

Hello everyone!

I am currently working on the project that consist of two type of microcontrollers and I am trying to make a communication between these microcontrollers via rs485.

Stm32f103 includes gps and gsm module and I am able to send a message to the server usin GSM module configurations.I am planning to use this controller board as a reciever It should send gps data and the data via rs485 from stm32f030 microcontroller to the server.

In this case, I am using stm32f030 microcontroller to transmit ID data via rs485 to stm32f103 microcontroller.It is something like pairing process.I used LTC2850 rs485 transciever to convert TTL signal.

My problem is;

I have created stm32f030 microcontroller using hal drivers and defined rs485 communication inside(Unfortunately , I couldn't run it on CMSIS drivers).For transmission of data, I set rs485 "DE" polarity to high as below.Also I created 3 pin connector which is connected to RX,TX,GND of USART1.So I can see relevant data with serial connection cable through the my computer with Hercules(baudrate 9600).

```
uint8_t buffer[100];
int len;
int main(void)
{
    /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
    HAL_Init();

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

    /* Initialize all configured peripherals */
    MX_GPIO_Init();
    MX_USART1_UART_Init();
    HAL_RS485Ex_Init(&huart1,0,1,100);

    while (1)
    {
        sprintf(buffer,"3567");//\r\n
        len=strlen(buffer);

        if(__HAL_UART_GET_IT(&huart1, UART_IT_TXE)!= RESET){
            HAL_UART_Transmit_IT(&huart1, buffer, len);
            HAL_Delay(500);}

    }
}

static void MX_USART1_UART_Init(void)
{
    huart1.Instance = USART1;
    huart1.Init.BaudRate = 9600;
    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;
```

```

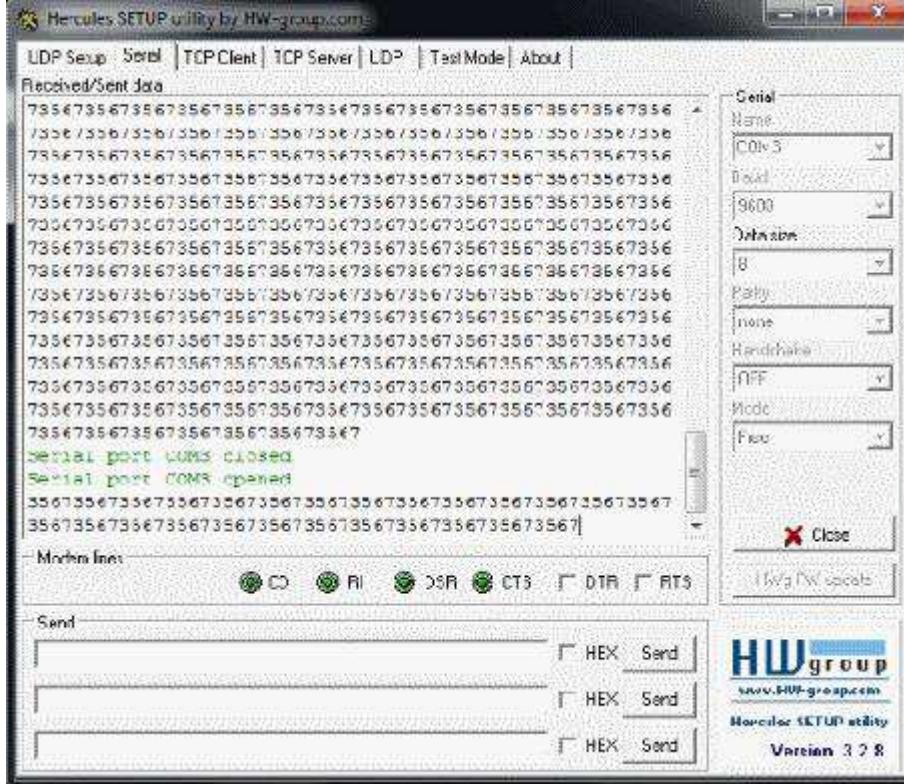
uart1.Init.OverSampling = UART_OVERSAMPLING_16;
uart1.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
uart1.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;

if (HAL_RS485Ex_Init(&uart1, UART_DE_POLARITY_HIGH, 0, 0) != HAL_OK)
{
    Error_Handler();
}

}

```

As you can see below, It sends data appropriately from usart1.



