7.Communication

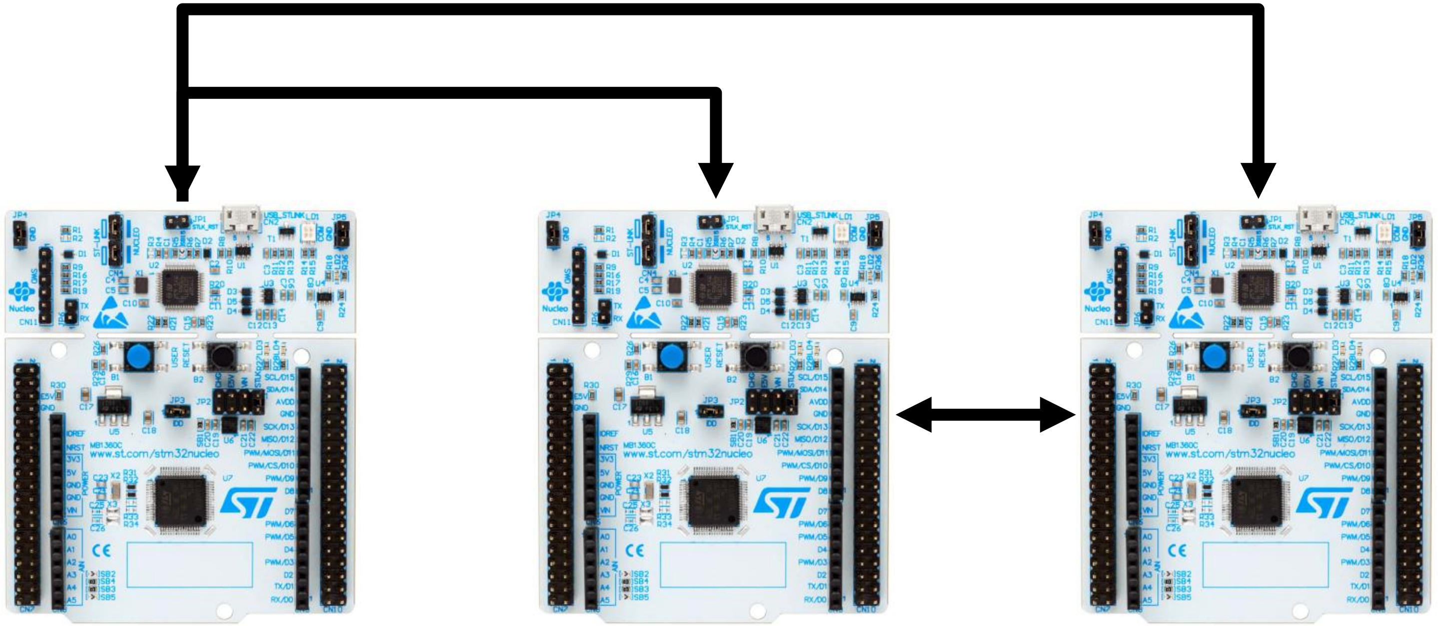
DR.SOMSIN THONGKRAIRAT


REPLY ME!
ACK
START BIT
NO, I'M MASTER
imgflip.com

Connectivity

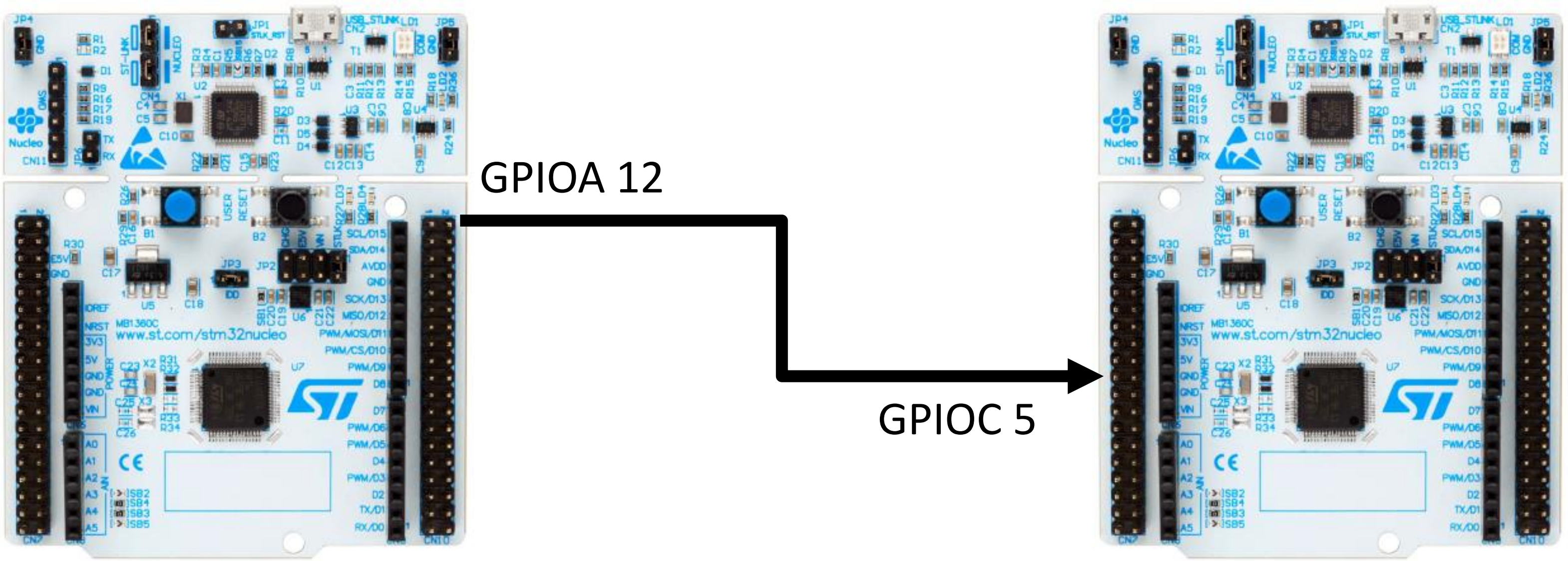
When two or more MCU are in the same system.

Communication are need!



Simplest communication

Send 1 bit or 1 state at a time (GPIO)



Type of communication

- Categorize by direction
  - Simplex
  - Half duplex
  - Full duplex

- Categorize by topology
  - Master-slave
  - One-to-one
  - broadcast

- Categorize by Clock
  - Synchronous (Time base)
  - Asynchronous (clock base)

LAN
GSP
UART
USART
SPI
I2C
One-wire-Serial

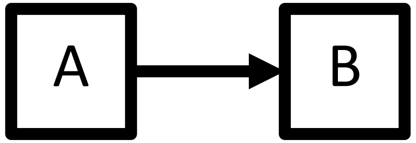
?

Categorize by direction

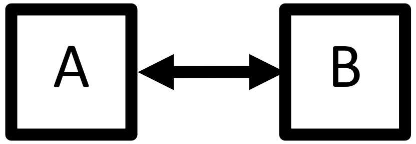
Simplex -> ส่งข้อมูลทางเดียว ไม่สามารถส่งข้อมูลกลับได้

Full duplex -> ส่งข้อไปกลับพร้อมกันได้

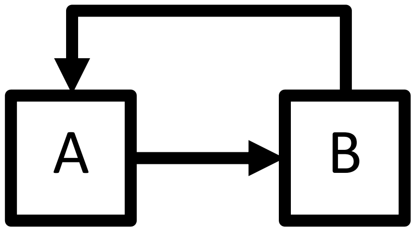
Half duplex -> ส่งข้อไปกลับ แต่ทำพร้อมกันไม่ได้



Simplex



Half-Duplex



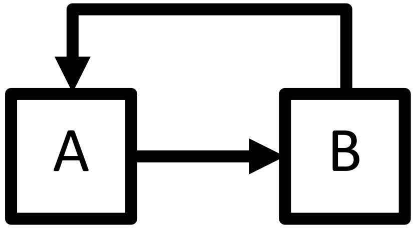
Full-Duplex

Categorize by topology

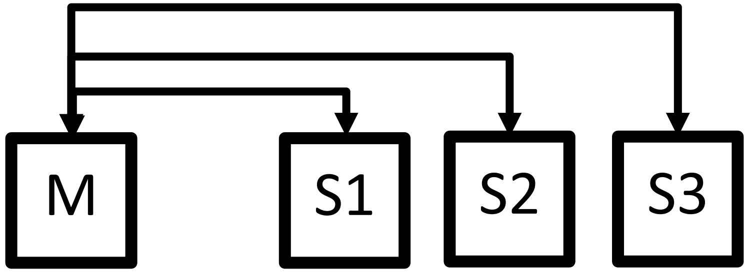
One-to-one -> only 2 MCU in network

Master-slave -> Master device control communication BUS

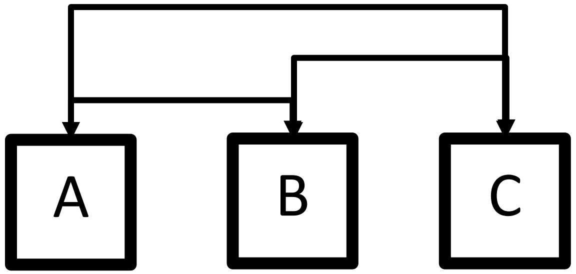
BUS -> every device do same behavior



One to One



Master slave



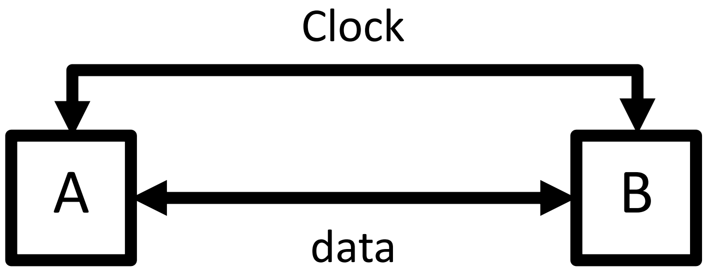
BUS

Categorize by Clock

Synchronous (Time base) -> speed define by time (pre-define)

Asynchronous (clock base) -> speed define by clock signal (data send with clock signal)

Asynchronous



Synchronous

STM availability

Serial -> UART -> Asynchronous , FULL-Duplex

SPI -> USART -> Synchronous , Master slave, Half-Duplex

I2C -> Two Wire -> Synchronous , Master slave , Full-Duplex

CAN -> CAN interface -> Asynchronous , BUS, Half-Duplex

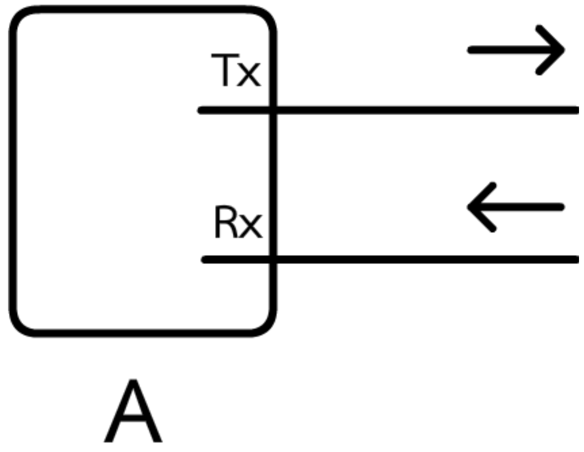
UCPD -> Asynchronized

ETH -> LAN

Serial communication

send serial data through media directory

‘H’ -> 0b01001000 (MSB)



