

**dinartal** 11 августа 2016 в 19:06

Портирование FreeModbus под STM32. Версия от Динара

Программирование микроконтроллеров*

[Из песочницы](#)

Доброго времени суток, любители и профессионалы программирования на микроконтроллерах. Эта статья посвящена портированию библиотеки freemodbus на STM32F100 (тот, что в discovery v1). Да, на habrahabr уже есть подобная [статья](#), но мне она кажется не самой удачной. Буду использовать Modbus RTU в режиме slave. Для успешного портирования библиотеки freemodbus на платформу без операционной системы, необходимо выполнить три шага:

1. прописать файл port.h
2. настроить таймер
3. настроить usart

Итак, план составлен — пора за работу.

Для удобства, сгенерируем проект при помощи STM Cube для IAR. Нам потребуется включить отладку, настроить таймер и я также задействовал кварцы, которые присутствуют на плате.

▼ [Скриншоты STM Cube](#)

STM32CubeMX mbport.ioc: STM32F100RBTx

File Project Pinout Window Help

Pinout Clock Configuration Configuration Power Consumption Calculator

Configuration

MiddleWares

- FATFS
- FreeRTOS

Peripherals

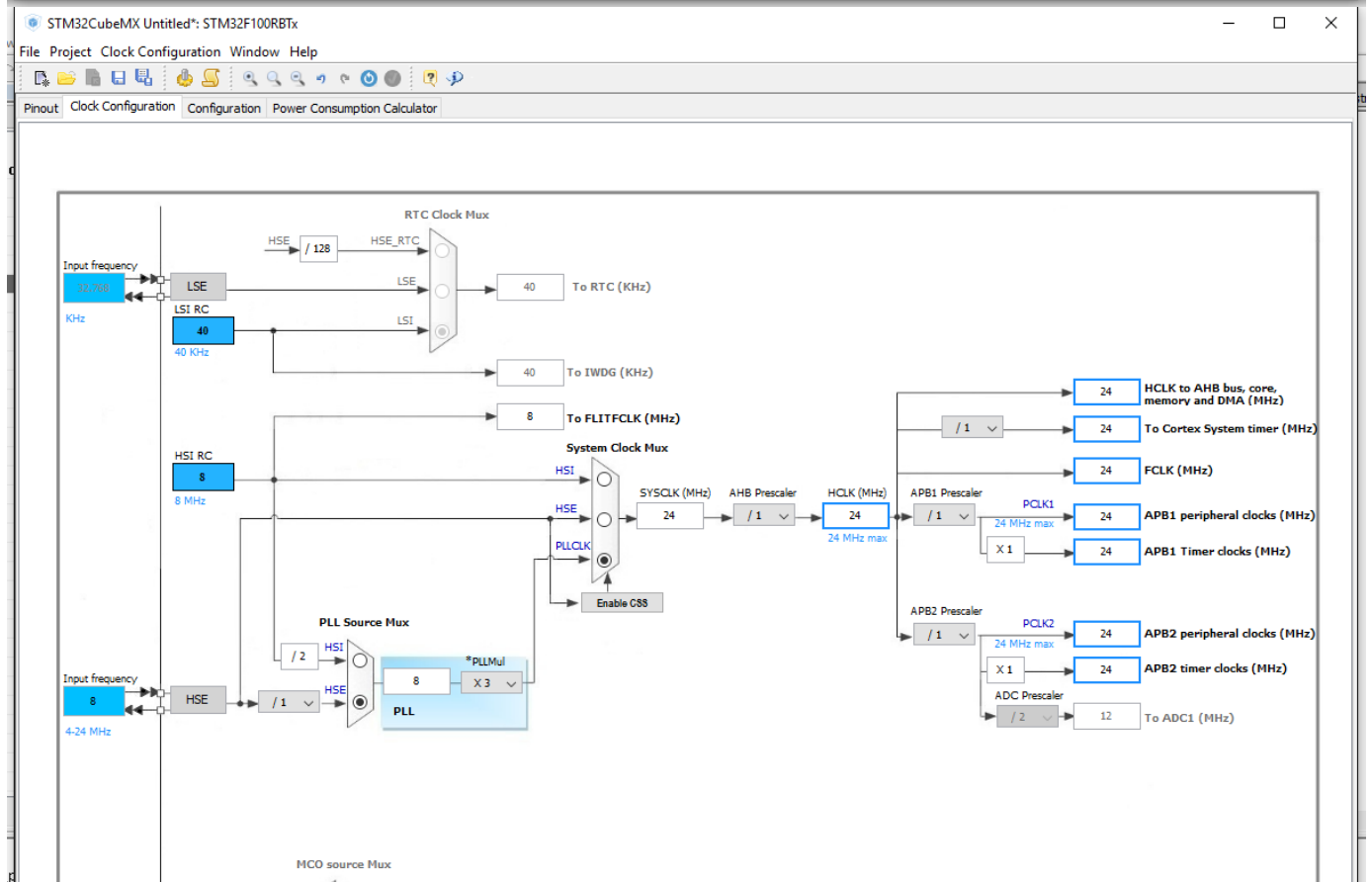
- ADC1
- CRC
- DAC
- HDML_CEC
- I2C1
- I2C2
- IWDG
- RCC
 - High Speed Clock (HSE) Crystal/Ceramic Resona...
 - Low Speed Clock (LSE) Crystal/Ceramic Resona...
 - Master Clock Output
- RTC
- SPI1
- SPI2
- SYS
 - Debug Serial Wire
 - System Wake-Up
 - Timebase Source SysTick
- TIM1
- TIM2
- TIM3
- TIM4
- TIM6
 - Activated
 - One Pulse Mode
- TIM7
- TIM15
- TIM16
- TIM17
- USART1
- USART2
- USART3
- WWDG

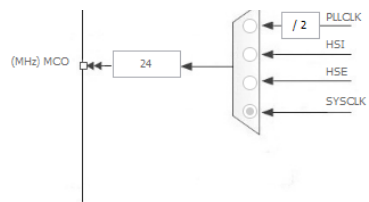
RCC_OSC32_IN
RCC_OSC32_OUT
RCC_OSC_IN
RCC_OSC_OUT

STM32F100RBTx
LQFP64

PA14 SYS_JTCK-SWCLK
PA13 SYS_JTMS-SWDIO
PC9 Debug

Series	Lines	Mcu	Package	Required Peripherals
STM32F1	STM32F100 Value Line	STM32F100R4Tx	LQFP64	None
STM32F1	STM32F100 Value Line	STM32F100R6Tx	LQFP64	None
STM32F1	STM32F100 Value Line	STM32F100R8Tx	LQFP64	None
STM32F1	STM32F100 Value Line	STM32F100RBTx	LQFP64	None
STM32F1	STM32F100 Value Line	STM32F100RCTx	LQFP64	None
STM32F1	STM32F100 Value Line	STM32F100DCTx	LQFP64	None





TIM6 Configuration



☒ Parameter Settings
 ☒ User Constants
 ☒ NVIC Settings
 ☒ DMA Settings

Configure the below parameters :

Search :



Counter Settings

Prescaler (PSC - 16 bits value)	23
Counter Mode	Up
Counter Period (AutoReload Register - 16 bits value)	49

Trigger Output (TRGO) Parameters

Trigger Event Selection	Reset (UG bit from TIMx_EGR)
-------------------------	------------------------------

