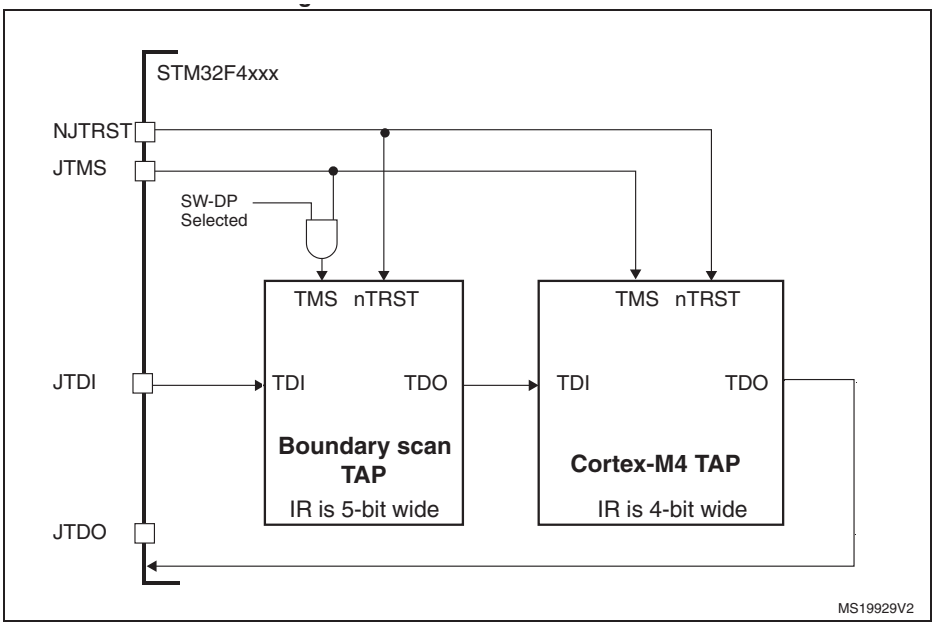
В STM32F4xx MCU есть два последовательно включённых JTAG TAP, сканер границ TAP (IR шириной 5-бит) и Cortex-M3 TAP (IR шириной 4-бит).

Доступ к TAP Cortex-M3 для отладки:

1. Сначала вдвигаем команду BYPASS сканера границ TAP.
2. Затем, для каждого сдвига IR, цепочка скана содержит 9 бит (=5+4) и должна быть вдвинута команда неиспользованного TAP с помощью команды BYPASS.
3. По каждому сдвигу данных, неиспользованный TAP, стоящий в режиме BYPASS, добавляет 1 лишний бит в цепочку скана данных.

NB: **Важно**: Поскольку SW включается JTAG последовательностью, то сканер границ TAP автоматически отключается (JTMS высокий).

Подключение JTAG TAP



38.6. ID коды и блокировка

Разработчикам инструментов рекомендуется блокировать их изделия по коду DEVICE ID, лежащему во внешней PPB памяти по адресу 0xE0042000.

38.6.1. ID код устройства MCU

ID содержит номер части ST MCU ревизию кристалла. Это часть компонента DBG_MCU. Он доступен и портам отладки JTAG и SW и программе даже во время сброса MCU.

Отладчики должны использовать только `DEV_ID`[11:0].

DBGMCU_IDCODE

Адрес: 0xE004 2000

Только чтение, доступ 32 бита.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV_ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DEV_ID[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

— Биты 31:16 **REV_ID[15:0]:** Ревизия

STM32F405xx/07xx и STM32F415xx/17xx:

0x1000 = Ревизия A

0x1001 = Ревизия Z

0x1003 = Ревизия 1

0x1007 = Ревизия 2

0x100F= Ревизия Y и 4

STM32F42xxx и STM32F43xxx:
0x1000 = Ревизия A
0x1003 = Ревизия Y
0x1007 = Ревизия 1
0x2001= Ревизия 3, 4, 5 и B
0x2007= Ревизия 4, 5 и B

— Биты 15:12 Резерв, не трогать.

— Биты 11:0 **DEV[15:0]:** Идентификатор устройства
STM32F405xx/07xx и STM32F415xx/17xx: 0x413.
STM32F42xxx и STM32F43xxx: 0x419

38.6.2. Сканер границ TAP (BSC)

ID код JTAG

- 0x06413041 для STM32F405xx/07xx и STM32F415xx/17xx
- 0x06419041 для STM32F42xxx и STM32F43xxx devices

38.6.3. Cortex-M4 с FPU TAP

Cortex-M4 JTAG ID код неизменяемый, доступен только по JTAG DP, равен 0x4BA00477.

38.6.4. Cortex-M4 JEDEC-106 ID код

Он лежит в 4KB таблице ROM на внутренней шине PPB по адресу 0xE00FF000_0xE00FFFFF. Доступен по JTAG DP, SW DP и программе.

38.7. Порт JTAG DP

Это стандартная машина состояний JTAG с 4-бит регистром команд (IR) и 4 регистрами данных.