Tx = Transmitter (ตัวส่ง)

Rx = Receiver (ตัวรับ)

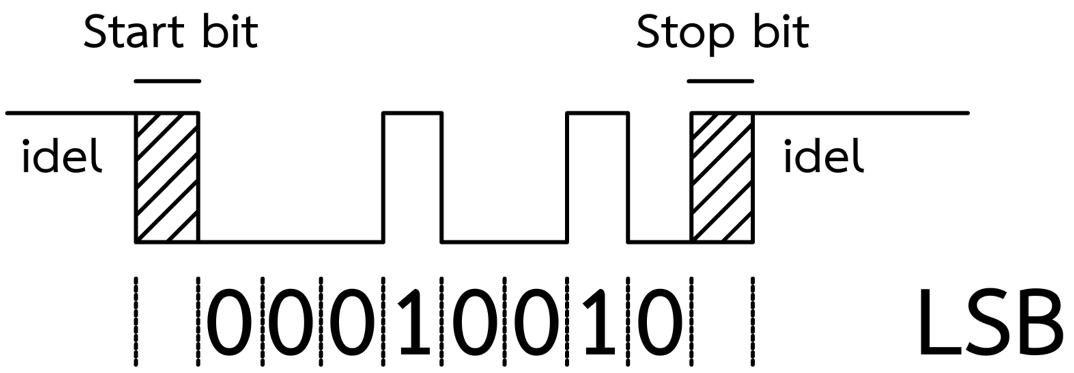
Serial communication data

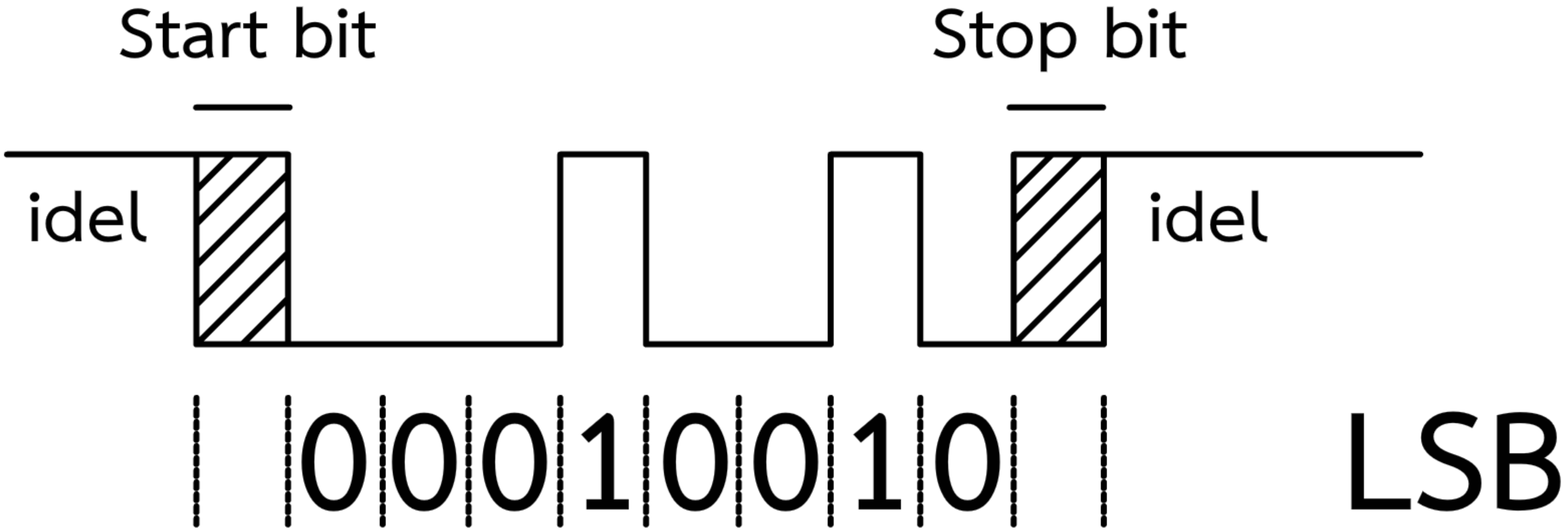
LSB (less significant bit first) format (Default)

Start bit -> start signal to receive incoming data (logic 0 by default)

Stop bit -> mark end of 1 data (e.g., end of byte) (logic 1 by default)

Parity bit -> check sum (transmit error checking)



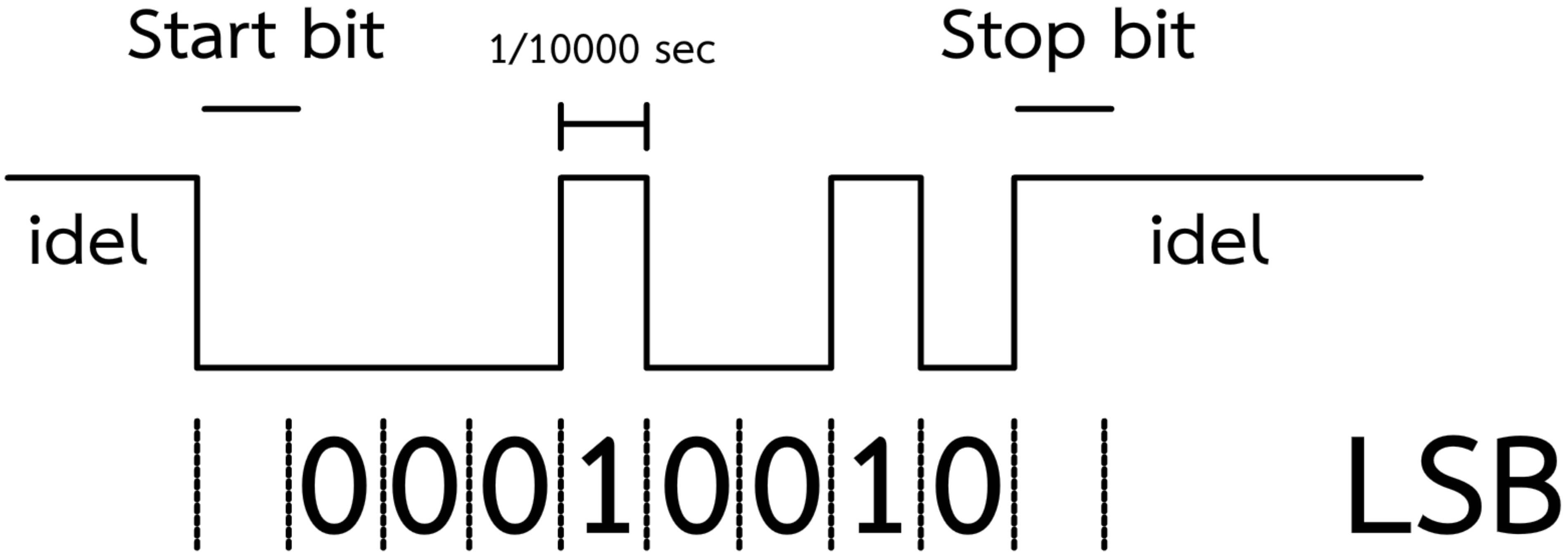


‘H’ -> 0b01001000 (MSB)

Serial communication

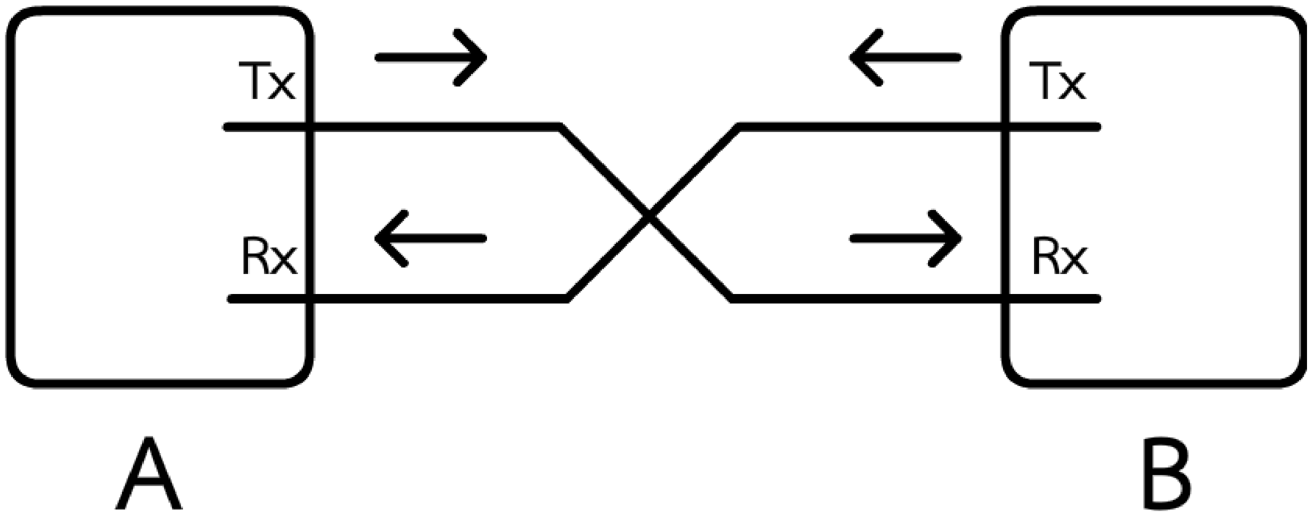
'H' -> 0b01001000 (MSB)

Baud rate = 10000 bps, Start bit = 1 baud, Stop bit = 1 baud



UART

Full duplex , One-to-one , Asynchronous

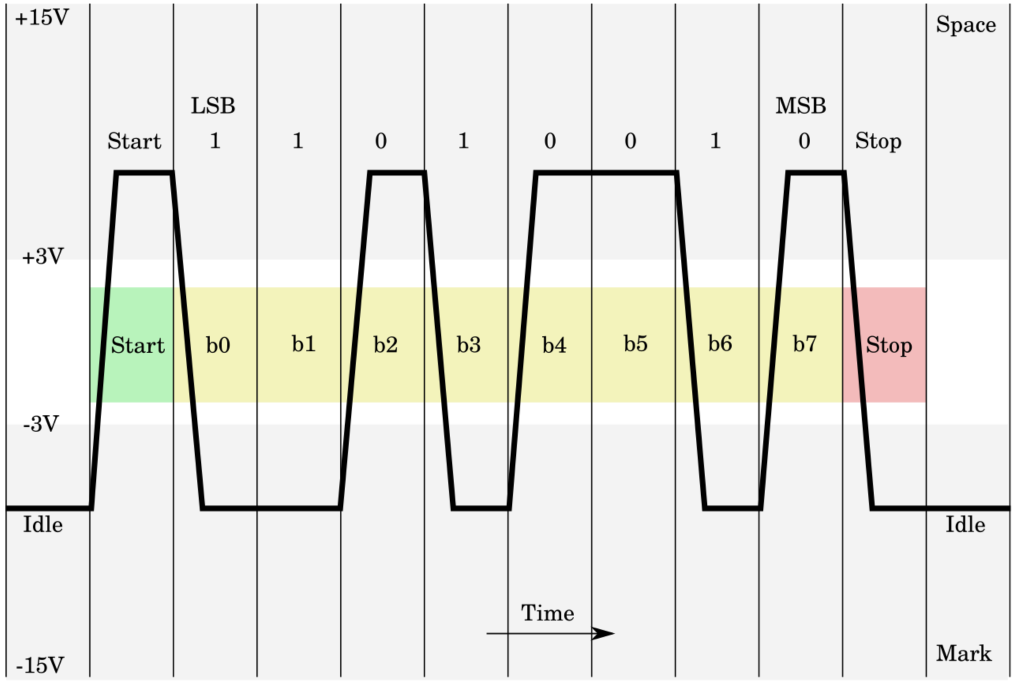


Rx connect to Tx
Tx connect to Rx
Same baud rate

Serial (UART) variant

RS232 -> convert logic level to positive and negative (-5V -> 1 , 5V -> 0) or (-15V -> 1 , 15V -> 0)