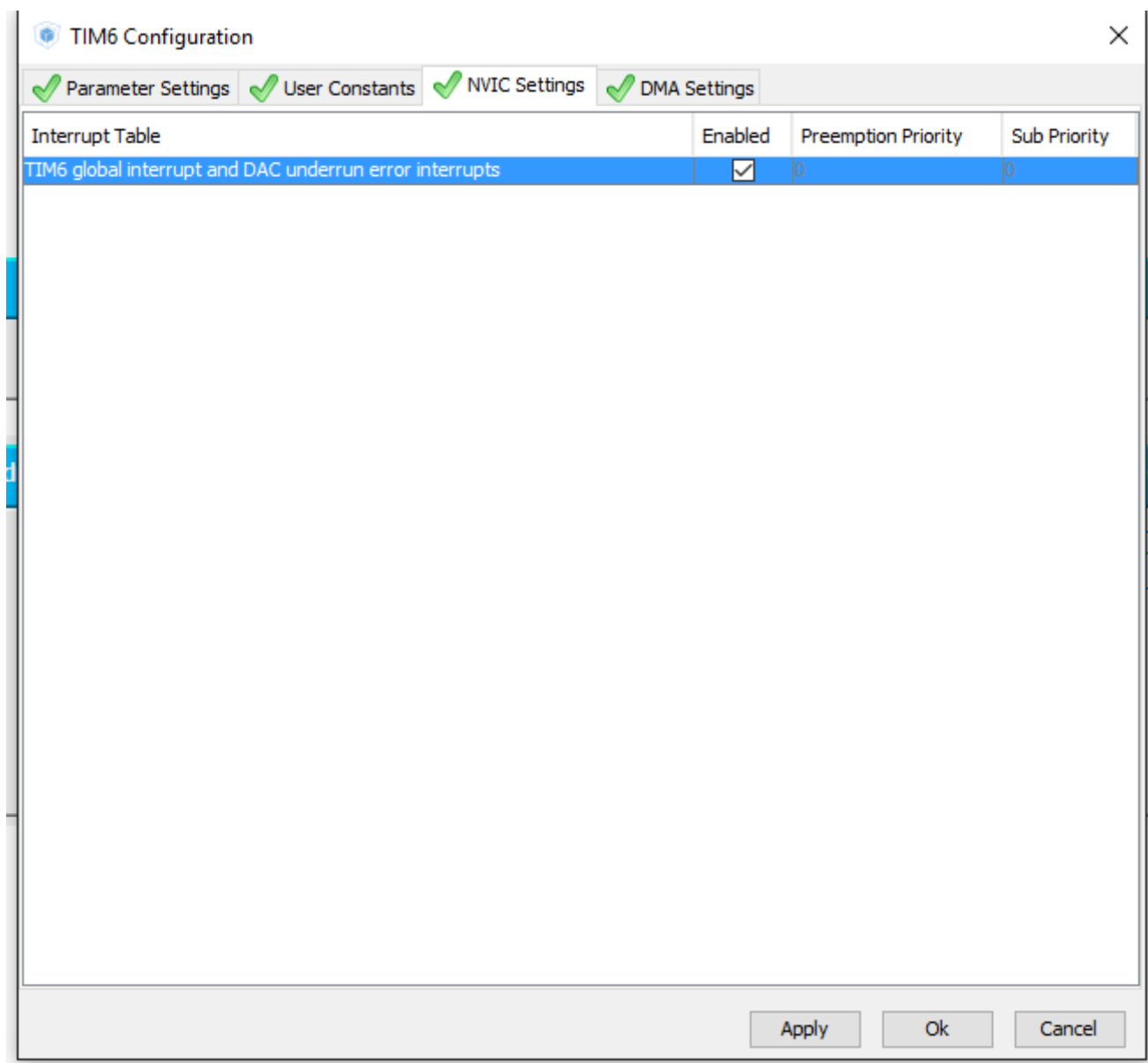
Trigger Event Selection

TIM_MasterOutputTrigger

Apply

Ok

Cancel



Генерируем проект. Скачаем исходники freemodbus-v1.5.0. Нам понадобится папка modbus. Поместим её в папку проекта в \Drivers.

Имя

- ✓ CMSIS
- ✓ modbus
- ✓ STM32F1xx_HAL_Driver

Туда же поместим папку port из freemodbus-v1.5.0\demo\BARE.

