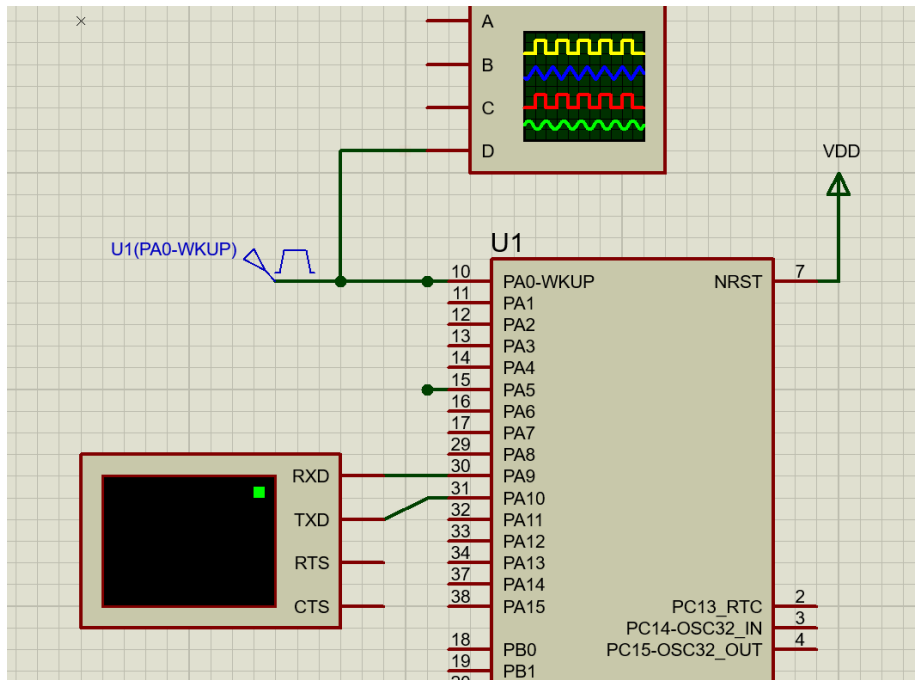
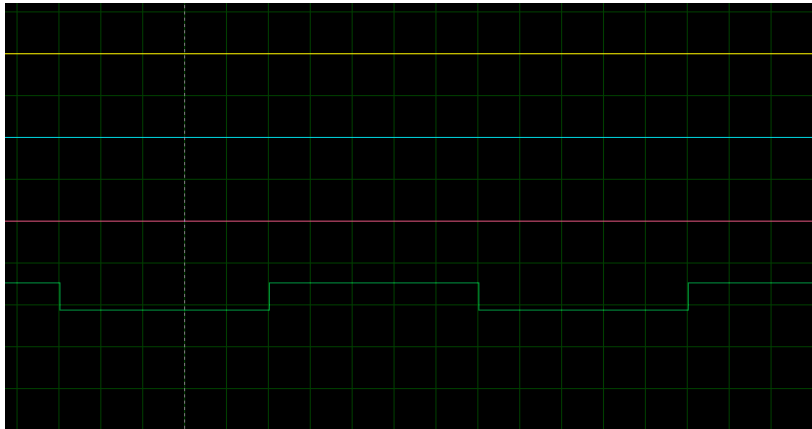


Схема в Proteus



Проверяю, что сигнал есть



Настройка генератора Pulse

Pulse Generator Properties ? X

Generator Name: U1(PA0-WKUP)

Analogue Types

- ☐ DC
- ☐ Sine
- ☒ Pulse
- ☐ Pwlin
- ☐ File
- ☐ Audio
- ☐ Exponent
- ☐ SFFM
- ☐ Random
- ☐ Easy HDL

Digital Types

- ☐ Steady State
- ☐ Single Edge
- ☐ Single Pulse
- ☐ Clock
- ☐ Pattern
- ☐ Easy HDL

☐ Current Source?
☐ Isolate Before?
☐ Manual Edits?
☒ Hide Properties?