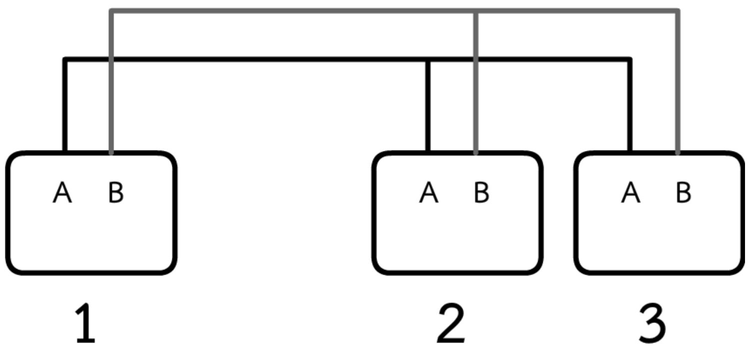
Improve robustness (longer and faster)

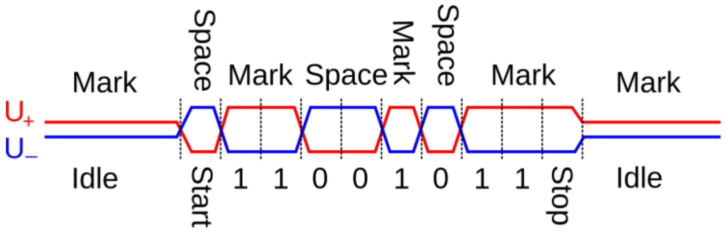


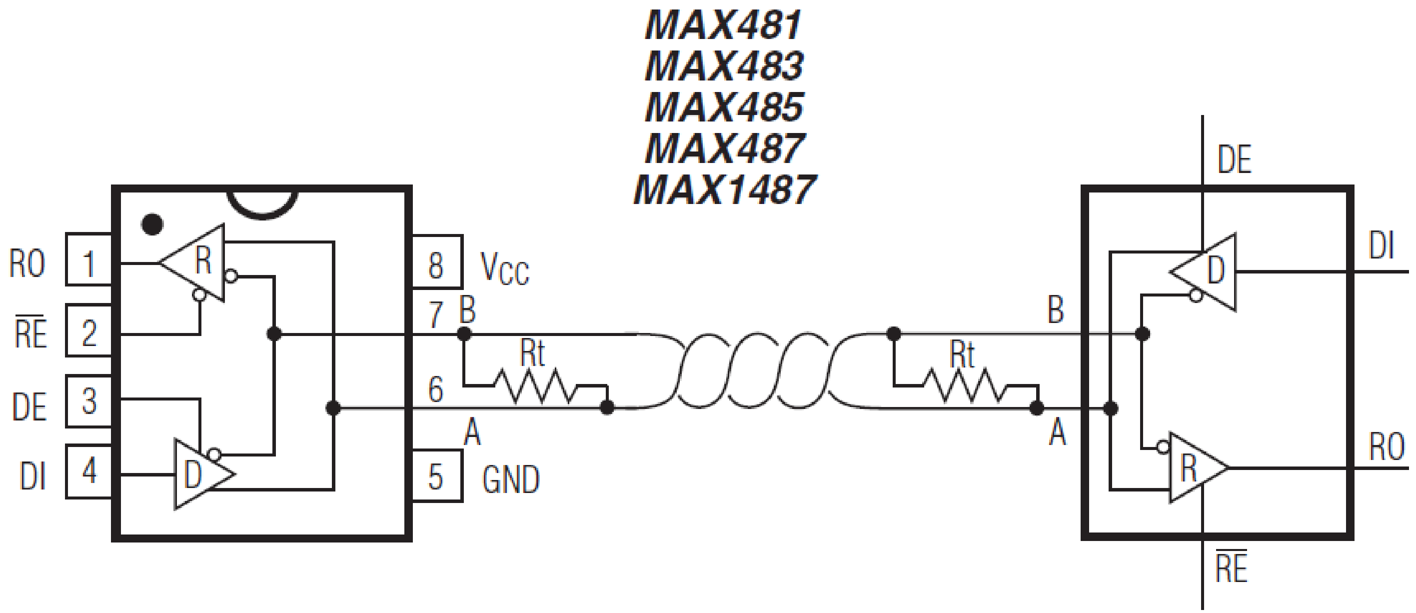
https://en.wikipedia.org/wiki/RS-232#/media/File:Rs232_oscilloscope_trace.svg

Serial (UART) variant

RS485-> using differential logic () but Half duplex , BUS

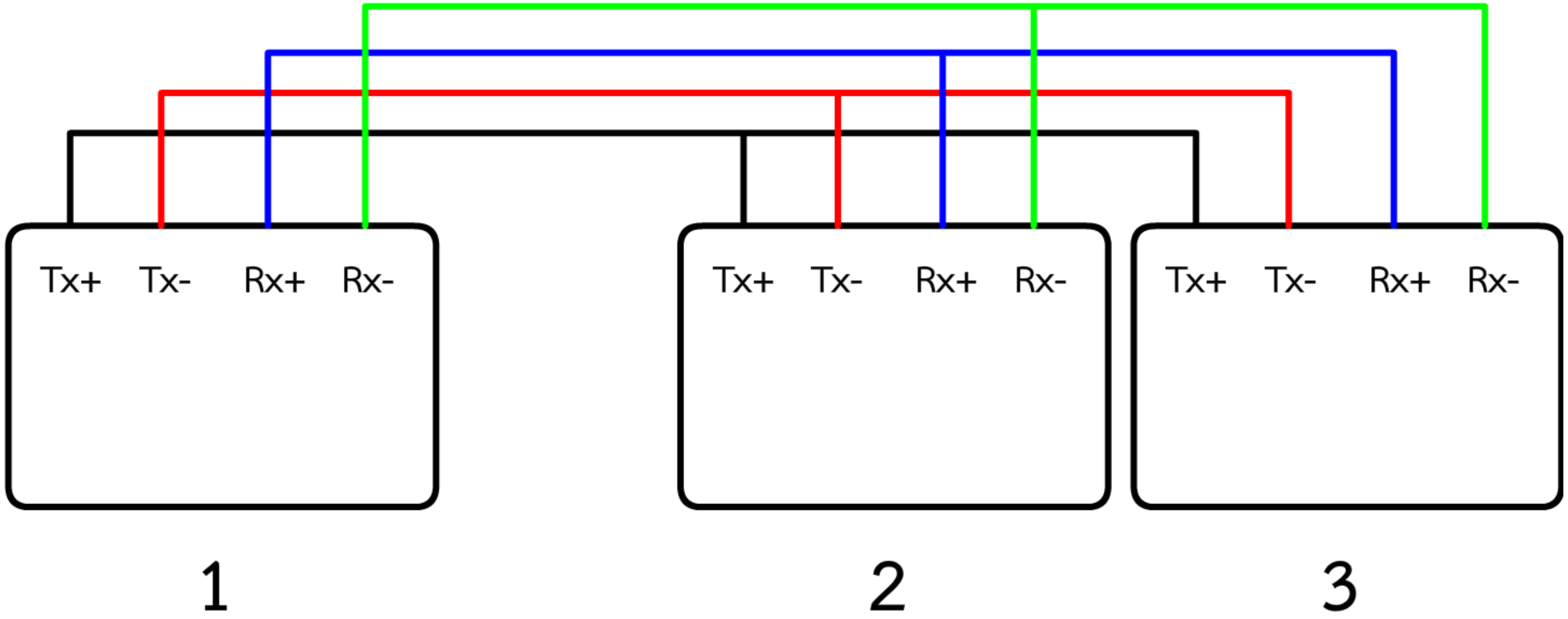






Serial (UART) variant

RS422-> Full duplex version of RS485 (add 1 pair of wire)

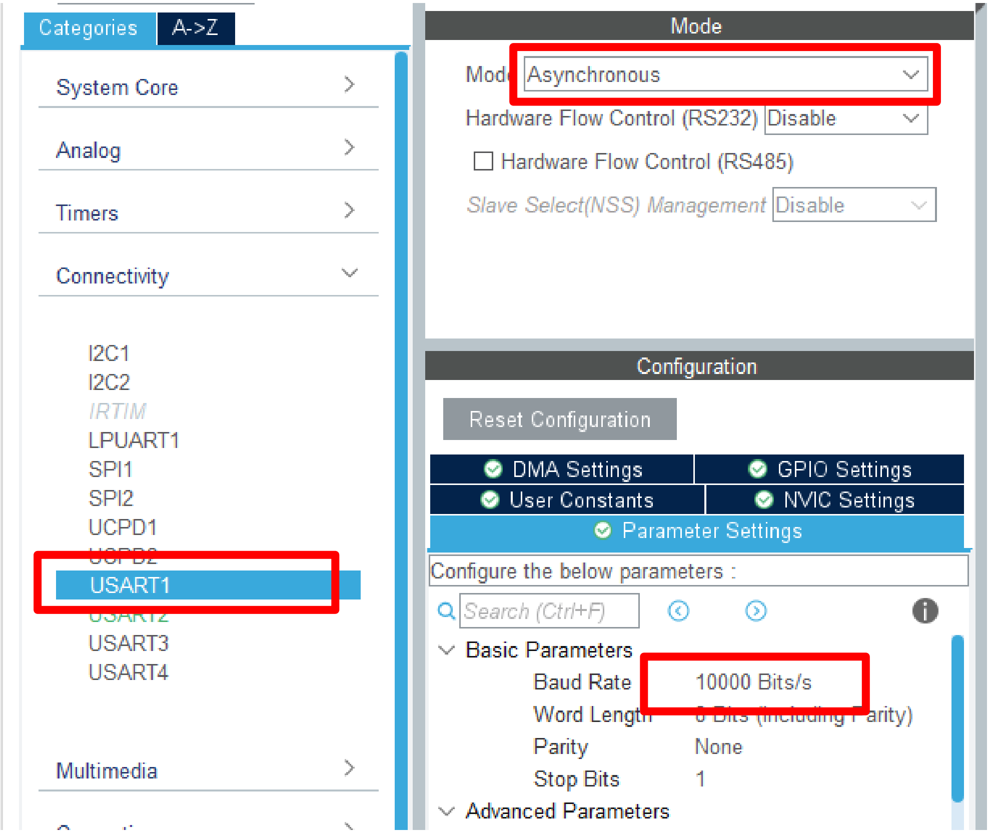


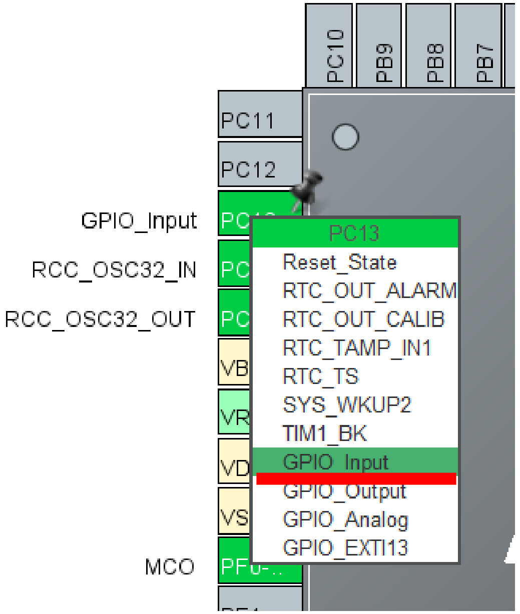
https://en.wikipedia.org/wiki/RS-485#/media/File:RS-485_waveform.svg