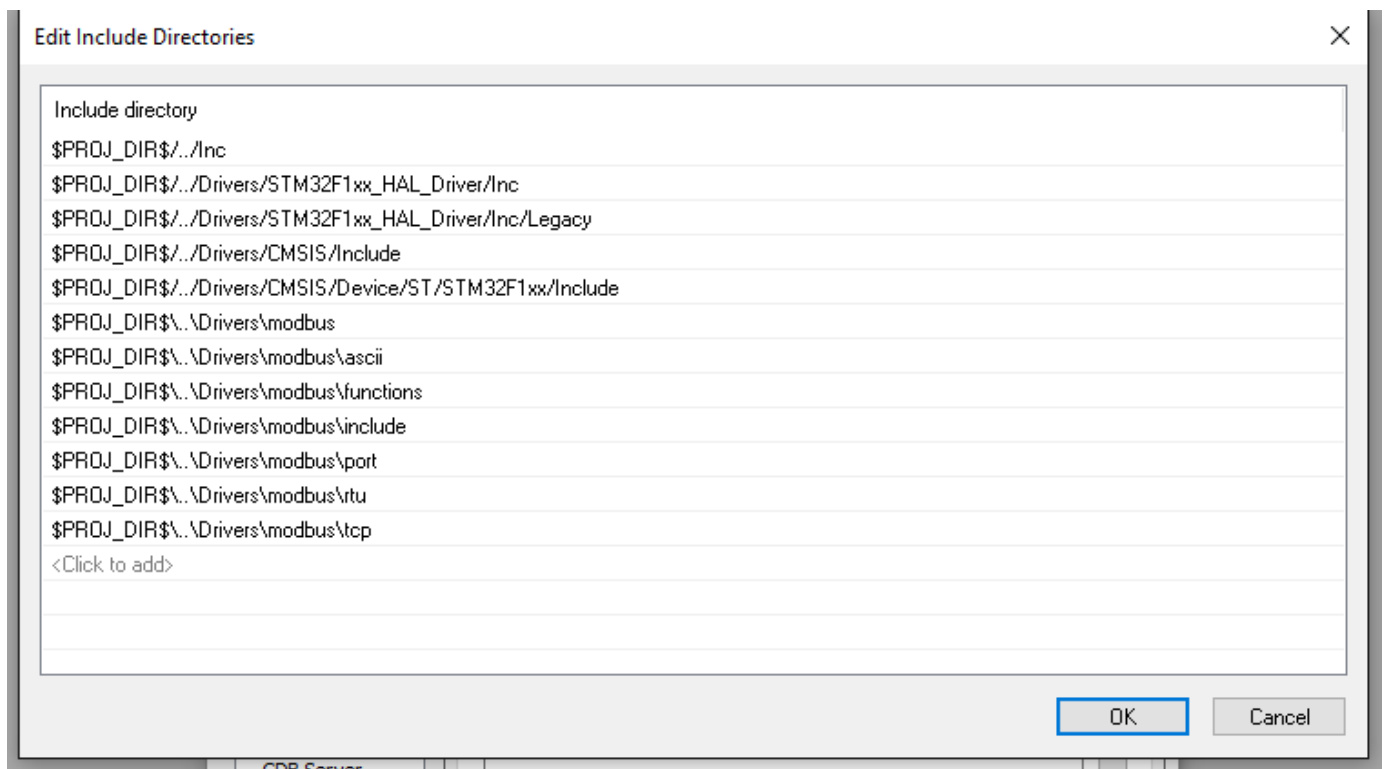
Имя

- ✓ ascii
- ✓ functions
- ✓ include
- ✓ port
- ✓ rtu
- ✓ tcp
- ✗ mb

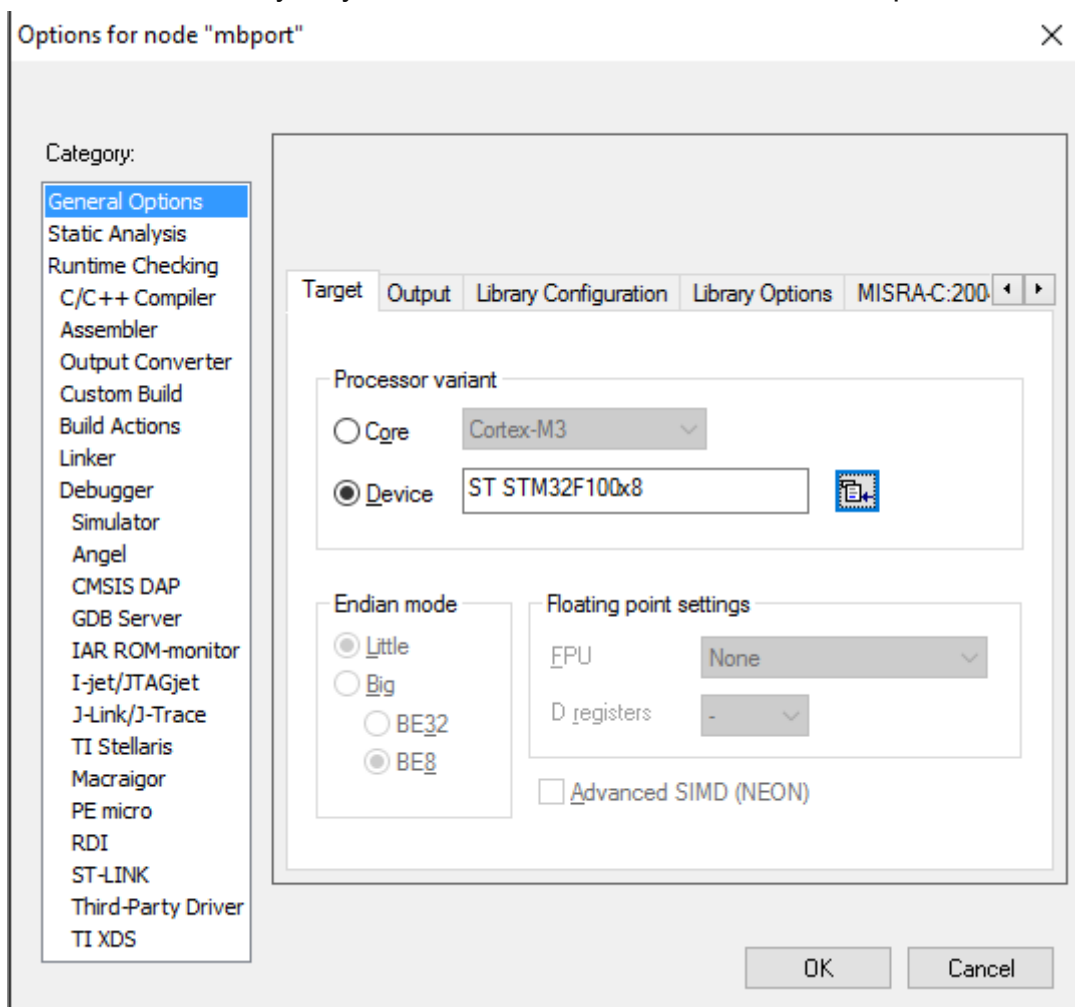
Откроем проект и прикрепим к нему только что скопированные исходники.

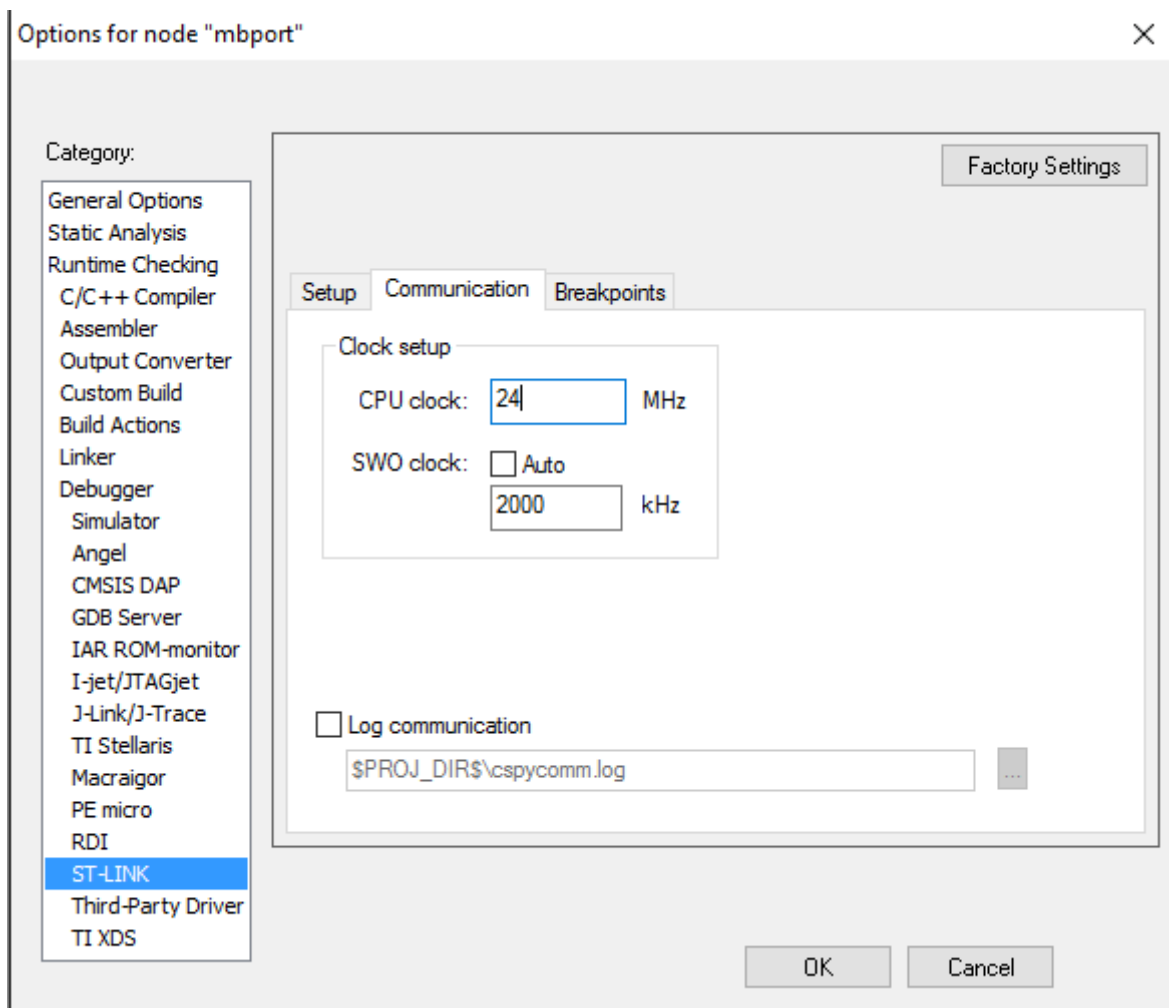
Files		
mbport - mbport	✓	
Application		*
Drivers		
CMSIS		*
modbus		
ascii		
mbascii.c		*
mbascii.h		
functions		
mbfunc coils.c		*
mbfuncdiag.c		*
mbfuncdisc.c		*
mbfunc holding.c		*
mbfuncinput.c		*
mbfunc other.c		*
mbutils.c		*
include		
mb.h		
mbconfig.h		
mbframe.h		
mbfunc.h		
mbport.h		
mbproto.h		
mbutils.h		
port		
port.h		
portevent.c		*
portserial.c		*
porttimer.c		*
rtu		
mbcrc.c		*
mbcrc.h		
mbrtu.c		*
mbrtu.h		
tcp		
mbtcp.c		*
mbtcp.h		
mb.c		*
STM32F1xx_HAL_Driver		*
Output		

Далее необходимо прописать пути к папкам в опциях проекта во вкладке Preprocessor.



STM QUBE почему то указал в качестве девайса none. Исправляем.





На этом этапе проект собирается, хоть и с предупреждениями. Перейдём непосредственно к портированию. Откроем port.h. Объявим функции, обеспечивающие атомарность операций. Сюда же вынесем объявления функций для UART.

```
void __critical_enter(void);
void __critical_exit(void);

#define ENTER_CRITICAL_SECTION( )    ( __critical_enter( ) )
#define EXIT_CRITICAL_SECTION( )    ( __critical_exit( ) )

void prvvUARTTxReadyISR( void );
void prvvUARTRxISR( void );
```

Дефинишн я написал в main().

```
static uint32_t lock_nesting_count = 0;
void __critical_enter(void)
{
    __disable_irq();
    ++lock_nesting_count;
```



```

}
void __critical_exit(void)
{
    /* Unlock interrupts only when we are exiting the outermost nested call. */
    --lock_nesting_count;
    if (lock_nesting_count == 0) {
        __enable_irq();
    }
}

```

Львиную часть таймера нам настроил Qube. Осталось лишь немного дописать в porttimer.c. Эта часть полностью написана на HAL и в лишних комментариях не нуждается.