Then I programmed stm32f103(RS485 receiver) using CMSIS drivers. It consists of several interrupts for Usarts and I use GPIOB pin-0 for receiver rx-tx selection. There were no library for rs485 in CMSIS drivers and I set this "DE" pin-0 to "low" manually inside of the code. I have created Interrupt handler with nvic configuration for usart1.

But unfortunately, the problem is; I am just able to send first character of ID data which comes from stm32f030, So if the ID number is adjusted as "3567", I get only "3" one single character in stm32f103 side. Even if I delete this "DE" configuration in STM32F103, it continues to take this single character successfully. In this case, I would like to have this ID data completely. The relevant code is available below for Receiver side(stm32f103)

USART INTERRUPT HANDLER

```

void USART1_IRQHandler(void)
{
    if (USART_GetITStatus(USART1, USART_IT_RXNE) != RESET)
        //if(USART1->SR & USART_SR_RXNE)
    {

        // GPIO_SetBits(GPIOB,GPIO_Pin_0);
        GPIO_WriteBit(GPIOB,GPIO_Pin_0, Bit_RESET);
        char ch = rs485data.serialPort.getch();

        if(ch!=50){

```

```

rs485data.serialPort.rxTrailerbuffer[rs485data.serialPort.rxTrailerbufferReadCounter++]=ch;

    }

    if(rs485data.rs485detect== false){

        for (int i = 0;i<rs485data.serialPort.rxTrailerbufferReadCounter;i++){
            TrailerData[i]=rs485data.serialPort.rxTrailerbuffer[i];
            rs485data.serialPort.rxTrailerbuffer[i]=0;
        }

        rs485data.serialPort.rxTrailerbuffer[rs485data.serialPort.rxTrailerbufferReadCounter]=0;

        rs485data.rs485detect=true;
    }

    rs485data.serialPort.rxTrailerbufferReadCounter=0;

    //char ch = sp1.getch();
    USART_ClearITPendingBit(USART1, USART_IT_RXNE);
}
}
}

```

MAIN.cpp

```

//MAIN FILE
void mainInit()
{
    HSE_Init();

    SystemCoreClockUpdate();
    SysTick_Init();

    RTC_EXTI_Configuration();
    RTC_config();

    //NVIC_Configuration();
    // (*(int*)0xE000ED88))|=0x0F00000; // Floating Point donanimini aktiflestir. !!! Basimizin
    //derdi !!!
    // RCC->AHB1ENR |= 0x00000008;           // GPIOD donaniminin clock sinyalini uygulayalim

    RCC->APB2ENR |= ((1UL << 3) );

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOB, &GPIO_InitStructure);
    GPIO_WriteBit(GPIOB,GPIO_Pin_9, Bit_SET);

    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_7;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA,ENABLE);

    RCC->APB2ENR |= (RCC_APB2ENR_IOPAEN );
}