Таблица 300. Регистры данных JTAG DP

IR(3:0)	Регистр	Детали
1111	BYPASS [1 бит]	—
1110	IDCODE [32 бита]	ID CODE = 0x4BA00477 (Cortex-M4 with FPU r0p1 ID Code)
1010	DPACC [35 бит]	Регистр доступа к порту отладки, инициализирует порт и разрешает доступ к регистрам. – При передаче данных IN: Биты 34:3 = DATA[31:0] = 32-бит данные для запроса записи Биты 2:1 = A[3:2] = 2-бит адрес регистра порта отладки. См. *Таблицу 220*. Бит 0 = RnW = Запрос чтения (1) или записи (0). – При передаче данных OUT: Биты 34:3 = DATA[31:0] = 32-бит данные для запроса чтения Биты 2:0 = ACK[2:0] = 3-бит Подтверждения: 010 = OK/FAULT 001 = WAIT OTHER = резерв
1011	APACC [35 бит]	Регистр доступа к порту доступа, инициализирует порт и разрешает доступ к регистрам. – При передаче данных IN: Биты 34:3 = DATA[31:0] = 32-бит данные для запроса записи Биты 2:1 = A[3:2] = 2-бит адрес регистра AP. Бит 0 = RnW = Запрос чтения (1) или записи (0). – При передаче данных OUT: Биты 34:3 = DATA[31:0] = 32-бит данные для запроса чтения Биты 2:0 = ACK[2:0] = 3-бит Подтверждения: 010 = OK/FAULT 001 = WAIT OTHER = резерв Регистров AP много (см. AHB-AP) они адресуются комбинацией: – значения A[3:2] – текущего значения регистра DP SELECT
1000	ABORT [35 бит]	Аборт-регистр – Биты 31:1 = Резерв – Бит 0 = DAPABORT: запись 1 делает аборт DAP.

Таблица 301. 32-бит регистры порта отладки по адресу A[3:2]

Адрес	A[3:2]	Описание
0x0	00	Резерв, не трогать.
0x4	01	Регистр DP CTRL/STAT для: – Запроса включения питания системы или отладки – Конфигурации передач доступа к AP – Управление сравнением и проверкой вталкивания. – Чтения некоторых флагов (переполнение, подтверждения включения)
0x8	10	Регистр DP SELECT для выбора порта доступа и 4-слов окна регистров: – Биты 31:24: APSEL: выбор текущего AP – Биты 23:8: резерв – Биты 7:4: APBANKSEL: выбор 4-слов окна регистров текущего AP – Биты 3:0: резерв
0xC	11	Регистр DP RDBUFF для получения результата последовательности операций (без запроса новой операции JTAG-DP)

38.8. Порт SW DP

38.8.1. Введение в протокол SW

Имеет две линии:

- SWCLK: такты от хоста
- SWDIO: двунаправленная

Протокол имеет два банка регистров (DPACC и APACC) для чтения и записи.

Биты передаются начиная с младшего.

Линия SWDIO должна иметь внешнюю подпорку на плате (рекомендуется 100 KΩ).

При каждом изменении направления SWDIO вставляется задержка переключения. Обычно это время одного бита, зависит от частоты SWCLK.

38.8.2. Последовательность протокола SW

Состоит из трёх фаз:

1. Пакет запроса (8 бит) от хоста
2. Ответ подтверждения (3 бита) для хоста
3. Передача данных (33 бита) от или для хоста

Таблица 302. Пакет запроса (8-бит)

Бит	Имя	Описание
0	Start	Должен быть “1”
1	APnDP	0: DP, 1: AP
2	RnW	0: Запрос записи 1: Запрос чтения
4:3	A[3:2]	Поле адреса регистров DP или AP (см. *Таблицу 220.*)
5	Parity	Бит чётности
6	Stop	0
7	Park	Не выдаётся хостом. Должен читаться “1” из-за подпорки

Пакет запроса всегда сопровождается задержкой переключения (обычно 1 бит).

Таблица 303. Ответ ACK (3 бита)

Бит	Имя	Описание
0..2	ACK	001: FAULT 010: WAIT 100: OK

Ответ ACK должен сопровождаться задержкой переключения для передачи READ или при получении ответа WAIT или FAULT.

Таблица 304. Передача DATA (33 бита)

Бит	Имя	Описание
0..31	WDATA RDATA	Данные записи или чтения
32	Parity	Бит чётности для 32 битов данных

Передача DATA должна сопровождаться задержкой переключения только для передачи READ.

38.8.3. Машина состояний SW-DP (сброс, простой, ID код)

Код ID машины состояний SW-DP соответствует стандарту JEP-106 и равен 0x2BA01477.

NB: Пока устройство читает ID код, машина состояний SW-DP неактивна.

- Машина идёт в состояние RESET после сброса по питанию, после переключения DP из JTAG в SWD или после более чем 50 тактов высокого состояния линии
- Машина идёт в состояние IDLE если линия низкая не менее двух тактов после состояния RESET.
- После состояния RESET при входе в IDLE надо сначала **обязательно** выполнить READ к регистру DP-SW ID CODE. На другой запрос устройство выдаст ответ FAULT.

38.8.4. Доступ чтения/записи DP и AP

- Чтение DP не постируется: ответ может быть немедленным (ACK=OK) или задержанным (ACK=WAIT).
- Чтение AP постируется, то есть результат возвращается в следующей передаче. Если следующее обращение это не доступ к AP, то результат надо получить из регистра DP-RDBUFF. Флаг успешного чтения READOK в регистре DP-CTRL/STAT обновляется по каждому запросу чтения из AP или из RDBUFF.
- У SW-DP есть буфер записи (для обоих DP или AP). Если он полон, то устройство отвечает “WAIT” за исключением чтения IDCODE или CTRL/STAT или записи ABORT.
- Из-за асинхронности доменов SWCLK и HCLK после передачи бита чётности операции записи нужны два дополнительных такта SWCLK низкого состояния линии (состояние IDLE). Это особенно важно при записи запроса включения питания CTRL/STAT. Иначе следующая передача будет сбойной.