▼ porttimer.c

```

/* ----- Platform includes -----*/
#include "port.h"
#include "stm32f1xx_hal.h"
/* ----- Modbus includes -----*/
#include "mb.h"
#include "mbport.h"

/* ----- static functions -----*/
static void prvvtIMERExpiredISR( void );
extern TIM_HandleTypeDef htim6;
uint16_t timeout = 0;
volatile uint16_t counter = 0;
/* ----- Start implementation -----*/
BOOL
xMBPortTimersInit( USHORT usTim1Timerout50us )
{
    timeout = usTim1Timerout50us;
    return TRUE;
}

void
vMBPortTimersEnable( )
{
    /* Enable the timer with the timeout passed to xMBPortTimersInit( ) */
    counter=0;
    HAL_TIM_Base_Start_IT(&htim6);
}

void
vMBPortTimersDisable( )
{

```

```

    /* Disable any pending timers. */
    HAL_TIM_Base_Stop_IT(&htim6);
}

/* Create an ISR which is called whenever the timer has expired. This function
 * must then call pxMBPortCBTimerExpired( ) to notify the protocol stack that
 * the timer has expired.
 */
static void prvvTIMERExpiredISR( void )
{
    ( void )pxMBPortCBTimerExpired( );
}

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    if(++counter)>=timeout)
    {
        prvvTIMERExpiredISR();
    }
}

```

Проверим, что всё идёт по плану. Проверим что тайминги совпадают ожиданиям.
 проверять буду дедовским методом, осциллографом. должен получиться импульс 1мс.
 Работает ли vMBPortTimersDisable — я проверять не буду =)

Временно напишем:

```

void
vMBPortTimersEnable( )
{
    /* Enable the timer with the timeout passed to xMBPortTimersInit( ) */
    HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_9);
    counter=0;
    HAL_TIM_Base_Start_IT(&htim6);
}

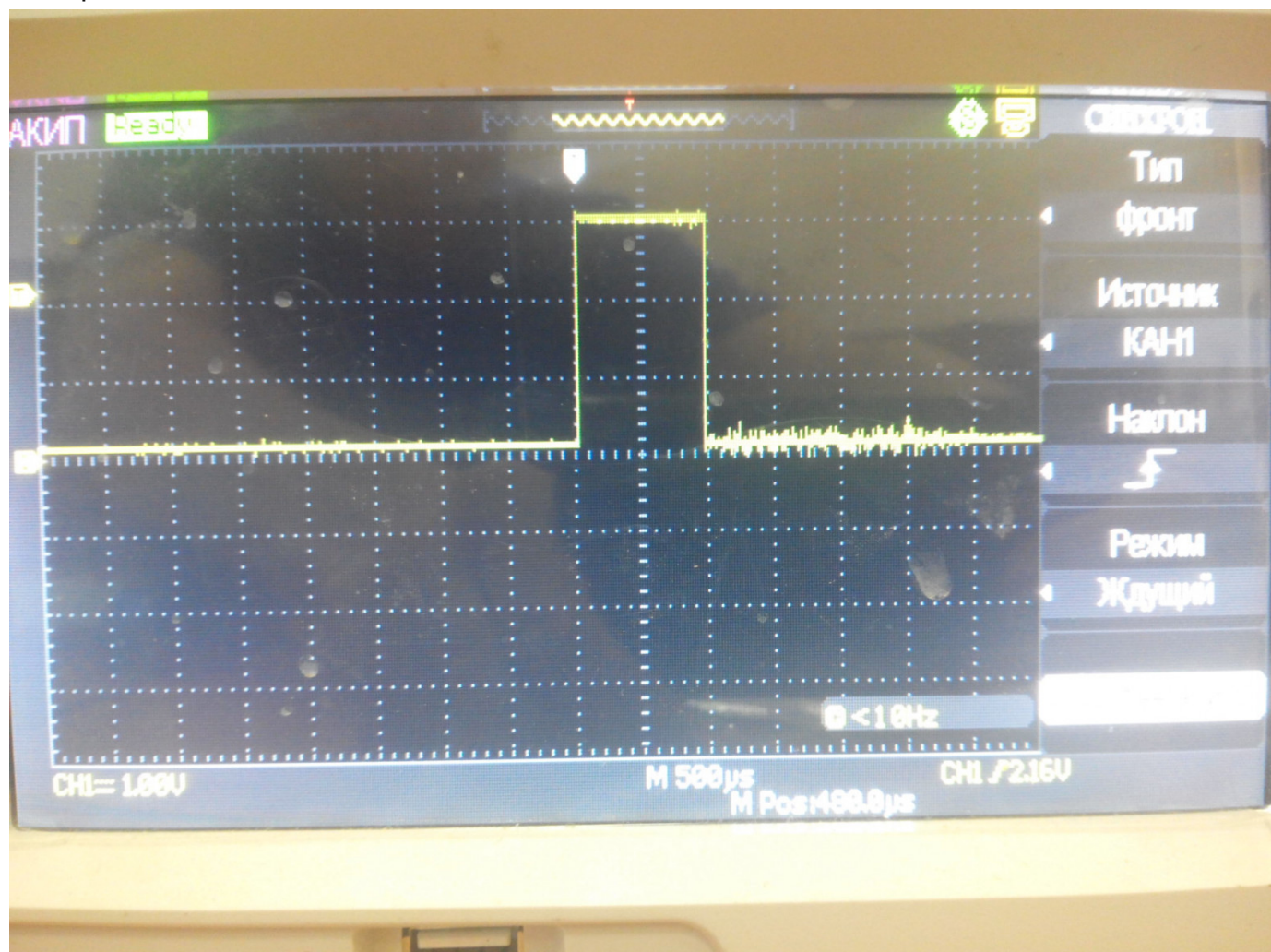
void
vMBPortTimersDisable( )
{
    /* Disable any pending timers. */
    HAL_GPIO_TogglePin(GPIOC, GPIO_PIN_9);
    HAL_TIM_Base_Stop_IT(&htim6);
}

```

И в main():

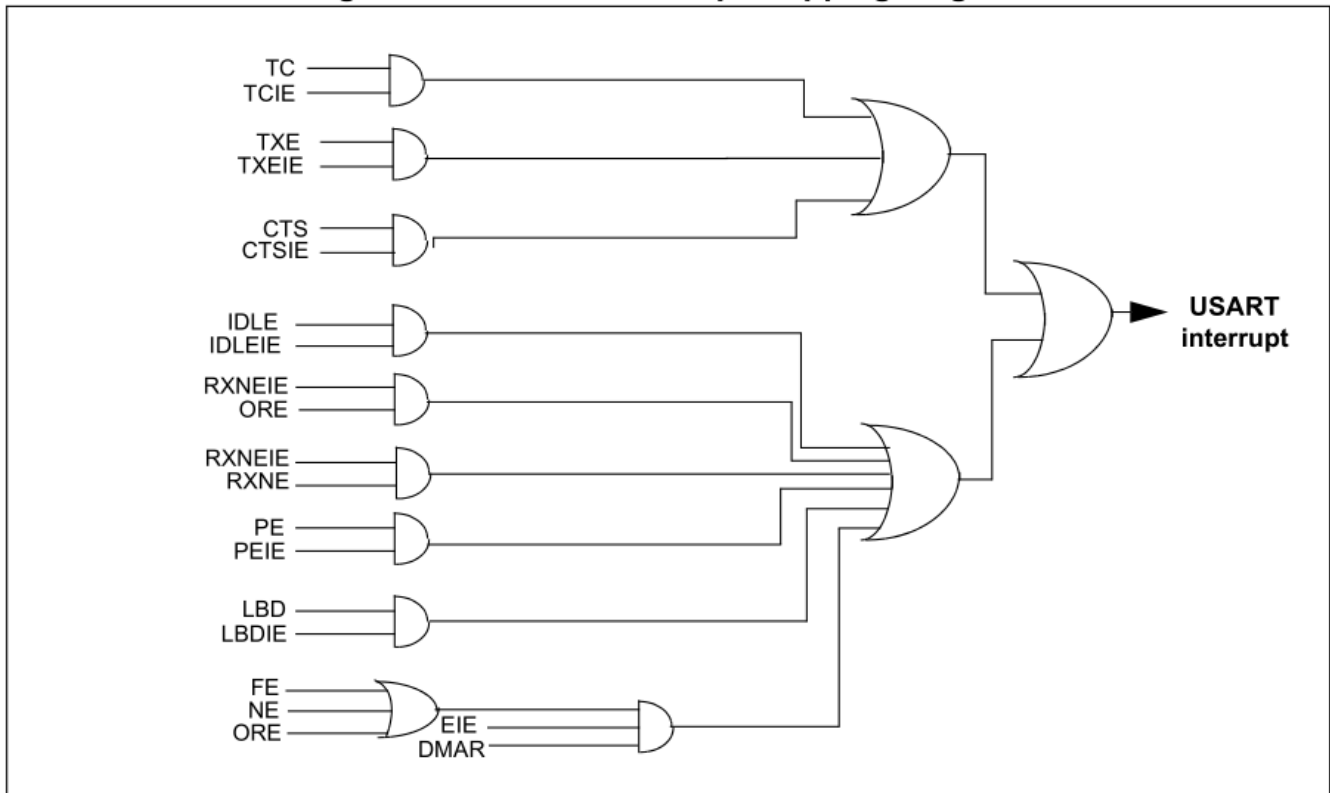
```
/* USER CODE BEGIN 2 */  
HAL_Delay(2000);  
xMBPortTimersInit( 20 );  
vMBPortTimersEnable( );
```