*MAX485 datasheet

Communication programming sender

Sender is easy because device can send when every they want! (LAB1)





Set PC 13 to Input

Setup USART 1 in Asynchronous mode

Set baud rate to 10000 Bps (8bit)

LAB1

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

    /* USER CODE BEGIN 3 */
      unsigned char data[10] = "A"; // = 65 DEC , 0X41 , 0b01000001
      while(HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13) == 0){
          HAL_UART_Transmit(&huart1, data, 1, 1000);
          data[0]++;
          HAL_Delay(3000);
      }
      HAL_Delay(1000);
  }
  /* USER CODE END 3 */
}
```

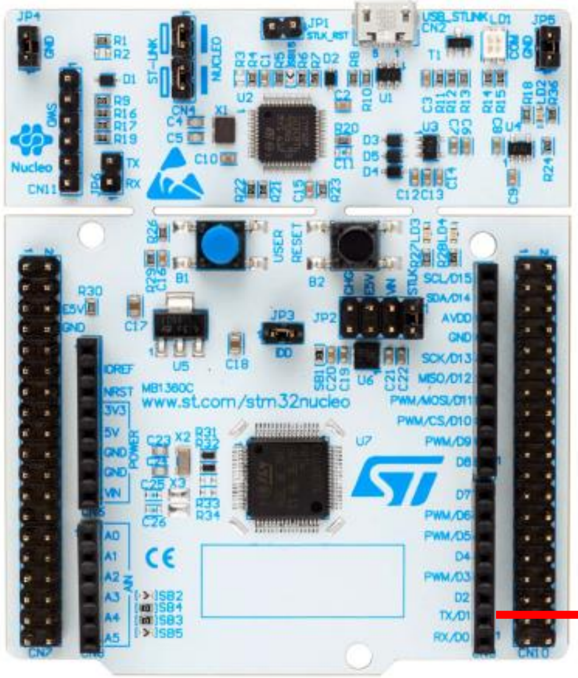
```c
/* USER CODE BEGIN 3 */
unsigned char data[10] = "A"; // = 65 DEC , 0X41 , 0b01000001
while(HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13) == 0){
HAL_UART_Transmit(&huart1, data, 1, 1000);
data[0]++;
HAL_Delay(3000);
}
HAL_Delay(1000);
}
/* USER CODE END 3 */
```

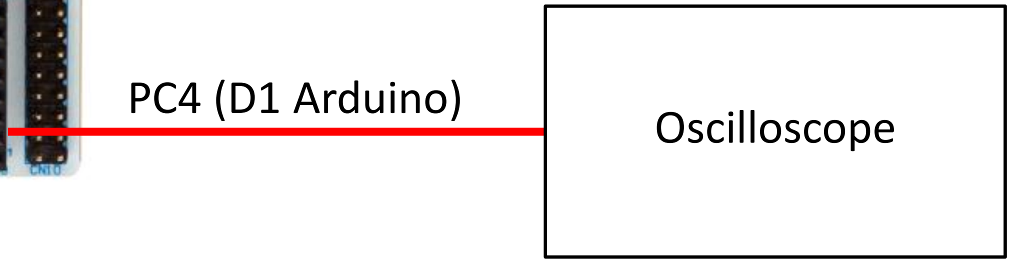
Send data when button pressed and increase data 1 when hold button

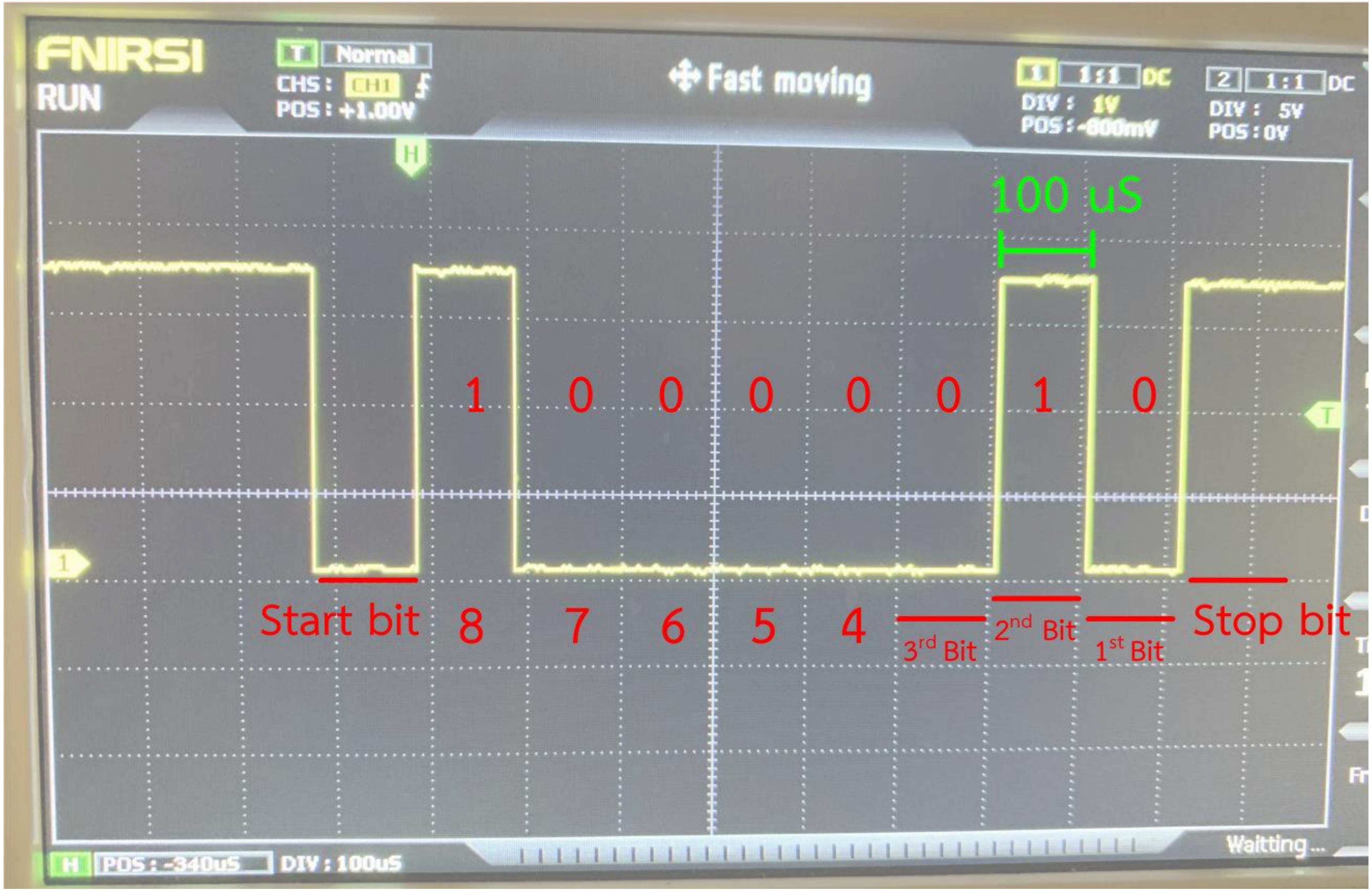
Seeing the data

Measure data using Oscilloscope
USART1 TX -> PC4 (D1 Arduino)

Then hold button





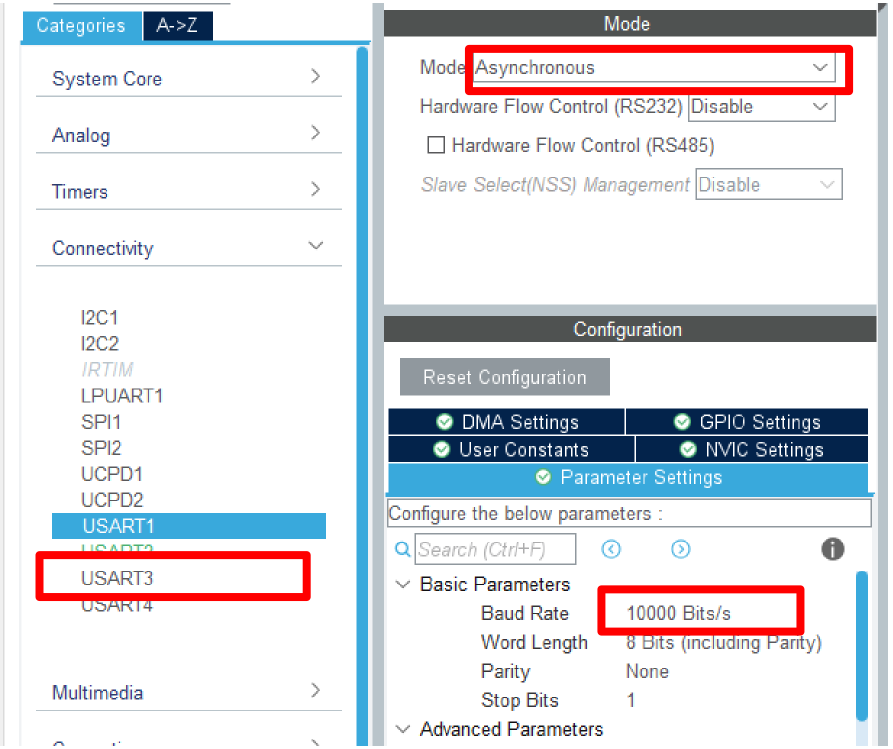

FNIRSI
RUN
T Normal
CHS : CH1
POS : +1.00V
Fast moving
1 1:1 DC
DIV : 1V
POS : -800mV
2 1:1 DC
DIV : 5V
POS : 0V
100 uS
1 0 0 0 0 0 1 0
Start bit 8 7 6 5 4
3rd Bit
2nd Bit
1st Bit
Stop bit
H POS : -340uS DIV : 100uS
Waitting...

*Don’t forget to connect GND

Serial Receiver

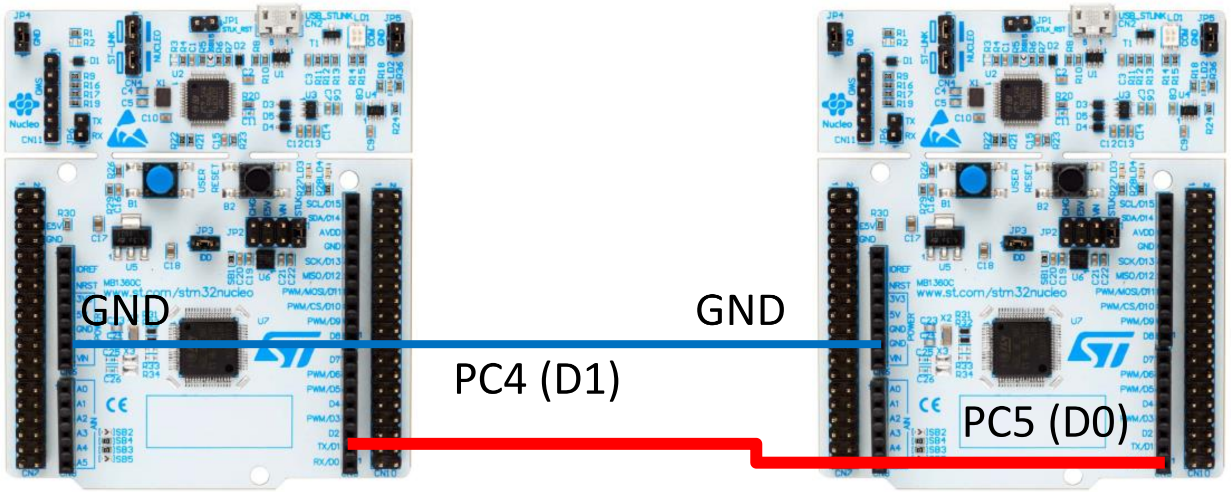
Sender is easy because device can send when every they want! (LAB1)

Receive is more complex because ,We don’t know when the data will come. (LAB2)



Setup USART 1 in Asynchronous mode

Set baud rate to 10000 Bps (8bit)



Sender

Receiver

Lab 2 coding

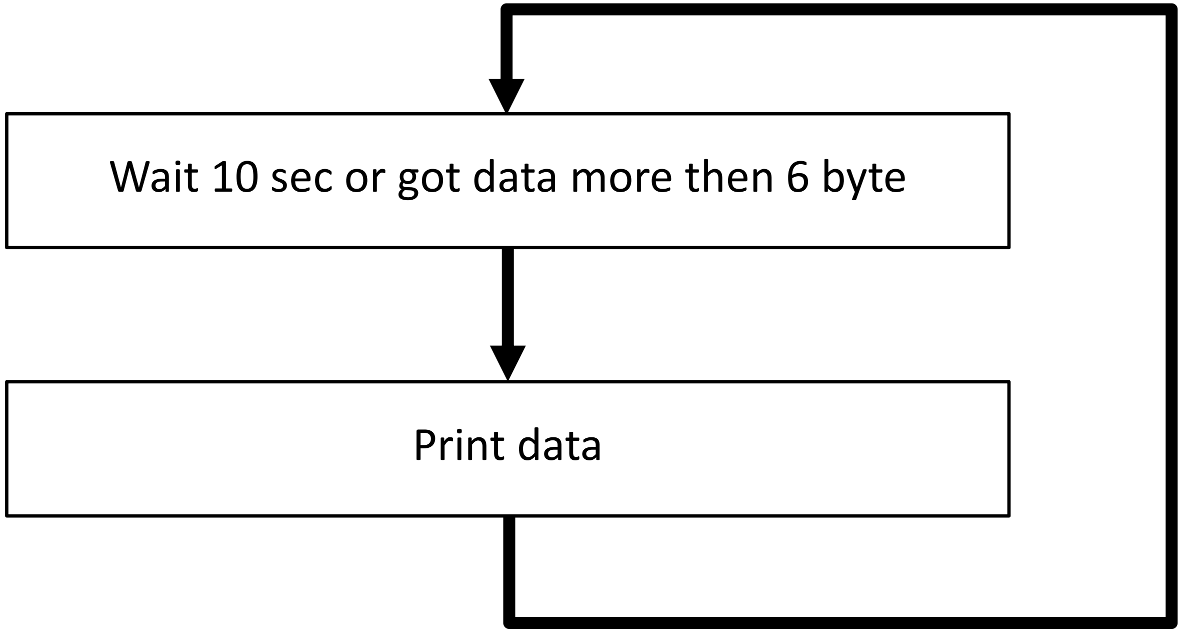
```c
  /* USER CODE END 2 */