38.8.5. Регистры SW DP

Доступ к ним идёт при APnDP=0

Таблица 305. Регистры SW DP

A[3:2]	R/W	Бит CTRLSEL регистра SELECT	Регистр	Заметки
0	Read	-	IDCODE	Код производителя не ST code. 0x2BA01477 (это SW-DP)
0	Write	-	ABORT	-
1	Read/ Write	0	DP- CTRL/STAT	Назначение: – запрос включения питания системы или отладки – конфигурация передач для доступа к AP – управление сравнением и проверкой вталкивания. – чтение некоторых флагов(переполнение, подтверждения включения)
1	Read/ Write	1	WIRE CONTROL	Для управления физическим протоколом (вроде длины задержки)
10	Read	-	READ RESEND	Повторная передача данных без повторения запроса AP.
10	Write	-	SELECT	Выбор текущего порта доступа и активного окна регистров
11	Read/ Write	-	READ BUFFER	Чтение буфера после предыдущего запроса чтения AP

38.8.6. Регистры SW AP

Регистры доступны при APnDP=1

Регистров AP много (см. AHB-AP), адресуются они с помощью:

- Значения A[3:2]
- Текущего значения регистра DP SELECT

38.9. AHB-AP (порт доступа к AHB) - для JTAG-DP и SW-DP

Свойства:

- Доступ к системе не зависит от состояния процессора.
- AHB-AP доступен и SW-DP и JTAG-DP.
- AHB-AP является ведущим AHB на Матрице Шин. Отсюда, ему доступны все шины данных (Dcode Bus, System Bus, внутренняя и внешняя шина PPB), кроме шины ICode.
- Поддерживаются Bitband-передачи.
- Передачи AHB-AP минуют FPB.

Адрес 32-бит регистров AHP-AP 6-битный (до 64 слов или 256 байт) и состоит из:

c) Биты [7:4] = биты[7:4] APBANKSEL регистра DP SELECT

d) Биты [3:2] = 2 бита адреса из A[3:2] из 35-бит пакета запроса SW-DP.

AHB-AP Cortex-M4 имеет 9 x 32-бит регистров:

Таблица 306. Регистры Cortex-M4 AHB-AP

Смещение адреса	Имя регистра	Замечания
0x00	Управление и статус AHB-AP	Управление и статус передач AHB
0x04	Адрес передачи AHB-AP	—
0x0C	Данные чтения/записи AHB-AP	—
0x10	Банк данных 0 AHB-AP	Прямое отображение 4 выравненных слов данных без изменения регистра Адреса Передачи.
0x14	Банк данных 1 AHB-AP	
0x18	Банк данных 2 AHB-AP	
0x1C	Банк данных 3 AHB-AP	
0xF8	Адрес ROM отладки AHB-AP	Базовый адрес интерфейса отладки
0xFC	Регистр ID AHB-AP	—

38.10. Отладка ядра

Она доступна по регистрам отладки через порт AHB-AP. Процессор может обращаться к ним по внутренней шине PPB. Их четыре:

Таблица 307. Регистры отладки ядра

Регистр	Описание
DHCSR	32-бит регистр статуса и управления пошаговой отладки ядра.
DCRSR	17-бит регистр выбора регистра ядра.
DCRDR	32-бит регистр данных из/для регистра ядра.
DEMCR	32-бит регистр отладки исключений и мониторинга, содержит бит ***TRCENA*** разрешения TRACE.

NB: **Важно**: эти регистры сбрасываются только по включению питания.

Для Останова по сбросу надо:

- поставить бит 0 (VC_CORRESET) регистра DEMCR.
- поставить бит 0 (C_DEBUGEN) регистра DHCSR.

38.11. Отладка под сбросом системы

Источники сброса системы:

- POR (включение питания) выставляет RESET по каждому включению питания.
- Внутренний сторожевой таймер
- Программный сброс
- Внешний сброс

Cortex-M4 различает сброс отладки (обычно PORRESETn) и другой (SYSRESETn). То есть отладчик может подключиться во время сброса, остановив ядро при чтении вектора сброса. Тогда хост освобождает системный сброс и ядро останавливается без выполнения команды. Кроме того, под системным сбросом можно ставить любые функции отладки.

NB: Настоятельно рекомендуется ставить контрольную точку вектора сброса под системным сбросом.

38.12.FPB (Точка патча Flash)

Блок FPB:

- реализует аппаратные контрольные точки
- направляет команды и данные из пространства кода в системное пространство. Так можно исправлять ошибки в пространстве кода.

Использование Программного патча и Аппаратных точек взаимно исключающее.

FPB содержит:

- 2 компаратора литералов для чтения из пространства кода и направления в пространство системы.
- 6 компараторов команд для выборки из пространства кода. Их можно использовать для переключения в соответствующее пространство системы или выдачи команды контрольной точки.

38.13.DWT (Запуск точки осмотра данных)

Блок DWT состоит из 4 компараторов. Они конфигурируются как:

- аппаратная контрольная точка
- запуск ETM
- читатель PC
- читатель адреса данных

Для профилирования DWT имеет счётчики:

- Тактов
- Вложенных команд
- Блоков операций загрузки /записи (LSU)
- Тактов сна
- CPI (тактов на команду)
- Прерываний

38.14.ITM (Макроячейка инструментальной трассировки)

38.14.1.Общее описание

ITM поддерживает трассировку ОС и программных событий в стиле *printf*. ITM выдаёт пакеты:

- **Software trace.** Программа может писать в регистр стимула ITM.
- **Hardware trace.** Пакеты создаёт DWT, а выдаёт ITM их.
- **Time stamping.** Метки времени выдаются относительно пакетов. Для этого в ITM есть 21-бит счётчик. Считает он такты Cortex-M3 или выходные такты от *Serial Wire Viewer* (SWV).