Смотрим:



Теперь самое интересное — это настройка UART =) Нужно начать с написания xMBPortSerialInit и vMBPortSerialEnable. Так как GetByte из библиотеки принимает char, то работу с 9битными сообщениями я исключаю в принципе. Для написания vMBPortSerialEnable обратимся к схеме прерываний от USART.

Figure 267. USART interrupt mapping diagram



Видно, что для разрешения прерывания по приему нужно включить RXNEIE: RXNE interrupt enable, а по событию передатчик готов — TXEIE: TXE interrupt enable.

Принятый байт лежит в регистре `huart_m->Instance->DR`. Запись в этот регистр вызывает передачу. Всё просто. Для удобства работы с USART, добавим `stm32f1xx_hal_uart.c` к проекту и задефайним `HAL_UART_MODULE_ENABLED`. Не буду писать много слов, а просто покажу, что в итоге внутри.

▼ [portserial.c](#)

```

#include "port.h"
#include "stm32f1xx_hal.h"
/* ----- Modbus includes -----*/
#include "mb.h"
#include "mbport.h"

/* ----- static functions -----*/
UART_HandleTypeDef huart_m;
HAL_StatusTypeDef USART_Init(UART_HandleTypeDef *huart);
void USART_MspInit(UART_HandleTypeDef* huart);
static void USART_SetConfig(UART_HandleTypeDef *huart);
/* ----- Start implementation -----*/
void
vMBPortSerialEnable( BOOL xRxEnable, BOOL xTxEnable )
{
    /* If xRxEnable enable serial receive interrupts. If xTxENable enable

```

```

    * transmitter empty interrupts.
    */
    if(xRxEnable)
    {
        __HAL_UART_ENABLE_IT(&huart_m, UART_IT_RXNE);
    }
    else
    {
        __HAL_UART_DISABLE_IT(&huart_m, UART_IT_RXNE);
    }

    if(xTxEnable)
    {
        __HAL_UART_ENABLE_IT(&huart_m, UART_IT_TXE);
    }
    else
    {
        __HAL_UART_DISABLE_IT(&huart_m, UART_IT_TXE);
    }
}

```

BOOL

```

xMBPortSerialInit( UCHAR ucPORT, ULONG ulBaudRate, UCHAR ucDataBits, eMBParity ePari
{
    switch (ucPORT)
    {
        case 0:
            huart_m.Instance = USART1;
            break;
        case 1:
            huart_m.Instance = USART2;
            break;
        case 2:
            huart_m.Instance = USART3;
            break;
        default:
            return FALSE;
    }
    huart_m.Init.BaudRate = ulBaudRate;
    switch (ucDataBits)
    {
        case 8:
            huart_m.Init.WordLength = UART_WORDLENGTH_8B;
            break;
        default:
            return FALSE;
    }
}

```

```

}
switch (eParity)
{
    case MB_PAR_NONE:
        huart_m.Init.Parity = UART_PARITY_NONE;
        break;
    case MB_PAR_EVEN:
        huart_m.Init.Parity = UART_PARITY_EVEN;
        break;
    case MB_PAR_ODD:
        huart_m.Init.Parity = UART_PARITY_ODD;
        break;
    default:
        return FALSE;
}
huart_m.Init.StopBits = UART_STOPBITS_1;
huart_m.Init.Mode = UART_MODE_TX_RX;
huart_m.Init.HwFlowCtl = UART_HWCONTROL_NONE;
huart_m.Init.OverSampling = UART_OVERSAMPLING_16;
return (HAL_OK == USART_Init(&huart_m));
}

```

BOOL

xMBPortSerialPutByte(CHAR ucByte)

```

{
    /* Put a byte in the UARTs transmit buffer. This function is called
     * by the protocol stack if pxMBFrameCBTransmitterEmpty( ) has been
     * called. */
    huart_m.Instance->DR=ucByte;
    return TRUE;
}

```

BOOL

xMBPortSerialGetByte(CHAR * pucByte)

```

{
    /* Return the byte in the UARTs receive buffer. This function is called
     * by the protocol stack after pxMBFrameCBByteReceived( ) has been called.
     */
    if(huart_m.Init.Parity == UART_PARITY_NONE)
    {
        *pucByte = (uint8_t)(huart_m.Instance->DR & (uint8_t)0x00FF);
    }
    else
    {
        *pucByte = (uint8_t)(huart_m.Instance->DR & (uint8_t)0x007F);
    }
}

```

```

    return TRUE;
}

/* Create an interrupt handler for the transmit buffer empty interrupt
 * (or an equivalent) for your target processor. This function should then
 * call pxMBFrameCBTransmitterEmpty( ) which tells the protocol stack that
 * a new character can be sent. The protocol stack will then call
 * xMBPortSerialPutByte( ) to send the character.
 */
void prvUARTTxReadyISR( void )
{
    pxMBFrameCBTransmitterEmpty( );
}

/* Create an interrupt handler for the receive interrupt for your target
 * processor. This function should then call pxMBFrameCBByteReceived( ). The
 * protocol stack will then call xMBPortSerialGetByte( ) to retrieve the
 * character.
 */
void prvUARTRxISR( void )
{
    pxMBFrameCBByteReceived( );
}

HAL_StatusTypeDef USART_Init(UART_HandleTypeDef *huart)
{
    /* Check the UART handle allocation */
    if(huart == NULL)
    {
        return HAL_ERROR;
    }

    /* Check the parameters */
    if(huart->Init.HwFlowCtl != UART_HWCONTROL_NONE)
    {
        /* The hardware flow control is available only for USART1, USART2, USART3 */
        assert_param(IS_UART_HWFLOW_INSTANCE(huart->Instance));
        assert_param(IS_UART_HARDWARE_FLOW_CONTROL(huart->Init.HwFlowCtl));
    }
    else
    {
        assert_param(IS_UART_INSTANCE(huart->Instance));
    }
    assert_param(IS_UART_WORD_LENGTH(huart->Init.WordLength));
    assert_param(IS_UART_OVERSAMPLING(huart->Init.OverSampling));
}

```

```

if(huart->State == HAL_UART_STATE_RESET)
{
    /* Allocate lock resource and initialize it */
    huart->Lock = HAL_UNLOCKED;

    /* Init the low level hardware */
    USART_MspInit(huart);
}

huart->State = HAL_UART_STATE_BUSY;

/* Disable the peripheral */
__HAL_UART_DISABLE(huart);

/* Set the UART Communication parameters */
USART_SetConfig(huart);

/* In asynchronous mode, the following bits must be kept cleared:
   - LINEN and CLKEN bits in the USART_CR2 register,
   - SCEN, HDSEL and IREN bits in the USART_CR3 register.*/
CLEAR_BIT(huart->Instance->CR2, (USART_CR2_LINEN | USART_CR2_CLKEN));
CLEAR_BIT(huart->Instance->CR3, (USART_CR3_SCEN | USART_CR3_HDSEL | USART_CR3_IREN));

/* Enable the peripheral */
__HAL_UART_ENABLE(huart);

/* Initialize the UART state */
huart->ErrorCode = HAL_UART_ERROR_NONE;
huart->State= HAL_UART_STATE_READY;

return HAL_OK;
}

void USART_MspInit(UART_HandleTypeDef* huart)
{
    GPIO_InitTypeDef GPIO_InitStructure;
    if(huart->Instance==USART1)
    {
        /* USER CODE BEGIN USART1_MspInit 0 */

        /* USER CODE END USART1_MspInit 0 */
        /* Peripheral clock enable */
        __HAL_RCC_USART1_CLK_ENABLE();

        /**USART1 GPIO Configuration
        PA9      -----> USART1_TX