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

    /* USER CODE BEGIN 3 */
      unsigned char data[20] = "XXXXXX";
      HAL_UART_Receive(&huart1, data, 6, 10000); // blocking receiving
      HAL_UART_Transmit(&huart2, data, 6, 1000); // print out to console
      HAL_UART_Transmit(&huart2, "\r\n", 2, 1000);  // ending line
      HAL_GPIO_TogglePin(LED_GREEN_GPIO_Port, LED_GREEN_Pin);
      HAL_Delay(1000);
  }
  /* USER CODE END 3 */
}
```

```c
/* USER CODE BEGIN 3 */
unsigned char data[20] = "XXXXXX";
HAL_UART_Receive(&huart1, data, 6, 10000); // blocking receiving
HAL_UART_Transmit(&huart2, data, 6, 1000); // print out to console
HAL_UART_Transmit(&huart2, "\r\n", 2, 1000); // ending line
HAL_GPIO_TogglePin(LED_GREEN_GPIO_Port, LED_GREEN_Pin);
HAL_Delay(1000);
}
/* USER CODE END 3 */
```

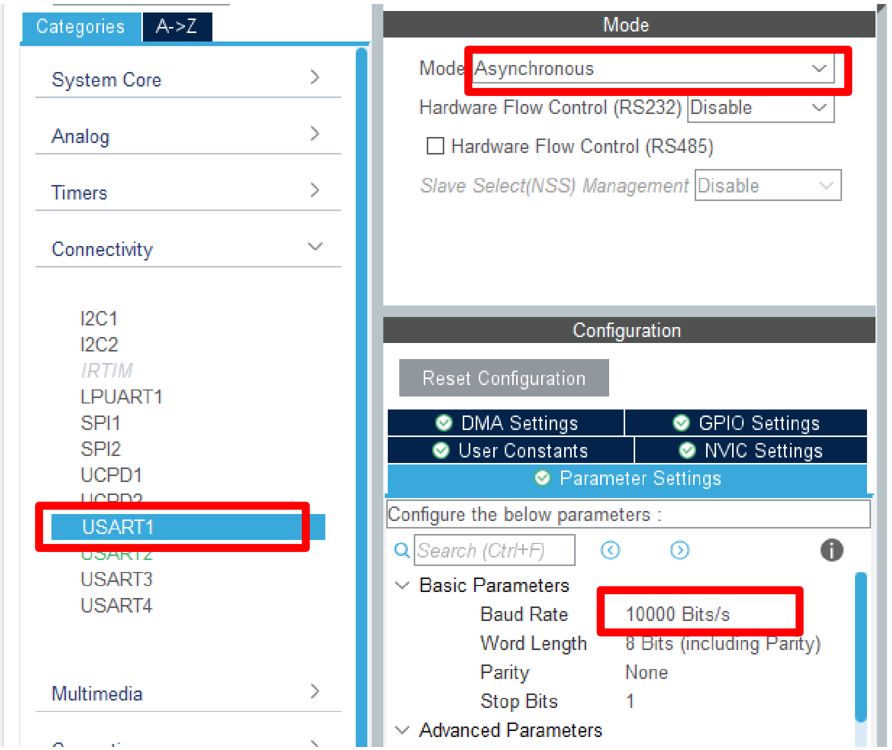
LAB2 blocking mode

```
XXXXXX
AXXXXX
BCDXXX
EFGHXX
IJKLXX
XXXXXX
ABCDXX
XXXXXX
XXXXXX
```

Wait 10 sec every display



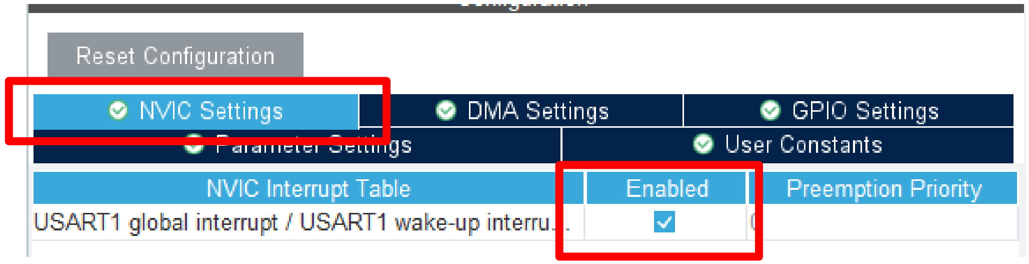
LAB 3 interrupt mode

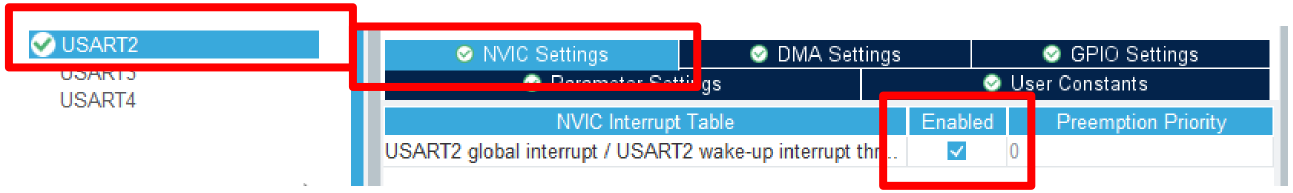


Setup USART 1 in Asynchronous mode

Set baud rate to 10000 Bps (8bit)

Enable interrupt (both USART1 AND USART2)





LAB 3 interrupt mode

```c
/* Private user code -------------------
/* USER CODE BEGIN 0 */
unsigned char data[10] = "XXXXXX\r\n";
/* USER CODE END 0 */

/**
```

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

    /* USER CODE BEGIN 3 */
      HAL_GPIO_TogglePin(LED_GREEN_GPIO_Port, LED_GREEN_Pin);
      HAL_Delay(1000);
  }
  /* USER CODE END 3 */
}
```

```c
/* USER CODE BEGIN 4 */
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
	if(huart == &huart1){
		HAL_UART_Transmit_IT(&huart2, data, 8);
		HAL_UART_Receive_IT(&huart1, data, 6);
	}
}
/* USER CODE END 4 */