Пакеты от ITM выдаются TPIU (Trace Port Interface Unit). Форматтер TPIU добавляет свои пакеты (см. TPIU) и выводит полную последовательность пакетов хосту отладчика.

При использовании ITM бит `TRCEN` в регистре `DEMCR` должен стоять.

38.14.2.Пакеты меток времени, синхронизации и переполнения

Пакеты меток времени содержат время, общее управление и синхронизацию. Они используют счётчик меток времени (21 бит) с возможным предделителем, который сбрасывается по каждой выдаче метки. Он тактируется от тактов CPU или SWV.

Пакет синхронизации состоит из 6 байтов 0x80_00_00_00_00_00, которые передаются TPIU как 00 00 00 00 00 80 (младший бит вперёд).

Пакет синхронизации управляет пакетами меток, он выдаётся по каждому запуску DWT.

Для этого DWT должен запускать ITM: стоит CYCCNTENA в регистре управления DWT. Кроме того, должен стоять бит SYNCENA регистре управления ITM.

Если бит SYNCENA не стоит, то DWT выдаёт запуск синхронизации TPIU, который посылает только пакеты синхронизации TPIU, но не пакеты синхронизации ITM.

Пакет переполнения состоит из специальных пакетов меток времени, извещающих о том, что данные писались в полный FIFO.

Таблица 308. Главные регистры ITM

Адрес	Регистр	Детали
@E0000FB0	Блокировка доступа к ITM	Запись 0xC5ACCE55 разрешает запись в другие регистры ITM
@E0000E80	Управление трассировкой ITM	Биты 31-24 = всегда 0
		Бит 23 = Занят
		Биты 22-16 = 7-бит ATB ID источника данных трассировки.
		Биты 15-10 = всегда 0
		Биты 9:8 = TSPrescale = Предделитель меток времени
		Биты 7-5 = Резерв
		Бит 4 = SWOENA = Разрешить такты SWV для меток времени.
		Бит 3 = DWTENA: Разрешить Стимул DWT
		Бит 2 = SYNCENA: 1 разрешает DWT выдавать запуск синхронизации TPIU для выдачи пакетов синхронизации.
		Бит 1 = TSENA (Разрешение меток времени)
		Бит 0 = ITMENA: Глобальный бит разрешения ITM
@E0000E40	Привилегии трассировки ITM	Бит 3: маска разрешения портов трассировки 31:24
		Бит 2: маска разрешения портов трассировки 23:16
		Бит 1: маска разрешения портов трассировки 15:8
		Бит 0: маска разрешения портов трассировки 7:0
@E0000E00	Разрешение трассировки ITM	Каждый бит разрешает один порт Стимула для выдачи трассировки.
@E0000000-E000007C	Регистры портов Стимула 0-31	Запиши 32-бит данные в выбранный порт Стимула (их 32) для его трассировки.

Пример конфигурации

Выводим простое число в TPIU:

- Организуем TPIU и назначаем TRACE I/O записью в DBGMCU_CR
- Пишем 0xC5ACCE55 в регистр блокировки ITM
- Пишем 0x00010005 в регистр управления трассировкой ITM для включения ITM с разрешённым Sync и ATB ID, отличным от 0x00
- Пишем 0x1 в регистр разрешения трассировки ITM для включения порта Стимула 0
- Пишем 0x1 в регистр привилегий трассировки ITM для снятия маски портов Стимула 7:0

- Пишем выводимое число в порт Стимула, регистр 0

38.15.ETM (Встроенная макроячейка трассировки)

38.15.1.Общее описание

ETM позволяет отследить выполнение программы. За данными следят через компонент DWT макроячейки ITM, за командами через ETM.

ETM передаёт информацию пакетами по запуску от встроенных ресурсов. Эти ресурсы программируются независимо, источник запуска определяется регистром События запуска (0xE0041008). Это может быть простым событием (от компаратора адреса) или логическим уравнением 2 событий. Источник запуска это один из 4 компараторов модуля DWT. Отслеживаются события:

- Совпадение такта
- Совпадение адреса данных

ETM выдаёт пакеты TPIU. Форматтер TPIU добавляет свои пакеты и выдаёт полную последовательность пакетов отладчику хоста.

38.15.2.Протокол сигналов ETM и типы пакетов

Эта часть описана в главе 7 документа *ETMv3 Signal Protocol of the Arm IHI 0014N*.

38.15.3.Главные регистры ETM

Таблица 309. Главные регистры ETM

Адрес	Регистр	Детали
0xE0041FB0	Блокировка доступа ETM	Запись 0xC5ACCE55 разрешает запись в другие регистры ETM.
0xE0041000	Управление ETM	Управление общими операциями ETM.
0xE0041010	Статус ETM	Информация о текущем статусе трассировки и логике запуска.
0xE0041008	Событие запуска ETM	Определяет событие запуска.
0xE004101C	Компаратор трассировки ETM	Выбирает компаратор.
0xE0041020	Событие трассировки ETM	Определяет событие трассировки.
0xE0041024	Старт/Стоп трассировки ETM	Определяет события Старта и Стопа трассировки.