```



```

PA10      -----> USART1_RX
*/

GPIO_InitStruct.Pin = GPIO_PIN_9;
GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

GPIO_InitStruct.Pin = GPIO_PIN_10;
GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
GPIO_InitStruct.Pull = GPIO_NOPULL;
HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

/* Peripheral interrupt init */
HAL_NVIC_SetPriority(USART1_IRQn, 0, 0);
HAL_NVIC_EnableIRQ(USART1_IRQn);
/* USER CODE BEGIN USART1_MspInit 1 */

/* USER CODE END USART1_MspInit 1 */
}
else if(huart->Instance==USART2)
{
/* USER CODE BEGIN USART2_MspInit 0 */

/* USER CODE END USART2_MspInit 0 */
/* Peripheral clock enable */
__HAL_RCC_USART2_CLK_ENABLE();

/**USART2 GPIO Configuration
PA2      -----> USART2_TX
PA3      -----> USART2_RX
*/
GPIO_InitStruct.Pin = GPIO_PIN_2;
GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

GPIO_InitStruct.Pin = GPIO_PIN_3;
GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
GPIO_InitStruct.Pull = GPIO_NOPULL;
HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

/* Peripheral interrupt init */
HAL_NVIC_SetPriority(USART2_IRQn, 0, 0);
HAL_NVIC_EnableIRQ(USART2_IRQn);
/* USER CODE BEGIN USART2_MspInit 1 */

```

```

/* USER CODE END USART2_MspInit 1 */
}
else if(huart->Instance==USART3)
{
/* USER CODE BEGIN USART3_MspInit 0 */

/* USER CODE END USART3_MspInit 0 */
/* Peripheral clock enable */
__HAL_RCC_USART3_CLK_ENABLE();

/**USART3 GPIO Configuration
PB10      -----> USART3_TX
PB11      -----> USART3_RX
*/
GPIO_InitStruct.Pin = GPIO_PIN_10;
GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

GPIO_InitStruct.Pin = GPIO_PIN_11;
GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
GPIO_InitStruct.Pull = GPIO_NOPULL;
HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

/* Peripheral interrupt init */
HAL_NVIC_SetPriority(USART3_IRQn, 0, 0);
HAL_NVIC_EnableIRQ(USART3_IRQn);
/* USER CODE BEGIN USART3_MspInit 1 */

/* USER CODE END USART3_MspInit 1 */
}
}

static void USART_SetConfig(UART_HandleTypeDef *huart)
{
    uint32_t tmpreg = 0x00;

    /* Check the parameters */
    assert_param(IS_UART_BAUDRATE(huart->Init.BaudRate));
    assert_param(IS_UART_STOPBITS(huart->Init.StopBits));
    assert_param(IS_UART_PARITY(huart->Init.Parity));
    assert_param(IS_UART_MODE(huart->Init.Mode));

    /*----- UART-associated USART registers setting : CR2 Configuration -----*/
    /* Configure the UART Stop Bits: Set STOP[13:12] bits according
    * to huart->Init.StopBits value */

```

```

MODIFY_REG(huart->Instance->CR2, USART_CR2_STOP, huart->Init.StopBits);

/*----- UART-associated USART registers setting : CR1 Configuration -----*/
/* Configure the UART Word Length, Parity and mode:
   Set the M bits according to huart->Init.WordLength value
   Set PCE and PS bits according to huart->Init.Parity value
   Set TE and RE bits according to huart->Init.Mode value */
tmpreg = (uint32_t)huart->Init.WordLength | huart->Init.Parity | huart->Init.Mode
MODIFY_REG(huart->Instance->CR1,
            (uint32_t)(USART_CR1_M | USART_CR1_PCE | USART_CR1_PS | USART_CR1_TE |
            tmpreg);

/*----- UART-associated USART registers setting : CR3 Configuration -----*/
/* Configure the UART HFC: Set CTSE and RTSE bits according to huart->Init.HwFlowC
MODIFY_REG(huart->Instance->CR3, (USART_CR3_RTSE | USART_CR3_CTSE), huart->Init.Hw

/*----- UART-associated USART registers setting : BRR Configuration -----*/
if((huart->Instance == USART1))
{
    huart->Instance->BRR = UART_BRR_SAMPLING16(HAL_RCC_GetPCLK2Freq(), huart->Init.B
}
else
{
    huart->Instance->BRR = UART_BRR_SAMPLING16(HAL_RCC_GetPCLK1Freq(), huart->Init.B
}
}

```

Теперь необходимо настроить прерывания в stm32f1xx_it.c.

```

void DINAR_UART_IRQHandler(UART_HandleTypeDef *huart)
{
    uint32_t tmp_flag = 0, tmp_it_source = 0;

    tmp_flag = __HAL_UART_GET_FLAG(huart, UART_FLAG_RXNE);
    tmp_it_source = __HAL_UART_GET_IT_SOURCE(huart, UART_IT_RXNE);
    /* UART in mode Receiver -----*/
    if((tmp_flag != RESET) && (tmp_it_source != RESET))
    {
        prvUARTRxISR( );
    }

    tmp_flag = __HAL_UART_GET_FLAG(huart, UART_FLAG_TXE);
    tmp_it_source = __HAL_UART_GET_IT_SOURCE(huart, UART_IT_TXE);
}

```

```

/* UART in mode Transmitter -----*/
if((tmp_flag != RESET) && (tmp_it_source != RESET))
{
    prvvUARTTxReadyISR( );
}
}

void USART1_IRQHandler(void)
{
    DINAR_UART_IRQHandler(&huart_m);
}

void USART2_IRQHandler(void)
{
    DINAR_UART_IRQHandler(&huart_m);
}

void USART3_IRQHandler(void)
{
    DINAR_UART_IRQHandler(&huart_m);
}

```

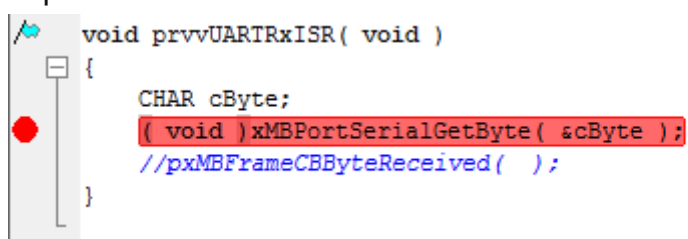
Как и в случае с таймером — надо убедиться что всё идёт по плану. Проверяем. Временно пропишем:

```

/* Initialize COM device 0 with 38400 baud, 8 data bits and no parity. */
if( xMBPortSerialInit( 0, 38400, 8, MB_PAR_NONE ) == FALSE )
{
}
else
{
    /* Enable the receiver. */
    vMBPortSerialEnable( TRUE, FALSE );
    /* Now block. Any character received should cause an interrupt now. */
    for( ;; );
}

```

Поставим брекпоинт и убедимся, что при приеме байта получаем прерывание. Я спал через hercules.



```

void prvvUARTRxISR( void )
{
    CHAR cByte;
    ( void )xMBPortSerialGetByte( &cByte );
    //pxMBFrameCByteReceived( );
}

```

```

void prvUARTRxISR( void )
{
    CHAR cByte;
    ( void )xMBPortSerialGetByte( &cByte );
    //pxMBFrameCBByteReceived( );
}

```

Также проверяем передатчик. Временно пропишем:

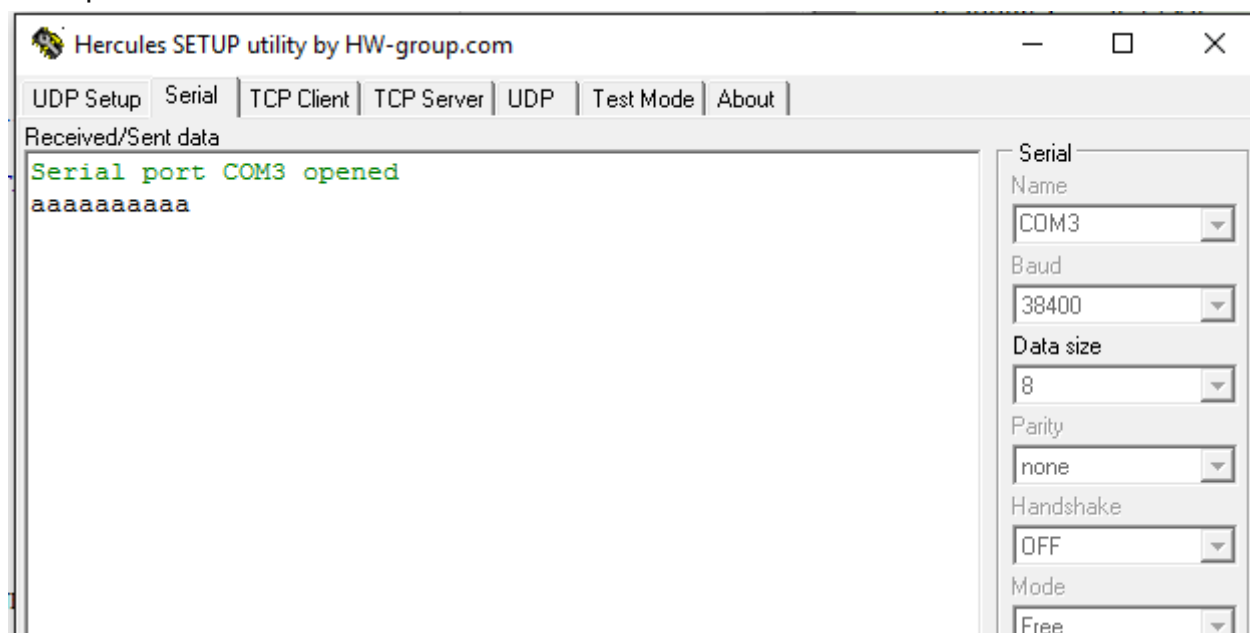
```

/*
static unsigned int uiCnt = 0;
void prvUARTTxReadyISR( void )
{
    //pxMBFrameCBTransmitterEmpty( );

    if( uiCnt++ < 10 )
    {
        ( void )xMBPortSerialPutByte( 'a' );
    }
    else
    {
        vMBPortSerialEnable( FALSE, FALSE );
    }
}
*/

```

Смотрим:



Попробуем теперь опросить наше устройство с помощью Modbus Poll.

▼ [Возьмем пример из Demo](#)

```

/* ----- Defines ----- */
#define REG_INPUT_START 1000
#define REG_INPUT_NREGS 4

/* ----- Static variables ----- */
static USHORT    usRegInputStart = REG_INPUT_START;
static USHORT    usRegInputBuf[REG_INPUT_NREGS];

/* USER CODE BEGIN 2 */
    eMBCErrorCode    eStatus;

    eStatus = eMBInit( MB_RTU, 0x11, 0, 38400, MB_PAR_NONE );
    eStatus = eMBEnable( );

/* USER CODE END 2 */

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

    /* USER CODE BEGIN 3 */
    ( void )    eMBPoll( );
    usRegInputBuf[0]++;
    }
/* USER CODE END 3 */
}

/* USER CODE BEGIN 4 */
eMBCErrorCode eMBRegInputCB( UCHAR * pucRegBuffer, USHORT usAddress, USHORT usNRegs )
{

eMBCErrorCode eMBRegHoldingCB( UCHAR * pucRegBuffer, USHORT usAddress, USHORT usNRegs,
    eMBRegisterMode eMode )
{

eMBCErrorCode eMBRegCoilsCB( UCHAR * pucRegBuffer, USHORT usAddress, USHORT usNCoils,
    eMBRegisterMode eMode )
{
    return MB_ENOREG;
}

eMBCErrorCode
eMBRegDiscreteCB( UCHAR * pucRegBuffer, USHORT usAddress, USHORT usNDiscrete )
{
    return MB_ENOREG;
}
/* USER CODE END 4 */

```

▼ [Запустим опрос](#)

Connection Setup

Connection
Serial Port

Serial Settings
Silicon Labs CP210x USB to UART Bridge
38400 Baud
8 Data bits
None Parity
1 Stop Bit
Advanced...

Mode
☒ RTU ☐ ASCII

Response Timeout
5000 [ms]

Delay Between Polls
10 [ms]

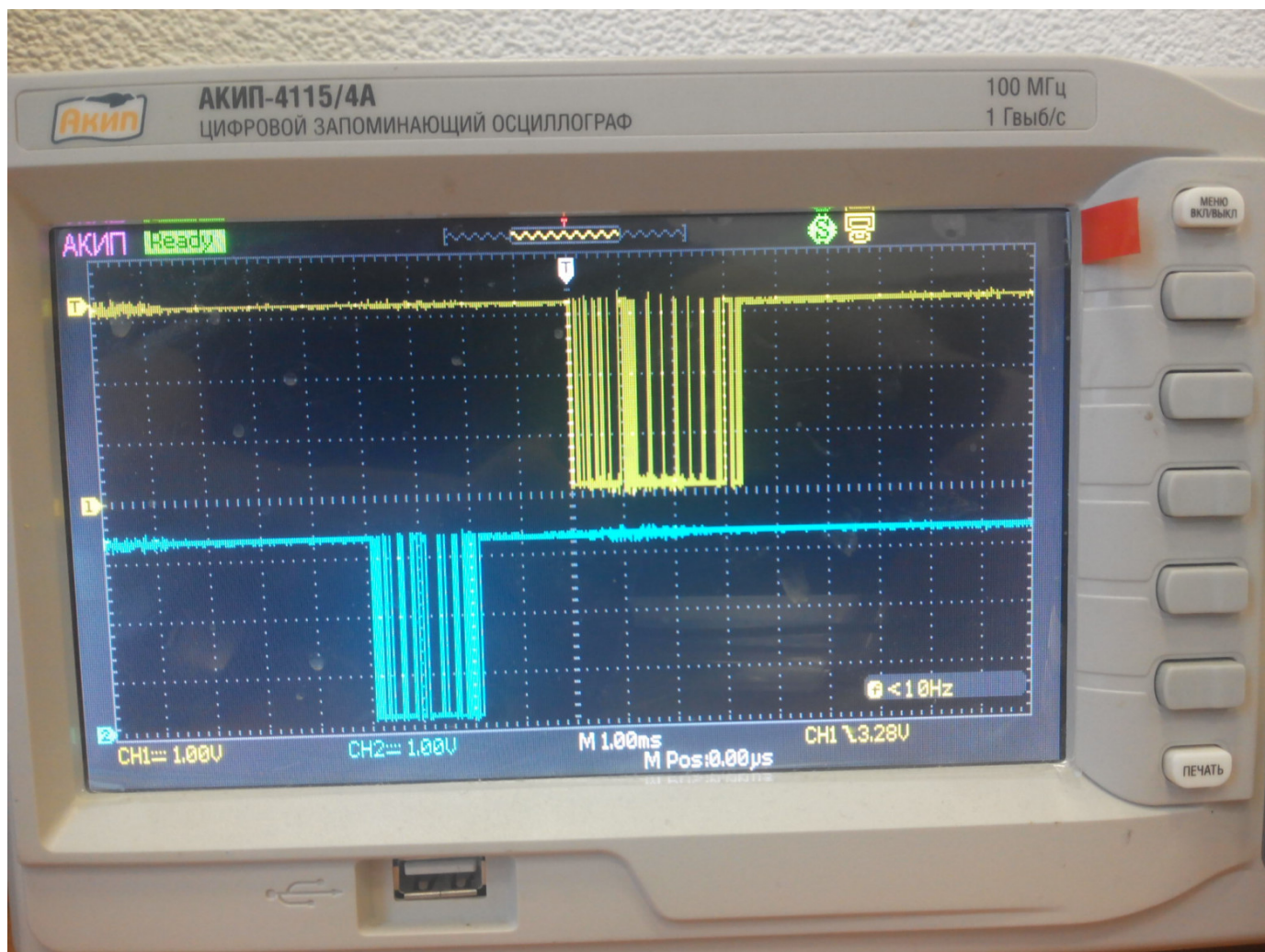
Remote Server
IP Address 127.0.0.1 Port 502 Connect Timeout 3000 [ms]

OK
Cancel

Mbpoll1

Tx = 13: Err = 0: ID = 17: F = 04: SR = 1000ms (DISABLED)

	Alias	00990	Alias	01000
0				0
1				0
2				0
3				
4				
5				
6				
7				
8				
9		-12630		



Работает! Надеюсь эта статья поможет начинающим, таким как я, в реализации этого простого но в то же время полезного протокола.