```

```c
/* USER CODE BEGIN 0 */
unsigned char data[10] = "XXXXXX\r\n";
/* USER CODE END 0 */


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


/* USER CODE BEGIN 3 */
HAL_GPIO_TogglePin(LED_GREEN_GPIO_Port, LED_GREEN_Pin);
HAL_Delay(1000);
}
/* USER CODE END 3 */


/* USER CODE BEGIN 4 */
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
if(huart == &huart1){
HAL_UART_Transmit_IT(&huart2, data, 8);
HAL_UART_Receive_IT(&huart1, data, 6);
}
}
/* USER CODE END 4 */
```

LAB 3 interrupt mode

```
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{

}
```

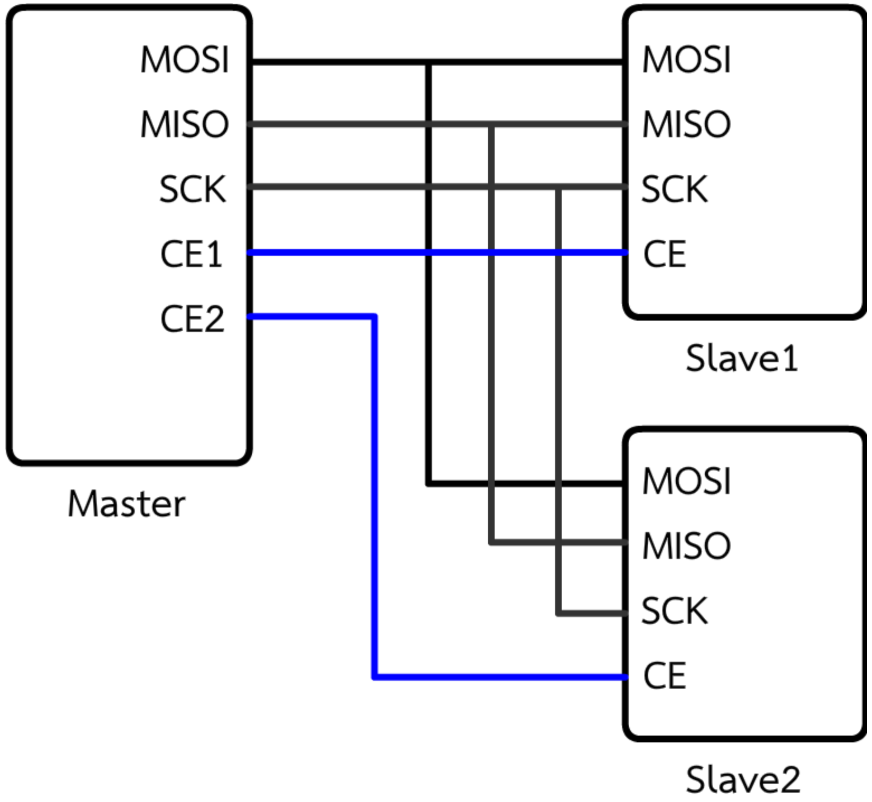
```
Called when received all byte
```

Non - blocking

```
FGHIJK
LMNOPQ
ABCDE
FGHIJK
```

SPI

Serial but Synchronous and Master – Slave topology.



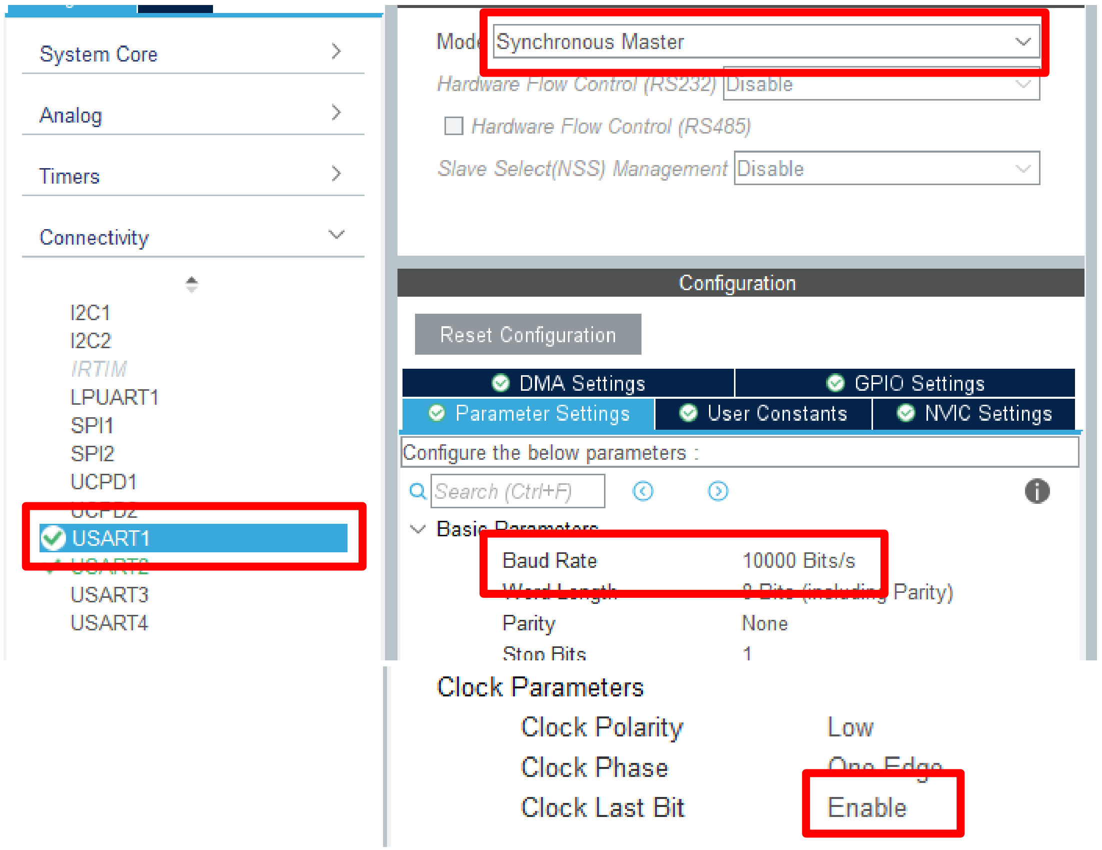
MOSI -> Master out SLAVE in

MISO -> MASTER in SLAVE out

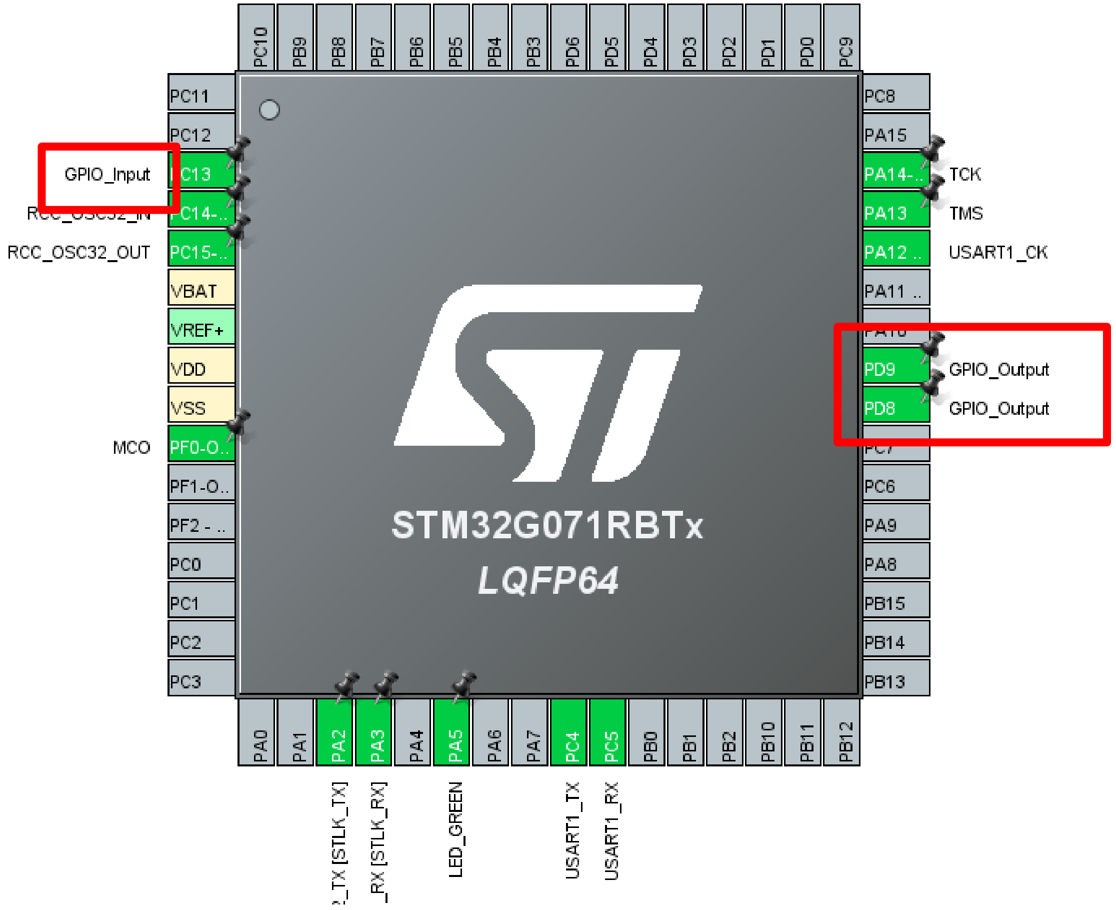
SCK -> Clock signal from MASTER

CE -> chip select (select chip to communicate)

LAB4 SPI (MASTER)