38.15.4.Пример конфигурации ETM

Выводим простое число в TPIU:

- Конфигурируем TPIU и разрешаем I/IO_TRACEN для назначения TRACE I/O.
- Пишем 0xC5AC CE55 в регистр ETM Lock Access для записи а регистры ITM
- Пишем 0x0000 1D1E в регистр управления ETM (конфигурируем трассировку)
- Пишем 0x0000 406F в регистр события запуска ETM (выбираем событие)
- Пишем 0x0000 006F в регистр разрешения события ETM (выбираем событие start/stop)
- Пишем 0x0000 0001 в регистр Start/stop ETM (включаем трассировку)
- Пишем 0x0000191E в регистр управления ETM (конец конфигурации)

38.16.Компонент отладки MCU (DBGMCU)

Он поддерживает :

- Режимы пониженного питания
- Управление тактированием таймеров, сторожевых таймеров, I^2C и bxCAN в контрольных точках
- Назначение ножек трассировки

38.16.1.Поддержка отладки в режимах пониженного питания

В эти режимы попадают после команд WFI и WFE. Их несколько.

Ядро не позволяет при отладке выключать FCLK или HCLK. Для отладки ставим регистры конфигурации:

- В режиме Сна отладчик заранее ставит бит DBG_SLEEP в регистре DBGMCU_CR. Так HCLK запитывается от FCLK.
- В режиме Останова отладчик заранее ставит бит DBG_STOP. Так FCLK и HCLK запитываются от внутреннего RC генератора.

38.16.2.Поддержка отладки для таймеров, сторожевиков, bxCAN и I²C

Надо решить будут ли таймеры работать внутри контрольной точки:

- Могут считать. Обычно это нужно при ШИМ управлении моторами и т.п.
- Не могут считать. Это надо сторожевым таймерам.

Для bxCAN можно выбрать блокировку обновления регистра приёма.

Для I²C можно выбрать блокировку таймаута SMBUS.

У остановленных таймеров (DBG_TIMx_STOP = 1) с инверсными выходами, они выключены (как будто снят бит MOE).

38.16.3.Регистр конфигурации отладки MCU

Это относится к поддержке:

- Режимов пониженного питания
- Таймеров и сторожевиков and
- bxCAN
- Назначения ножек трассировки

Регистр DBGMCU_CR лежит на внешней шине PPB по адресу 0xE0042004

Он асинхронно сбрасывается PORESET (не системным). Отладчик может писать в него под системным сбросом. Если хост отладчика не поддерживает этих функций, то это можно делать программно.

DBGMCU_CR register

Адрес: 0xE004 2004

PORESET: 0x0000 0000

Доступен только словами (32 бита)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TRACE_MODE [1:0]		TRACE_IOEN	Reserved		DBG_STANDBY	DBG_STOP	DBG_SLEEP
								rw	rw	rw			rw	rw	rw

— Биты 31:8 Резерв, не трогать.

— Биты 7:5 **TRACE_MODE[1:0]** и **TRACE_IOEN**: Назначение ножек трассировки

– TRACE_IOEN=0:
TRACE_MODE=xx: TRACE ноги не назначены (по умолчанию)

– TRACE_IOEN=1:
– TRACE_MODE=00: Назначения TRACE для асинхронного режима
– TRACE_MODE=01: Назначения TRACE для синхронного режима с TRACEDATA=1
– TRACE_MODE=10: Назначения TRACE для синхронного режима с TRACEDATA=2
– TRACE_MODE=11: Назначения TRACE для синхронного режима с TRACEDATA=4

— Биты 4:3 Резерв, не трогать.

— Бит 2 **DBG_STANDBY**: Режим Standby отладки

0: (FCLK=Off, HCLK=Off) Вся цифровая часть отключена.
Для программы выход из Standby равен выбору вектора сброса (кроме битов статуса)

1: (FCLK=On, HCLK=On) Цифровая часть запитана, FCLK и HCLK кормятся от внутреннего RC генератора. При выходе из Standby MCU выдаёт системный сброс.

— Бит 1 **DBG_STOP:** Режим Stop отладки

0: (FCLK=Off, HCLK=Off) Все такты, включая HCLK и FCLK отключены. Выход из STOP равен выходу после RESET (CPU тактируется от внутреннего 8 MHz RC генератора (HSI)).

1: (FCLK=On, HCLK=On) FCLK и HCLK питаются от внутреннего RC генератора.

В обоих случаях при выходе из STOP надо восстановить нормальное тактирование

— Бит 0 **DBG_SLEEP:** Режим Sleep отладки

0: (FCLK=On, HCLK=Off) FCLK кормится системными тактами, HCLK выключен. Контроллер тактов не сбрасывается.

1: (FCLK=On, HCLK=On) HCLK и FCLK получают системные такты (как заповедано Программой).

38.16.4.Регистр заморозки APB1 MCU (DBGMCU_APB1_FZ)

Регистр DBGMCU_APB1_FZ определяет остановку периферии APB1 при остановленном ядре, лежит на внешней шине PPB по адресу 0xE004 2008

Он асинхронно сбрасывается PORESET (не системным). Отладчик может писать в него под системным сбросом.

Адрес: 0xE004 2008

PORESET: 0x0000 0000

Доступен только словами (32 бита)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					DBG_CAN2_STOP	DBG_CAN1_STOP	Reserved	DBG_I2C3_SMBUS_TIMEOUT	DBG_I2C2_SMBUS_TIMEOUT	DBG_I2C1_SMBUS_TIMEOUT	Reserved				
					rw	rw		rw	rw	rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			DBG_IWDG_STOP	DBG_WWDG_STOP	DBG_RTC_STOP	Reserved	DBG_TIM14_STOP	DBG_TIM13_STOP	DBG_TIM12_STOP	DBG_TIM7_STOP	DBG_TIM6_STOP	DBG_TIM5_STOP	DBG_TIM4_STOP	DBG_TIM3_STOP	DBG_TIM2_STOP
				rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw

0: Как обычно

1: Заморозка

— Биты 31:27 Резерв, не трогать.