```

```
RCC->APB2ENR |= ((1UL << 3) );  
  
gsm = Telit_GL865(2,9600); //gsm configurations  
gsm.serialPort.enableRxInterrupt(); //gsm nvic configuration for interrupt(usart2)  
  
gps = Ublox_Max_M8Q(3,9600); //gps configuration  
gps.serialPort.enableRxInterrupt(); //gps nvic conf  
  
rs485data= RS485(1,9600); //RS485 CONFIGURATIONS(METHOD)  
rs485data.serialPort.enableRxInterrupt(); //MAKE NVIC CONFIGURATION FOR INTERRUPTS(USART1)  
GPIO_WriteBit(GPIOB,GPIO_Pin_0, Bit_RESET); //SET PIN_0 TO LOW FOR RECEIVING DATA FROM  
USART, DE PIN OF //RS485 TRANSC?EVER IS CONNECTED TO THIS PIN  
  
debugPort = SoftwareSerialPort(GPIOB, GPIO_Pin_9, 9600);  
  
RCC_APB1PeriphClockCmd(RCC_APB1Periph_PWR | RCC_APB1Periph_BKP, ENABLE);  
}  
  
int main()  
{  
  
    mainInit();  
  
    debugPort.printf("Start...\\r\\n");  
  
    while(1)  
    {  
        rs485data.rs485detect= false;  
        gps.foundData = false;  
        for(int wait = 0; wait<300; wait++) // 5 dakika beklet.  
        {  
            if (gps.foundData)  
            {  
                break;  
            }  
            delay(1000);  
        }  
        gsm.sendGPSData(gpsData,TrailerData); //Sending gps data and Id character from  
stm32f030 microcontroller(RS485)  
  
        for (int i=0;i<trailer_size;i++){  
            TrailerData[i]=0;  
        }  
  
        gps.foundData = false;  
        RTC_reConfigure();  
        debugPort.printf("Sleep mode active: %d mins. %d  
secs\\r\\n", __RTC_ALARM_M, __RTC_ALARM_S);  
        PWR_EnterSTOPMode(PWR_Regulator_LowPower, PWR_STOPEntry_WFE);  
        HSE_Init();  
        delay_ms(5000);  
        debugPort.printf("Sleep Mode Exited.\\r\\n");  
    }  
//    label11:
```

{}

MAIN.h

```
#ifndef MAIN_H_
#define MAIN_H_
#include <stdlib.h>      /* srand, rand */
#include <stdio.h>
#include <cstdlib>
#include <stdarg.h>
#include <string.h>
#include <stm32f10x.h>
#include <stm32f10x_flash.h>
#include <stm32f10x_gpio.h>

#include "Delay.h"
#include "Flash.h"
#include "RS485.h"

#include "SerialPort.h"
#include "Telit_GL865.h"
#include "Ublox_Max_M8Q.h"

#include "SoftwareSerialPort.h"
#include "RTC.h"

#define trailer_size 100
#define usart_size 100

Telit_GL865 gsm=Telit_GL865(2,9600); //= Telit_GL865(2,9600);
Ublox_Max_M8Q gps(3,9600); //= Ublox_Max_M8Q(3,9600);
RS485 rs485data(1,9600);
SoftwareSerialPort debugPort = SoftwareSerialPort(GPIOB, GPIO_Pin_9, 9600);

GPIO_InitTypeDef GPIO_InitStructure;

char teststr[100];
volatile char gpsData[100] = "_____WAITING_____";
volatile char TrailerData[100] = "-DATA";
#endif /* MAIN_H_ */
```