Initial (Low) Voltage: 0

Pulsed (High) Voltage: 3.3

Start (Secs): 0

Rise Time (Secs): 1u

Fall Time (Secs): 1u

Pulse Width:

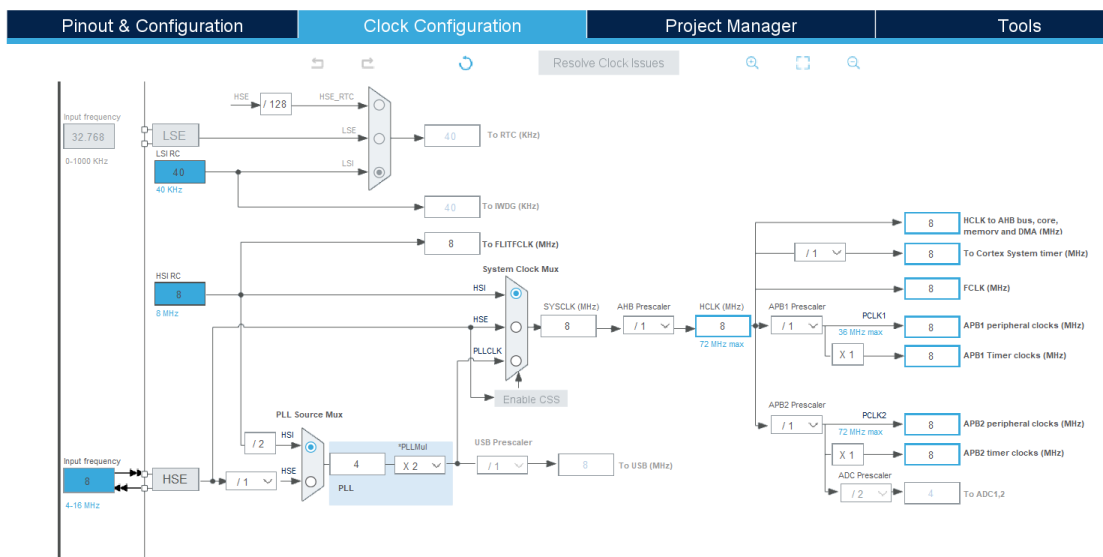
- ☒ Pulse Width (Secs): 5m
- ☐ Pulse Width (%):

Frequency/Period:

- ☒ Frequency (Hz): 100
- ☐ Period (Secs):
- ☐ Cycles/Graph:

OK Cancel

Настройки STM32Cube. В Clock Configuration ничего не меняю



Далее настройки, как в статье

✓ SYS
 WWDG

Analog

⚠ ADC1
 ⚠ ADC2

Timers

RTC
 TIM1
 ✓ TIM2
 TIM3

Connectivity

Slave Mode: Disable
 Trigger Source: Disable
 Clock Source: Internal Clock
 Channel1: Input Capture direct mode
 Channel2: Input Capture indirect mode
 Channel3: Disable
 Channel4: Disable
 Combined Channels: Disable
☐ Use ETR as Clearing Source
☐ XOR activation
☐ One Pulse Mode

Configuration

Reset Configuration

✓ Parameter Settings
 ✓ User Constants
 ✓ NVIC Settings
 ✓ DMA Settings
 ✓ GPIO Settings

✓ Parameter Settings
 ✓ User Constants
 ✓ NVIC Settings
 ✓ DMA Settings
 ✓ GPIO Settings

Configure the below parameters :

Search (Ctrl+F)

Counter Settings

Prescaler (PSC - 16 bits value) 71
 Counter Mode Up
 Counter Period (AutoReload Register - 16 bits va 30000
 Internal Clock Division (CKD) No Division
 auto-reload preload Disable

✓ Trigger Output (TRGO) Parameters
 Master/Slave Mode (MSM bit) Disable (Trigger input effect not delayed)
 Trigger Event Selection Reset (UG bit from TIMx_EGR)

✓ Input Capture Channel 1
 Polarity Selection Rising Edge
 IC Selection Direct
 Prescaler Division Ratio No division
 Input Filter (4 bits value) 0

✓ Input Capture Channel 2
 Polarity Selection Falling Edge
 IC Selection Indirect
 Prescaler Division Ratio No division

Parameter Settings	User Constants	NVIC Settings	DMA Settings	GPIO Settings
NVIC Interrupt Table		Enabled	Preemption Priority	Sub Priority
TIM2 global interrupt		✓	0	0

Код для STM32

```
/* USER CODE BEGIN Header */
```

```
/**
```

```
*****
```

```
* @file      : main.c
```

```
* @brief     : Main program body
```

```
*****
```

```
* @attention
```

```
*
```

```
* Copyright (c) 2025 STMicroelectronics.
```

```
* All rights reserved.
```

*

* This software is licensed under terms that can be found in the LICENSE file

* in the root directory of this software component.