— Бит 26 **DBG_CAN2_STOP**: Регистры приёмника CAN2 при остановленном ядре

— Бит 25 **DBG_CAN1_STOP**: Регистры приёмника CAN1 при остановленном ядре

— Бит 24 Резерв, не трогать.

— Бит 23 **DBG_I2C3_SMBUS_TIMEOUT**: Таймаут SMBUS

— Бит 22 **DBG_I2C2_SMBUS_TIMEOUT**: Таймаут SMBUS

— Бит 21 **DBG_I2C1_SMBUS_TIMEOUT**: Таймаут SMBUS

— Биты 20:13 Резерв, не трогать.

— Бит 12 **DBG_IWDG_STOP**: Сторожевик **IWDG**

— Бит 11 **DBG_WWDG_STOP**: Сторожевик **WWDG**

— Бит 10 **DBG_RTC_STOP**: Таймер **RTC**

— Бит 9 Резерв, не трогать.

— Биты 8:0 **DBG_TIMx_STOP**: Таймеры **TIMx** (x=2..7, 12..14)

0: Как обычно

1: Заморозка, выходы выключены.

38.16.5.Регистр заморозки APB2 MCU (DBGMCU_APB2_FZ)

Регистр DBGMCU_APB2_FZ определяет остановку периферии APB2 при остановленном ядре, лежит на внешней шине PPB по адресу 0xE004 200C

Он асинхронно сбрасывается PORESET (не системным). Отладчик может писать в него под системным сбросом.

Адрес: 0xE004 200C

PORESET: 0x0000 0000

Доступен только словами (32 бита)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved													DBG_TIM11_STOP	DBG_TIM10_STOP	DBG_TIM9_STOP
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														DBG_TIM8_STOP	DBG_TIM1_STOP
														rw	rw

— Биты 31:19 Резерв, не трогать.

— Биты 18:16 **DBG_TIMx_STOP**: Таймеры **TIMx** (x=11..9)

0: Как обычно

1: Заморозка, выходы выключены

— Биты 15:2 Резерв, не трогать.

— Биты 1:0 **DBG_TIMx_STOP**: Таймеры **TIMx** (x=8, 1)

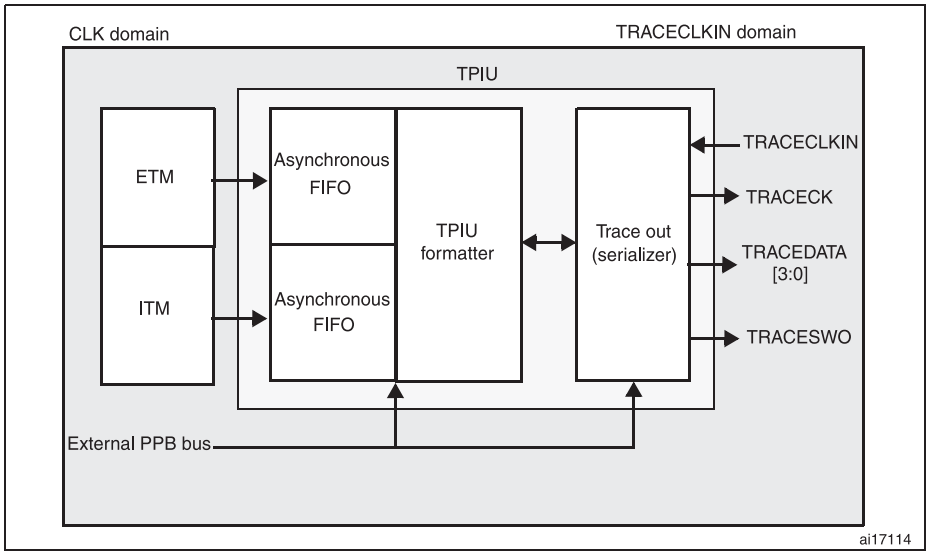
0: Как обычно

1: Заморозка, выходы выключены.

38.17.TPIU (Блок интерфейса порта трассировки)

38.17.1.Введение

TPIU это мост для данных от ITM и ETM. Выходной поток данных включает ID источника трассировки и отправляется *анализатору порта трассировки* (TPA). Тут мы имеем простенький TPIU, специальную версию CoreSight TPIU.



38.17.2.Назначение ног TRACE

- Асинхронный режим

Ему нужна 1 дополнительная нога, она есть всегда. Доступен только в режиме SW (не в JTAG).

Таблица 310. Асинхронная нога TRACE

Имя ножки TPUI	Type	Description	STM32F10xxx pin assignment
TRACESWO	O	TRACE Async Data Output	PB3

- Синхронный режим

Здесь нужны от 2 до 6 дополнительных ног, они есть только в больших корпусах. Доступен в режимах JTAG и SW.

Таблица 311. Синхронные ноги TRACE

Имя ножки TPUI	Type	Description	Нога
TRACECK	O	TRACE Clock	PE2
TRACED[3:0]	O	TRACE Sync Data Outputs Can be 1, 2 or 4.	PE[6:3]

Назначение ножек TRACE TPUI

По умолчанию, эти ножки не назначены. Они назначаются битами TRACE_IOEN и TRACE_MODE в регистре конфигурации отладки MCU. Это выполняет хост. Число ног зависит от режима работы (асинхронный или синхронный).