Set USART1 to Synchronous Master (10000 bps) , Enable Clock Last bit



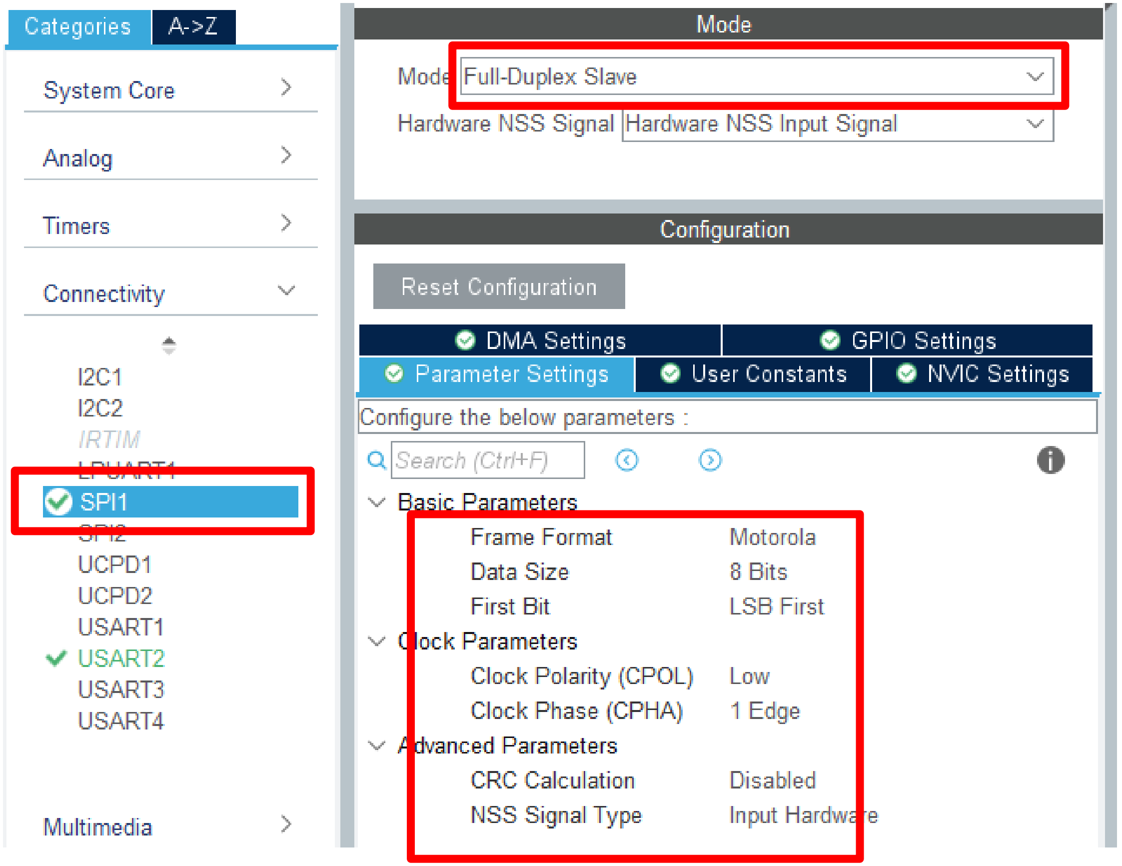
Set PD9,PD8 to GPIO output ,PC13 to input

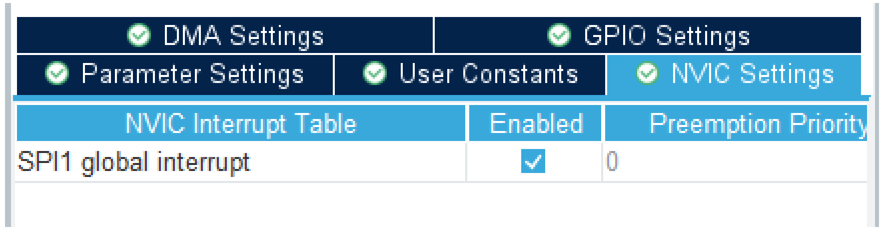
LAB4 SPI (MASTER)

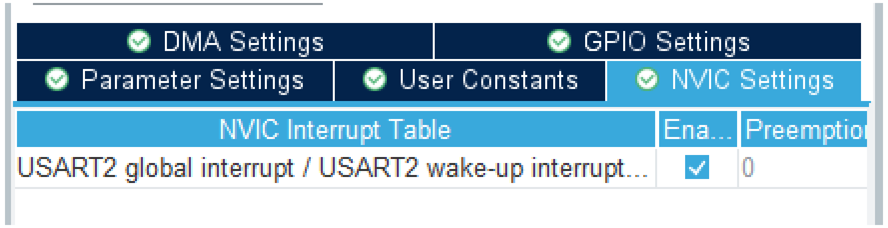
```c
/* USER CODE BEGIN 2 */
unsigned char data[10] = "A";
HAL_GPIO_WritePin(GPIOD, GPIO_PIN_8, 0);
HAL_GPIO_WritePin(GPIOD, GPIO_PIN_9, 0);
/* USER CODE END 2 */

while (1)
{
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
    if(HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13) == 0){
        HAL_GPIO_WritePin(LED_GREEN_GPIO_Port, LED_GREEN_Pin, 0);
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_8, 1);
        HAL_USART_Transmit(&husart1, data, 1, 1000);
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_8, 0);
    }
    else{
        HAL_GPIO_WritePin(LED_GREEN_GPIO_Port, LED_GREEN_Pin, 1);
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_9, 1);
        HAL_USART_Transmit(&husart1, data, 1, 1000);
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_9, 0);
    }
    HAL_Delay(500);
    data[0]++;
    if(data[0] > 'Z'){
        data[0] = 'A';
    }
}
/* USER CODE END 3 */
}
```

```c
/* USER CODE BEGIN 2 */
unsigned char data[10] = "A";
HAL_GPIO_WritePin(GPIOD, GPIO_PIN_8, 0);
HAL_GPIO_WritePin(GPIOD, GPIO_PIN_9, 0);
/* USER CODE END 2 */


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


/* USER CODE BEGIN 3 */
if(HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13) == 0){
HAL_GPIO_WritePin(LED_GREEN_GPIO_Port, LED_GREEN_Pin, 0);
HAL_GPIO_WritePin(GPIOD, GPIO_PIN_8, 1);
HAL_USART_Transmit(&husart1, data, 1, 1000);
HAL_GPIO_WritePin(GPIOD, GPIO_PIN_8, 0);
}
else{
HAL_GPIO_WritePin(LED_GREEN_GPIO_Port, LED_GREEN_Pin, 1);
HAL_GPIO_WritePin(GPIOD, GPIO_PIN_9, 1);
HAL_USART_Transmit(&husart1, data, 1, 1000);
HAL_GPIO_WritePin(GPIOD, GPIO_PIN_9, 0);
}
HAL_Delay(500);
data[0]++;
if(data[0] > 'Z'){
data[0] = 'A';
}
}
/* USER CODE END 3 */
```

LAB5 SPI (Slave)







Set SPI1 to Full-Duplex Slave
Set 8 bit LSB
Enable SPI1 and UART2 Interrupt

*not necessary to assign baud rate why?

LAB5 SPI (Slave)

```c
/* Private user code ---------------------
/* USER CODE BEGIN 0 */
unsigned char data[10] = "a\r\n";
/* USER CODE END 0 */

/**
  * @brief  The application entry point.
```

```c
  MX_SPI1_Init();
  /* USER CODE BEGIN 2 */
  HAL_SPI_Receive_IT(&hspi1, data, 1);
  /* USER CODE END 2 */

  /* Infinite loop */
```

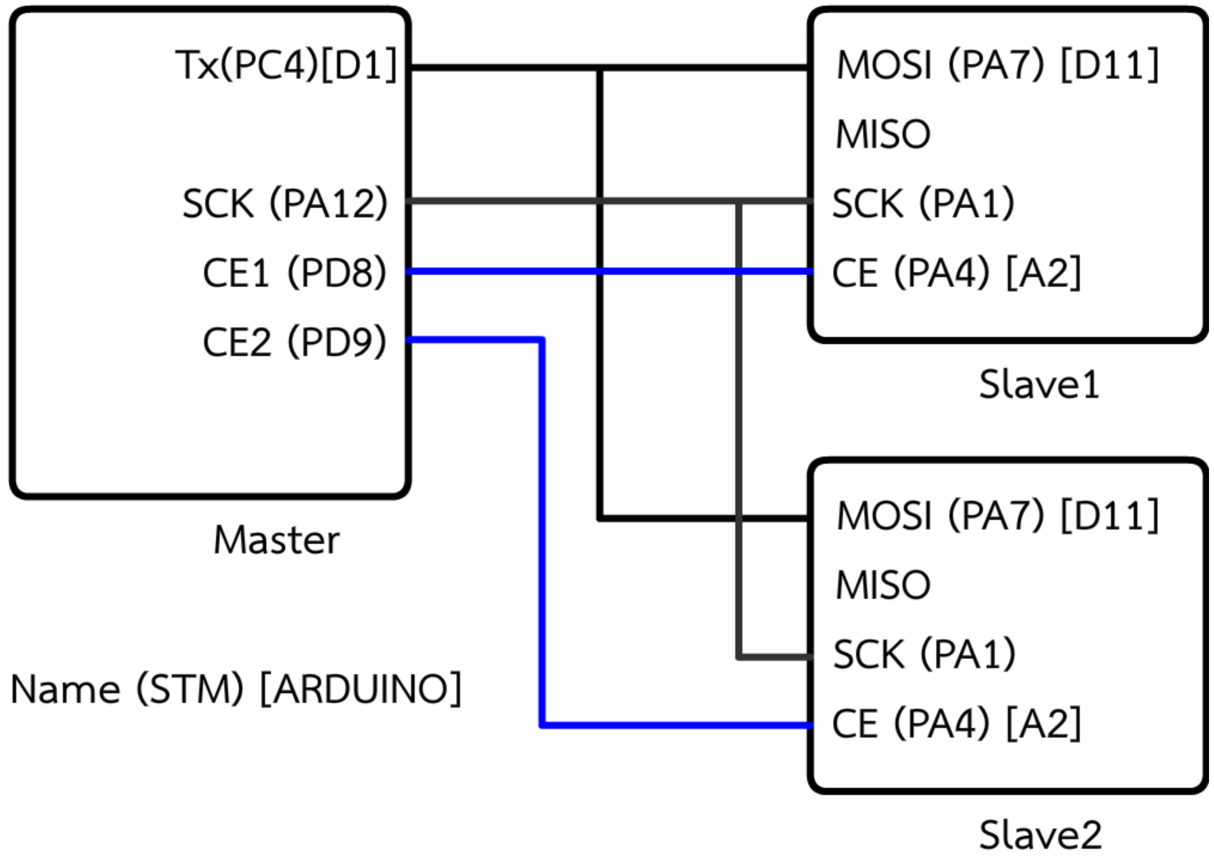
```c
/* USER CODE BEGIN 4 */
void HAL_SPI_RxCpltCallback(SPI_HandleTypeDef *hspi)
{
    if(hspi == &hspi1){
        HAL_UART_Transmit_IT(&huart2, data, 3);
        HAL_SPI_Receive_IT(&hspi1, data, 1);
    }
}
/* USER CODE END 4 */
```


```c
/* USER CODE BEGIN 0 */
unsigned char data[10] = "a\r\n";
/* USER CODE END 0 */
```