Теги: [stm32](#), [freemodbus](#), [stmcube](#), [iar](#)

Хабы: [Программирование микроконтроллеров](#)

Редакторский дайджест

Присылаем лучшие статьи раз в месяц



16

Карма

0

Рейтинг

Динар Талибуллин [@dinartal](#)

Инженер-программист



НЛО прилетело и опубликовало эту надпись здесь

**dinartal**

11.08.2016 в 23:36

Спасибо за замечание.



0

[Ответить](#)**d_ilyich**

11.08.2016 в 21:20

Рекомендую быть осторожным при работе с Coils. В FreeModbus-1.5.0 неправильно реализованы функции xxxGetBits и xxxSetBits (xxx — не помню точно префикс). Согласно спецификации Coils располагаются по возрастанию, а возвращаются по убыванию (старший бит в байте == старшему номеру Coil). А в FreeModbus они возвращаются как есть. Пример.

Исходные Coils: N0...N7 = 01001101

Требуется получить Coils с 4 по 7. Правильный результат должен быть следующим:

Байт состояния Coils: 00001011

Старшие 4 бита не используются, далее состояние Coil7...Coil4

А по версии FreeModbus получается

00001101 (или даже 11010000 - точно не помню, а проверить сейчас не могу)

Запись пока не проверял, но думаю, что аналогичная бяка.



+1

[Ответить](#)**dinartal**

18.08.2016 в 11:57

Странно. А как у вас колбэк функция реализована? Пользуетесь ли утилитой xMBUtilGetBits?

► [у меня так](#)



0

[Ответить](#)**d_ilyich**

18.08.2016 в 21:14

Я про GetBits и SetBits и написал. Они неправильные. Все колбэки и сопутствующий код брал исключительно из исходников FreeModbus.

Более того, эти самые функции устанавливают макс 8-бит за раз, но для этого используют переменную типа USHORT. Получается что-то вроде

```
usWordBuf = ucCoilsBuf[ usByteOffset ] << BITS_UCHAR;  
usWordBuf != ucCoilsBuf[ usByteOffset + 1 ];
```

Аналогично для записи. Т.е. можно выйти за границы. Возможно, ничего и не произойдёт, но как-то неуютно :)

0 Ответить



dinartal

18.08.2016 в 13:56



дополнил записью

0 Ответить



Только полноправные пользователи могут оставлять комментарии. [Войдите](#), пожалуйста.

ПОХОЖИЕ ПУБЛИКАЦИИ

2 июля в 00:38

STM32 Modular USB Composite device

+18

2.4K

65

7 +7

17 июня в 11:10

Виртуальный COM-порт на STM32 или как управлять контроллером через USB не привлекая внимания санитаров

+33

5.2K

101

9 +9

9 мая в 16:02

Библиотека для адресных светодиодов STM32

+21

7.4K

88

23 +23

МИНУТОЧКУ ВНИМАНИЯ

[Разместить](#)

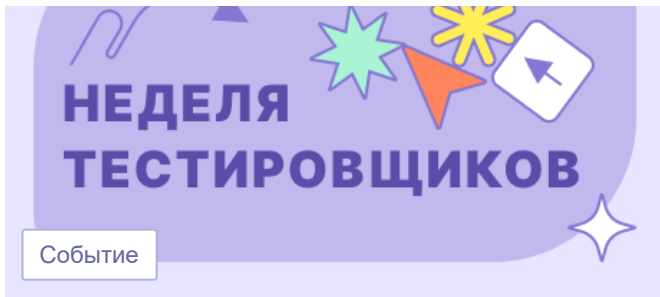


_ НА ХАБР КАРЬЕРЕ /

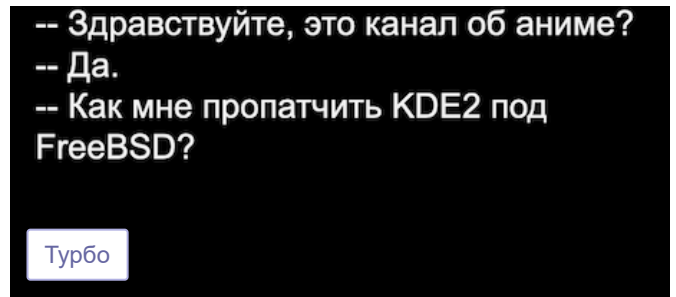
#42

2004-08-30 в 16:00

<Sashok>



Неделя тестировщиков на Хабр Карьере

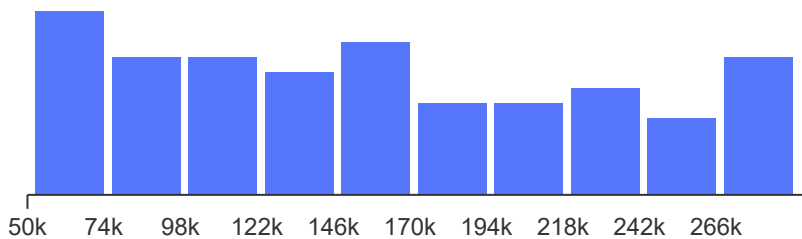


Мы спросили, нам ответили: каких тем ждут на Хабре

СРЕДНЯЯ ЗАРПЛАТА В IT

153 085 ₽/мес.

— средняя зарплата во всех IT-специализациях по данным из 11 900 анкет, за 1-ое пол. 2022 года. Проверьте «в рынке» ли ваша зарплата или нет!



[Проверить свою зарплату](#)

ЛУЧШИЕ ПУБЛИКАЦИИ ЗА СУТКИ

вчера в 17:20

Личный опыт выгорания

📌 +40

👁 14K

🔖 43

💬 47 +47

вчера в 17:00

Антиматерия и бариогенезис. Три причины, почему нет антивещества, но есть мы

📌 +18

👁 4.1K

🔖 25

💬 2 +2

сегодня в 08:32

Легальный способ копировать импортные изделия. Пошаговая инструкция

📌 +46

👁 6.2K

🔖 11

💬 45 +45

сегодня в 08:32

Blue-green deployment, canary release: рецепт приготовления безрисковых релизов

◆ +13

👁 1.1K

📖 15

💬 4 +4

вчера в 19:30

Микроконтроллеры Megawin серии MG32F02 на базе ядра Cortex-M0

◆ +12

👁 2.4K

📖 22

💬 13 +13

Хотим узнать, что вы думаете о своём месте работы (анонимно)

Опрос

ЧИТАЮТ СЕЙЧАС

Легальный способ копировать импортные изделия. Пошаговая инструкция

👁 6.4K

💬 15 +15

«Мы в одной лодке!»: Game Insight жалуется на обстоятельства и не хочет платить сотрудникам, имея чистую прибыль €4 млн

👁 4.7K

💬 15 +15

Личный опыт выгорания

👁 14K

💬 47 +47

Украинский изобретатель собрал рабочий «математический велосипед»

👁 31K

💬 51 +51

Завершился международный изоляционный эксперимент SIRIUS-21 длиной 240 суток

👁 19K

💬 33 +33

Ну-ка повтори: почему разработчики и дизайнеры не понимают друг друга

Мегапост

Ваш аккаунт

[Войти](#)

[Регистрация](#)

Разделы

[Публикации](#)

[Новости](#)

[Хабы](#)

[Компании](#)

[Авторы](#)

[Песочница](#)

Информация

[Устройство сайта](#)

[Для авторов](#)

[Для компаний](#)

[Документы](#)

[Соглашение](#)

[Конфиденциальность](#)

Услуги

[Корпоративный блог](#)

[Медийная реклама](#)

[Нативные проекты](#)

[Образовательные](#)

[программы](#)

[Мегапроекты](#)



[Настройка языка](#)

[Техническая поддержка](#)

[Вернуться на старую версию](#)

© 2006–2022, Habr