- асинхронный режим: 1 дополнительная нога
- синхронный режим: от 2 до 5 ног, в зависимости от размера регистра порта (1, 2 или 4):
  - TRACECK
  - TRACED(0) при размере порта 1, 2 или 4
  - TRACED(1) при размере порта 2 или 4
  - TRACED(2) при размере порта 4
  - TRACED(3) при размере порта 4

Хост назначает ноги TRACE битами TRACE_IOEN и TRACE_MODE[1:0] регистра DBGMCU_CR.

Он лежит на внешней шине PPB и сбрасывается PORESET, отладчик может писать всего под системным сбросом.

Таблица 312. Гибкое назначение ног TRACE

DBGMCU_CR		Режим	Нога TRACE IO					
TRACE _IOEN	TRACE _MODE [1:0]		PB3 /JTDO/ TRACESWO	PE2/ TRACECK	PE3 / TRACED[0]	PE4 / TRACED[1]	PE5 / TRACED[2]	PE6 / TRACED[3]
0	XX	No Trace	Свободно[1]	-				
1	0	Асихнрон.	TRACESWO	-	-	Свободно[1]		
1	1	Синхр. 1 бит	Свободно[1]	TRACECK	TRACED[0]	-	-	-
1	10	Синхр. 2 бита		TRACECK	TRACED[0]	TRACED[1]	-	-
1	11	Синхр. 4 бита		TRACECK	TRACED[0]	TRACED[1]	TRACED[2]	TRACED[3]

1. Свободна в режиме SW. В режиме JTAG назначена JTDO.

NB: По умолчанию вход тактов TPIU (TRACECLKIN) замкнут на GND. Он подключается к HCLK через 2 такта после установки бита TRACE_IOEN.

Отладчик ставит режим трассировки в битах PROTOCOL[1:0] регистра SPP_R TPIU.

- PROTOCOL=00: Синхронный
- PROTOCOL=01 или 10: SW (Manchester или NRZ) (асинхронный). По умолчанию - 01

Размер порта TRACE пишется в биты[3:0] регистра CPSPS_R:

- 0x1 — 1 нога (по умолчанию)
- 0x2 — 2 ноги
- 0x8 — 4 ноги

38.17.3.Форматтер TPUI

Он выводит данные 16-байт фреймами:

- 7 байтов данных
- 8 байтов, содержащих:
  - 1 бит (младший) индикатор байта DATA (‘0) или ID (‘1).
  - 7 битов (старшие) с данными или ID источника.
- 1 вспомогательный байт в котором 1 бит соответствует 1 байту из предыдущих 8:

– для байта с данными это бит 0 данных.
– для байта со сменой ID бит показывает, что ID change меняется.

38.17.4.Пакеты синхронизации фреймов TPUI

TPUI выдаёт два типа пакетов синхронизации:

- Пакет Синхронизации Фреймов (пакет Синхронизации Полным Словом)
  Это слово: 0x7F_FF_FF_FF (младший бит выдаётся первым). Оно больше нигде не появляется, источника с ID е 0x7F нету.
  Он периодически выдаётся ***между*** фреймами. В непрерывном режиме TPA их выбрасывает.
- Пакет Синхронизации Полу-словом
  Это полу-слово: 0x7F_FF (младший бит выдаётся первым).
  Он периодически выдаётся ***между*** или ***внутри*** фреймов.
  Выдаётся в непрерывном режиме, показывая TPA, что TRACE порт стоит в режиме IDLE (считывать TRACE не надо). TPA его выбрасывает.

38.17.5.Передача пакета синхронизации фреймов

Регистра Счётчика Синхронизации в TPIU нет и запуск синхронизации выдаёт **DWT**.

Пакет фрейма синхронизации (0x7F_FF_FF_FF) выдаётся:

- после каждого снятия сброса TPIU. Он снимается синхронно с передним фронтом такта TRACECLKIN. То есть выдаётся при стоящем бите TRACE_IOEN регистра DBGMCU_CFG. В этом случае за словом 0x7F_FF_FF_FF не следует форматированный пакет.
- по каждому запуску DWT (DWT уже сообразован). Вариантов два:
  – Если бит SYNENA ITM снят, то выдаётся только слово 0x7F_FF_FF_FF без форматированного пакета за ним.
  – Если бит SYNENA ITM стоит, то последуют пакеты синхронизации ITM (0x80_00_00_00_00_00), форматированные TPUI (добавлен ID источника).

38.17.6.Синхронный режим

Данные могут выдаваться по 4, 2 или 1 ножкам: TRACED(3:0) Для отладчика подаются такты (TRACECK), которые подключаются к HCLK только при работе TRACE.

NB: В этом режиме стабильность тактов не нужна.

TRACE I/O (включая TRACECK) управляются передним фронтом TRACLKIN (равно HCLK). Отсюда частота TRACECK равна HCLK/2.

38.17.7.Асинхронный режим

Это дешёвый вариант вывода через одну ножку TRACESWO. Полоса ограничена.

Ножка TRACESWO мультиплексируется с JTDO и доступна во всех корпусах STM32F10xxx.

Здесь нужна постоянная частота TRACECLKIN с точностью 5% для UART (NRZ) и 10% для Манчестерского кода.

38.17.8.Подключение TRACECLKIN внутри STM32F4xx

Вход TRACECLKIN уже подключён к HCLK. То есть в асинхронном режиме надо использовать фреймы времени при стабильной частоте CPU.

NB: **Важно:** в асинхронном режиме остерегайтесь того, что:

MCU тактируется от внутреннего генератора RC. Под сбросом его частота отличается от частоты после снятия сброса. То есть под сбросом TPA не должен включать трассировку битом TRACE_IOEN поскольку время в пакетах синхронизации под сбросом и без оного будет различаться.