* If no LICENSE file comes with this software, it is provided AS-IS.

*

*/

/* USER CODE END Header */

/* Includes -----*/

#include "main.h"

/* Private includes -----*/

/* USER CODE BEGIN Includes */

#include "string.h"

#include "stdio.h"

/* USER CODE END Includes */

/* Private typedef -----*/

/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define -----*/

/* USER CODE BEGIN PD */

/* USER CODE END PD */

/* Private macro -----*/

/* USER CODE BEGIN PM */

/* USER CODE END PM */

```

/* Private variables -----*/
TIM_HandleTypeDef htim2;

UART_HandleTypeDef huart1;

/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes -----*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART1_UART_Init(void);
static void MX_TIM2_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code -----*/
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/**
 * @brief The application entry point.
 * @retval int
 */
int main(void)
{
    /* USER CODE BEGIN 1 */

    /* USER CODE END 1 */

```

```

/* MCU Configuration-----*/

/* Reset of all peripherals, Initializes the Flash interface and the Systick. */
HAL_Init();

/* USER CODE BEGIN Init */

/* USER CODE END Init */

/* Configure the system clock */
SystemClock_Config();

/* USER CODE BEGIN SysInit */

/* USER CODE END SysInit */

/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_USART1_UART_Init();
MX_TIM2_Init();
/* USER CODE BEGIN 2 */
HAL_TIM_IC_Start_IT(&htim2, TIM_CHANNEL_1);
HAL_TIM_IC_Start_IT(&htim2, TIM_CHANNEL_2);

//Проверяю, что отправка на Virtual Terminal работает
HAL_UART_Transmit(&huart1, (uint8_t*)"Hi!", strlen("Hi!"), 1000);
/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)

```

```

{
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
        HAL_Delay(20);
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
        HAL_Delay(30);
    }
    /* USER CODE END 3 */
}

/**
 * @brief System Clock Configuration
 * @retval None
 */
void SystemClock_Config(void)
{
    RCC_OscInitTypeDef RCC_OscInitStruct = {0};
    RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

    /** Initializes the RCC Oscillators according to the specified parameters
     * in the RCC_OscInitTypeDef structure.
     */
    RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
    RCC_OscInitStruct.HSIState = RCC_HSI_ON;
    RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
    RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
    if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
    {
        Error_Handler();
    }
}

```

```

/** Initializes the CPU, AHB and APB buses clocks
 */

RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYCLK
                               |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;

RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_HSI;
RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0) != HAL_OK)
{
    Error_Handler();
}
}

/**
 * @brief TIM2 Initialization Function
 * @param None
 * @retval None
 */
static void MX_TIM2_Init(void)
{

    /* USER CODE BEGIN TIM2_Init 0 */

    /* USER CODE END TIM2_Init 0 */

    TIM_ClockConfigTypeDef sClockSourceConfig = {0};
    TIM_MasterConfigTypeDef sMasterConfig = {0};
    TIM_IC_InitTypeDef sConfigIC = {0};