GPIO,NVIC AND USART CONFIGURATIONS

```
SerialPort::SerialPort(int SerialPortNo, uint32_t USART_BaudRate, uint16_t USART_Parity, uint16_t
USART_WordLength, uint16_t USART_StopBits)
{
    this->SerialPortNo = SerialPortNo;

    GPIO_InitTypeDef GPIO_InitStructure;

    USART_InitTypeDef USART_InitStructure;

    USART_InitStructure USART_BaudRate = USART_BaudRate;
    USART_InitStructure USART_WordLength = USART_WordLength;
    USART_InitStructure USART_StopBits = USART_StopBits;
    USART_InitStructure USART_Parity = USART_Parity ;
    USART_InitStructure USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
    USART_InitStructure USART_HardwareFlowControl =
USART_HardwareFlowControl_None;

    if (this->SerialPortNo == 1)
    {
        RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA | RCC_APB2Periph_GPIOB | RCC_APB2Periph_AFIO,
ENABLE); /* Enable GPIO clock */
        RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1, ENABLE); /* Enable UART clock */
    }
}
```

```
//Configure DE Rx-Tx selection GPIO
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
GPIO_InitStructure.GPIO_Pin=GPIO_Pin_0;
GPIO_InitStructure.GPIO_Speed=GPIO_Speed_50MHz;
GPIO_Init(GPIOB,&GPIO_InitStructure);

/* Configure the GPIO ports( USARTx Transmit and Receive Lines) */
/* Configure the USARTx_Tx as Alternate function Push-Pull */
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_Init(GPIOA, &GPIO_InitStructure);

/* Configure the USART1_Rx as input floating */
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10 ;
GPIO_Init(GPIOA, &GPIO_InitStructure);
//GPIO_PinRemapConfig(GPIO_Remap_USART1, ENABLE);

USART_Init(USART1, &USART_InitStructure);

USART_Cmd(USART1, ENABLE);

} else if (this->SerialPortNo == 2)
{
    //enable bus clocks
RCC_APB2PeriphClockCmd( RCC_APB2Periph_GPIOA | RCC_APB2Periph_AFIO, ENABLE);
RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART2,ENABLE);

//Set USART2 Tx (PA2) as AF push-pull
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2;
GPIO_Init(GPIOA, &GPIO_InitStructure);

//Set USART2 Rx (PA3) as input floating
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_3;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
GPIO_Init(GPIOA, &GPIO_InitStructure);

USART_Init(USART2, &USART_InitStructure);
USART_Cmd(USART2, ENABLE);
} else if (this->SerialPortNo == 3)
{
RCC_APB2PeriphClockCmd( RCC_APB2Periph_GPIOB | RCC_APB2Periph_AFIO, ENABLE);
RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART3,ENABLE);

//Set USART2 Tx (PB10) as AF push-pull
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;
GPIO_Init(GPIOB, &GPIO_InitStructure);

//Set USART2 Rx (B11) as input floating
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_11;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
GPIO_Init(GPIOB, &GPIO_InitStructure);
```

```

        USART_Init(USART3, &USART_InitStructure);
        USART_Cmd(USART3, ENABLE);
    }
    this->clearRxBuffer();
}

void SerialPort::enableRxInterrupt(void)
{
    NVIC_InitTypeDef NVIC_InitStructure;
    if (this->SerialPortNo == 1)
    {
        USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);

        NVIC_InitStructure.NVIC IRQChannel = USART1_IRQn;
        NVIC_InitStructure.NVIC IRQChannelPreemptionPriority = 0;
        NVIC_InitStructure.NVIC IRQChannelSubPriority = 0;
        NVIC_InitStructure.NVIC IRQChannelCmd = ENABLE;
        NVIC_Init(&NVIC_InitStructure);

    } else if (this->SerialPortNo == 2)
    {
        USART_ITConfig(USART2, USART_IT_RXNE, ENABLE);

        NVIC_InitStructure.NVIC IRQChannel = USART2_IRQn;
        NVIC_InitStructure.NVIC IRQChannelPreemptionPriority = 1;
        NVIC_InitStructure.NVIC IRQChannelSubPriority = 0;
        NVIC_InitStructure.NVIC IRQChannelCmd = ENABLE;
        NVIC_Init(&NVIC_InitStructure);

    } else if (this->SerialPortNo == 3)
    {
        USART_ITConfig(USART3, USART_IT_RXNE, ENABLE);

        NVIC_InitStructure.NVIC IRQChannel = USART3_IRQn;
        NVIC_InitStructure.NVIC IRQChannelPreemptionPriority = 0;
        NVIC_InitStructure.NVIC IRQChannelSubPriority = 0;
        NVIC_InitStructure.NVIC IRQChannelCmd = ENABLE;
        NVIC_Init(&NVIC_InitStructure);
    }
}

```

Receive method

```

char SerialPort::getch()
{
    char ch;
    if (this->SerialPortNo == 1)
    {
        while((USART1->SR & USART_SR_RXNE) == RESET);
        ch = USART1->DR;
    } else if (this->SerialPortNo == 2)
    {
        while(USART_GetFlagStatus(USART2, USART_FLAG_RXNE) == RESET);
        ch = USART_ReceiveData(USART2);
    } else if (this->SerialPortNo == 3)
    {
        while(USART_GetFlagStatus(USART3, USART_FLAG_RXNE) == RESET);
        ch = USART_ReceiveData(USART3);
    }
}

```

```
    }
    return ch;
}
```

rs485 enable ?nterrupt method

```
#include "RS485.h"

RS485::RS485(){}
    RS485::RS485(int SerialPortNo, uint32_t USART_BaudRate){
        this->serialPort =
SerialPort(SerialPortNo,USART_BaudRate,USART_Parity_No,USART_WordLength_8b,USART_StopBits_1);
        this->serialPort.enableRxInterrupt();
        this->rs485detect = false;

    }
```