38.17.9.registers TPIU

Регистры TPIU APB доступны только при стоящем бите TRCENA в регистре DEMCR. Иначе всегда читаются нули (их вывод разрешает PCLK в TPIU).

Таблица 313. Важные регистры TPIU

Адрес	Регистр	Описание
0xE0040004	Размер текущего порта	Бит 0: Размер порта = 1 Бит 1: Размер порта = 2 Бит 2: Размер порта = 3, не поддерживается Бит 3: Размер порта = 4 Должен стоять только один бит. По умолчанию размер равен 1 (0x00000001)
0xE00400F0	Протокол выбранной ножки	Биты 1:0 = 00: Синхронный порт 01: SW вывод - Манчестер (по умолчанию) 10: SW вывод - NRZ 11: резерв
0xE0040304	Управление форматтером и сливом	Биты 31-9 = всегда ‘0 Бит 8 = TrigIn = индикация запуска, всегда ‘1 Биты 7-4 = всегда 0 Биты 3-2 = всегда 0 Бит 1 = EnFCont. У синхронного порта (в регистре протокола биты 1:0=00), этот бит всегда ‘1: форматер разрешён в непрерывном режиме. В асинхронном режиме (в регистре протокола биты 1:0 <> 00), этот бит ставится самостоятельно. Бит 0 = всегда 0 По умолчанию общее значение равно 0x102 **NB:** В синхронном режиме ножка TRACECTL наружу не выводится и форматтер всегда разрешён в непрерывном режиме - он вставляет пакеты идентификации источника.
0xE0040300	Статус форматтера и слива	В Cortex-M4 не используется, всегда читается как 0x00000008

38.17.10.Пример конфигурации

- Ставим бит `TRCENA` в регистре `DEMCR`
- В регистр размера порта TPIU пишем нужное число (по умолчанию там `0x1` для 1-бит порта)
- В регистр управления форматтером и сливом TPIU пишем `0x102` (значение по умолчанию)
- В регистре протокола TPIU выбираем синхронный или асинхронный протокол. Пример: `0x2` для асинхронного режима NRZ (как у UART)
- В регистр `DBGMCU` пишем `0x20` (бит `IO_TRACEN`) для назначения TRACE I/O асинхронного режима. В это время выдаётся пакет синхронизации TPIU (`FF_FF_FF_7F`)
- Конфигурируем ITM и пишем в регистр Стимула ITM для выдачи числа.

38.18.Карта регистров DBG

Addr.	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xE004 2000	**DBGMCU _IDCODE**	REV_ID																Reserved				DEV_ID											
	Reset value(1)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0xE004 2004	**DBGMCU_CR**	Reserved											DBG_TIM7_STOP	DBG_TIM6_STOP	DBG_TIM5_STOP	DBG_TIM8_STOP	DBG_I2C2_SMBUS_TIMEOUT	Reserved								TRACE_MODE [1:0]		TRACE_	Reserved			DBG_STANDBY	DBG_STOP	DBG_SLEEP
	Reset value												0	0	0	0	0									0	0	0				0	0	0
0xE004 2008	**DBGMCU_ APB1_FZ**	Reserved					DBG_CAN2_STOP	DBG_CAN1_STOP	Reserved	DBG_I2C3_SMBUS_TIMEOUT	DBG_I2C2_SMBUS_TIMEOUT	DBG_I2C1_SMBUS_TIMEOUT	Reserved							DBG_IWDG_STOP	DBG_WWDG_STOP	Reserved	DBG_RTC_STOP	DBG_TIM14_STOP	DBG_TIM13_STOP	DBG_TIM12_STOP	DBG_TIM7_STOP	DBG_TIM6_STOP	DBG_TIM5_STOP	DBG_TIM4_STOP	DBG_TIM3_STOP	DBG_TIM2_STOP					
	Reset value						0	0	0	0	0	0								0	0		0	0	0	0	0	0	0	0	0	0	0	0
0xE004 200C	**DBGMCU_ APB2_FZ**	Reserved													DBG_TIM11_STOP	DBG_TIM10_STOP	DBG_TIM9_STOP	Reserved														DBG_TIM8_STOP	DBG_TIM1_STOP
	Reset value														0	0	0															0	0

39. Электронная сигнатура устройства

Хранится в Flash области памяти. Пишется производителем.

39.1. Регистр уникального ID устройства (96 бит)

Читать можно байтами, полусловами или словами и делать с ним что хочешь.

Базовый адрес: 0x1FFF 7A10

Смещение адреса: 0x00

Только чтение = 0xXXXX XXXX от производителя.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UID(31:0)																															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

— Биты 31:0 **UID(31:0)**: координаты X и Y на подложке.

Смещение адреса: 0x04

Только чтение = 0xXXXX XXXX от производителя.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
U_ID(63:48)															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U_ID(47:32)															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

— Биты 31:16 **UID(63:48)**: LOT_NUM[23:0] Номер лота (ASCII)

— Биты 15:0 **UID(47:32)**: WAF_NUM[7:0] Номер подложки (ASCII).

Смещение адреса: 0x08

Только чтение = 0xXXXX XXXX от производителя.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UID(95:80)															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UID(79:64)															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

— Биты 63:48 **UID(95:64)**: LOT_NUM[55:24] Номер лота (ASCII)

39.2. Регистр размера памяти

39.2.1. Регистр размера Флэш-памяти

Базовый адрес: 0x1FFF 7A22

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F_SIZE															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

— Биты 15:0 **F_ID(15:0):** Размер памяти от производителя в КБайтах, пример: 0x0400 = 1024 КБ.

Alles machen tru-la-la zusammen!