```

```

/* USER CODE BEGIN TIM2_Init 1 */

/* USER CODE END TIM2_Init 1 */
htim2.Instance = TIM2;
htim2.Init.Prescaler = 71;
htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
htim2.Init.Period = 30000 ;
htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
if (HAL_TIM_Base_Init(&htim2) != HAL_OK)
{
    Error_Handler();
}
sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
if (HAL_TIM_ConfigClockSource(&htim2, &sClockSourceConfig) != HAL_OK)
{
    Error_Handler();
}
if (HAL_TIM_IC_Init(&htim2) != HAL_OK)
{
    Error_Handler();
}
sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
if (HAL_TIMEx_MasterConfigSynchronization(&htim2, &sMasterConfig) != HAL_OK)
{
    Error_Handler();
}
sConfigIC.ICPolarity = TIM_INPUTCHANNELPOLARITY_RISING;
sConfigIC.ICSelection = TIM_ICSELECTION_DIRECTTI;
sConfigIC.ICPrescaler = TIM_ICPSC_DIV1;
sConfigIC.ICFilter = 0;

```

```

if (HAL_TIM_IC_ConfigChannel(&htim2, &sConfigIC, TIM_CHANNEL_1) != HAL_OK)
{
    Error_Handler();
}

sConfigIC.ICPolarity = TIM_INPUTCHANNELPOLARITY_FALLING;
sConfigIC.ICSelection = TIM_ICSELECTION_INDIRECTTI;
if (HAL_TIM_IC_ConfigChannel(&htim2, &sConfigIC, TIM_CHANNEL_2) != HAL_OK)
{
    Error_Handler();
}

/* USER CODE BEGIN TIM2_Init 2 */

/* USER CODE END TIM2_Init 2 */

}

/**
 * @brief USART1 Initialization Function
 * @param None
 * @retval None
 */
static void MX_USART1_UART_Init(void)
{
    /* USER CODE BEGIN USART1_Init 0 */

    /* USER CODE END USART1_Init 0 */

    /* USER CODE BEGIN USART1_Init 1 */

    /* USER CODE END USART1_Init 1 */
    huart1.Instance = USART1;

```

```

huart1.Init.BaudRate = 1200;

huart1.Init.WordLength = UART_WORDLENGTH_8B;

huart1.Init.StopBits = UART_STOPBITS_1;

huart1.Init.Parity = UART_PARITY_NONE;

huart1.Init.Mode = UART_MODE_TX_RX;

huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;

huart1.Init.OverSampling = UART_OVERSAMPLING_16;

if (HAL_UART_Init(&huart1) != HAL_OK)
{
    Error_Handler();
}

/* USER CODE BEGIN USART1_Init 2 */

/* USER CODE END USART1_Init 2 */

}

/**
 * @brief GPIO Initialization Function
 * @param None
 * @retval None
 */
static void MX_GPIO_Init(void)
{
    GPIO_InitTypeDef GPIO_InitStructure = {0};
    /* USER CODE BEGIN MX_GPIO_Init_1 */
    /* USER CODE END MX_GPIO_Init_1 */

    /* GPIO Ports Clock Enable */

    __HAL_RCC_GPIOD_CLK_ENABLE();

    __HAL_RCC_GPIOA_CLK_ENABLE();

```

```

/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);

/*Configure GPIO pin : PA5 */
GPIO_InitStruct.Pin = GPIO_PIN_5;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}

/* USER CODE BEGIN 4 */

//Если прерывание произошло - продублировать сообщение
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
{
    if (htim->Instance == TIM2)
    {
        if(htim->Channel == HAL_TIM_ACTIVE_CHANNEL_1)
        {
            TIM2->CNT = 0;

            HAL_UART_Transmit(&huart1, (uint8_t*)"Hi!", strlen("Hi!"), 1000);
        }
    }
}

/* USER CODE END 4 */

/**
 * @brief This function is executed in case of error occurrence.

```

```

* @retval None

*/

void Error_Handler(void)
{
    /* USER CODE BEGIN Error_Handler_Debug */
    /* User can add his own implementation to report the HAL error return state */
    __disable_irq();
    while (1)
    {
    }
    /* USER CODE END Error_Handler_Debug */
}

#ifdef USE_FULL_ASSERT
/**
 * @brief Reports the name of the source file and the source line number
 *        where the assert_param error has occurred.
 * @param file: pointer to the source file name
 * @param line: assert_param error line source number
 * @retval None
 */
void assert_failed(uint8_t *file, uint32_t line)
{
    /* USER CODE BEGIN 6 */
    /* User can add his own implementation to report the file name and line number,
       ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
    /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */

```