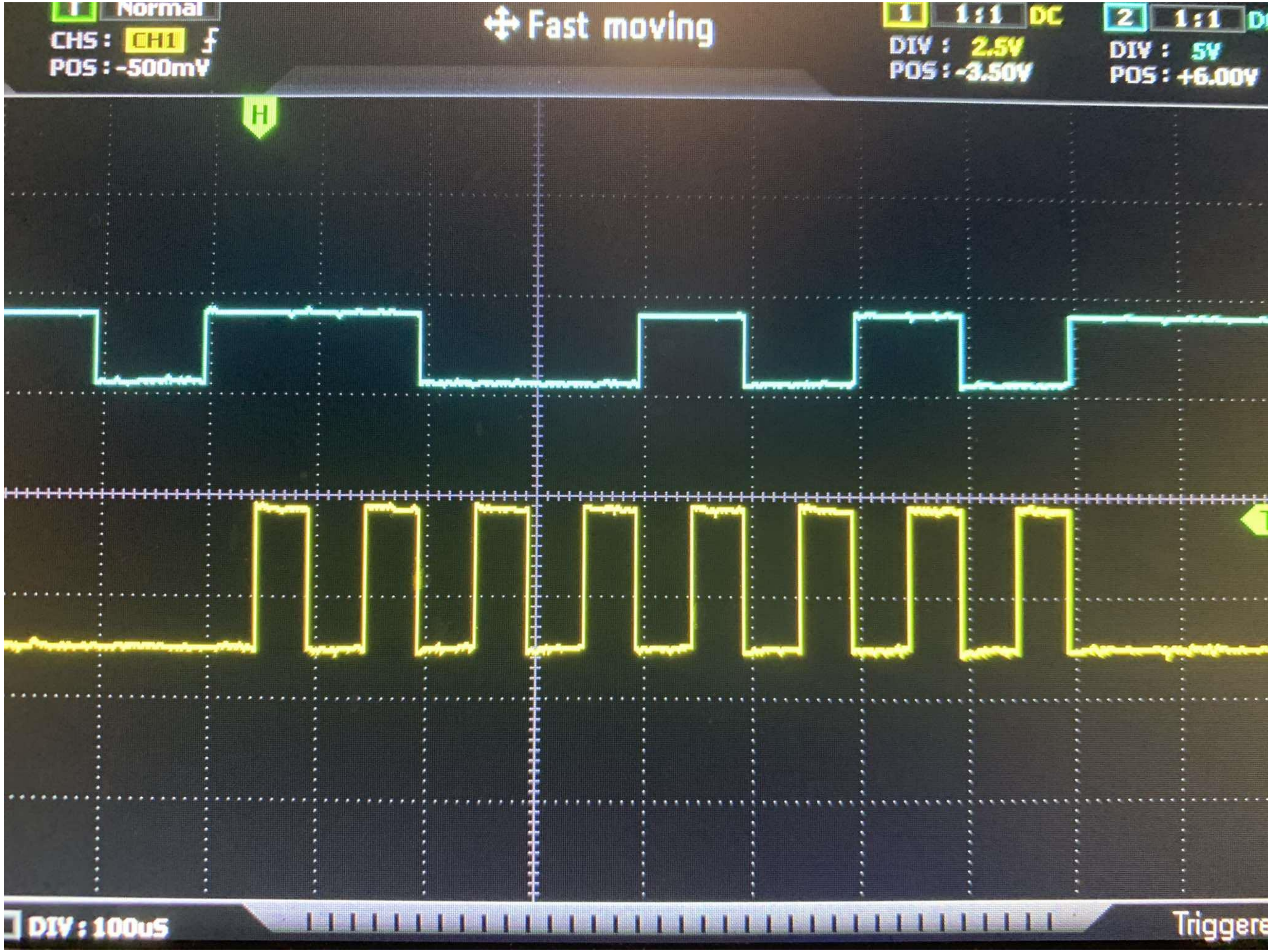
```c
/* USER CODE BEGIN 2 */
HAL_SPI_Receive_IT(&hspi1, data, 1);
/* USER CODE END 2 */
```

```c
/* USER CODE BEGIN 4 */
void HAL_SPI_RxCpltCallback(SPI_HandleTypeDef *hspi)
{
if(hspi == &hspi1){
HAL_UART_Transmit_IT(&huart2, data, 3);
HAL_SPI_Receive_IT(&hspi1, data, 1);
}
}
/* USER CODE END 4 */
```

LAB 4 & 5 Topology



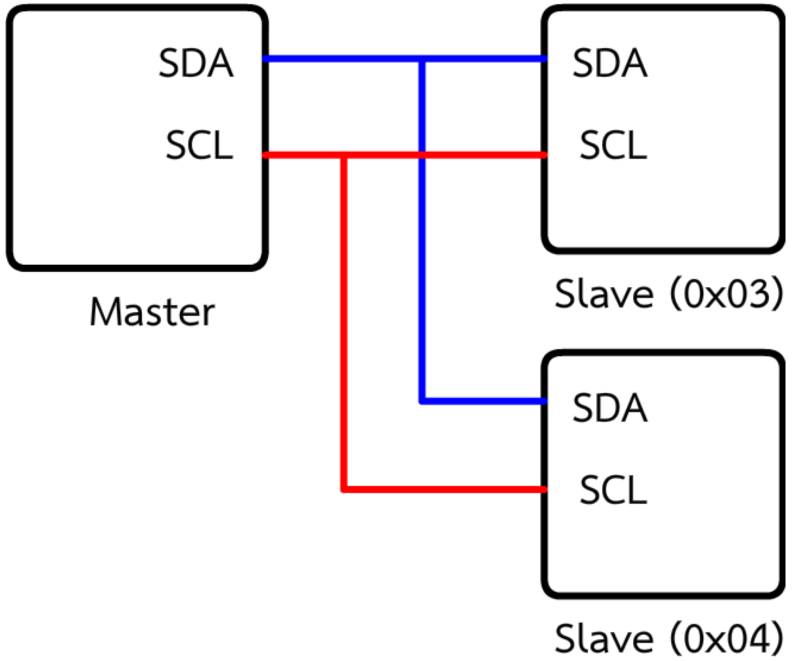
Baud rate is defined by master



Normal
CHS : CH1
POS : -500mV
Fast moving
1 1:1 DC
DIV : 2.5V
POS : -3.50V
2 1:1
DIV : 5V
POS : +6.00V
H
DIV : 100uS

I2C

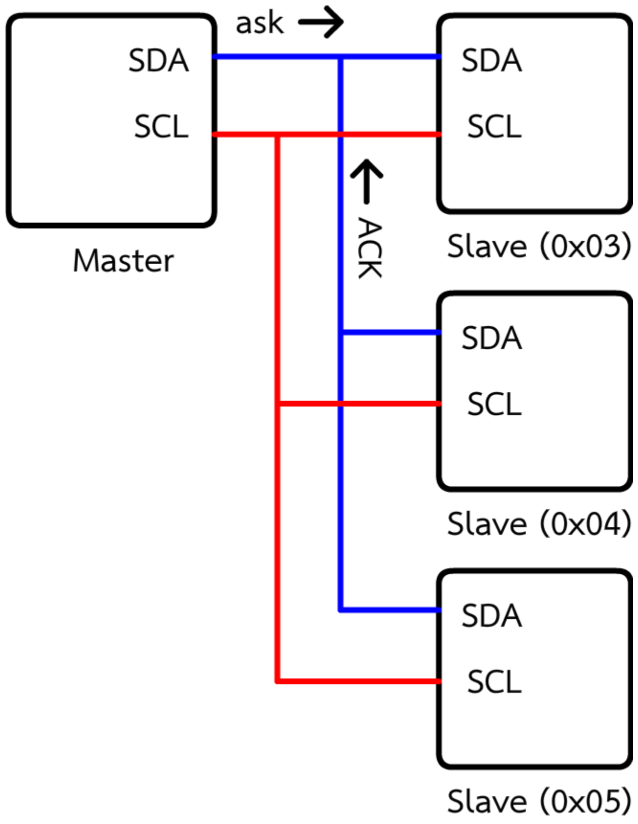
SPI but use address in communication to select chip (use only 2 wire)



SDA -> DATA
SCL -> CLOCK

Address was assigned to every device except master
Select slave by address

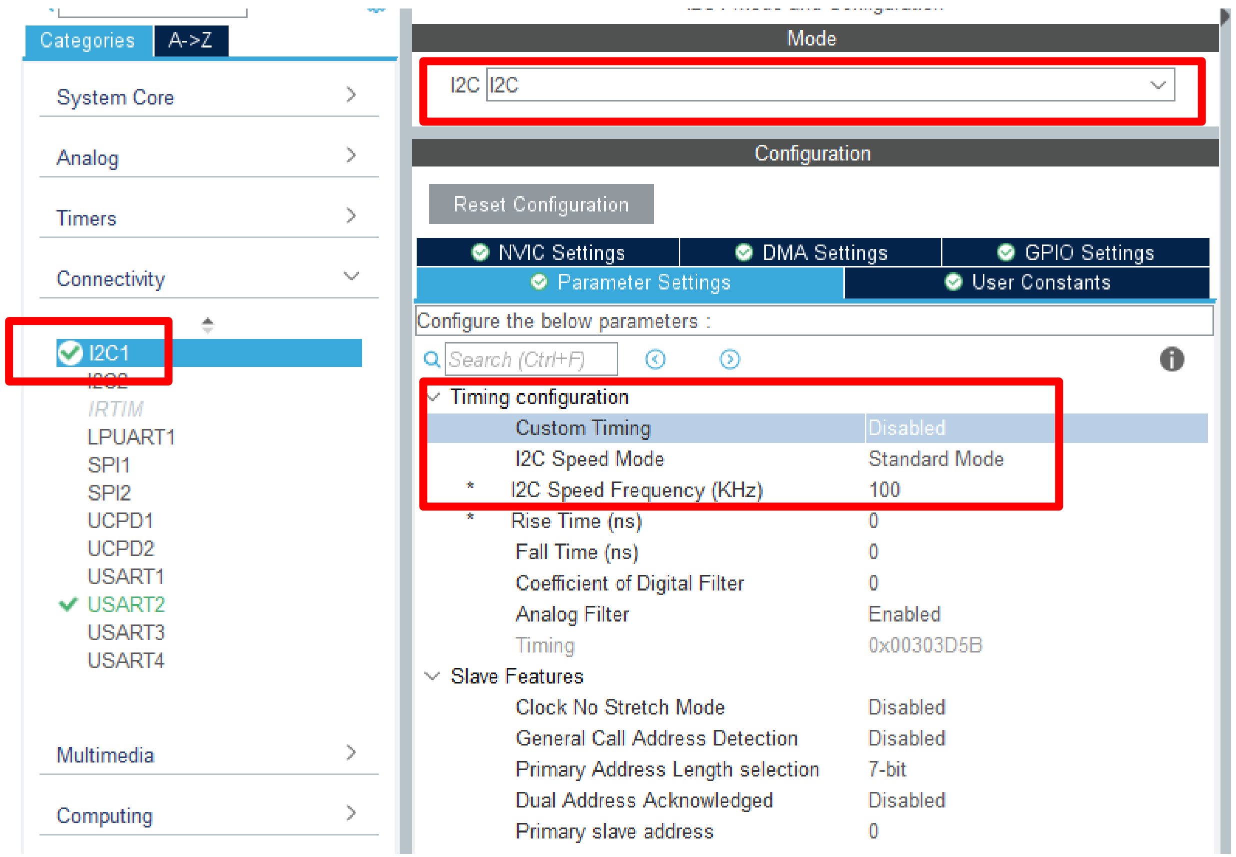
I2C

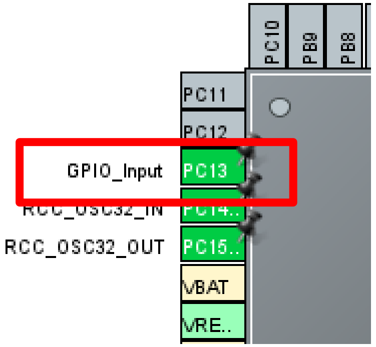


Connect start by master ask for device xx
If there are device address xx , device will send ACK back to BUS

ACK = acknowledgement = สัญญาณตอบกลับ
NACK = negative-acknowledgement = สัญญาณตอบกลับแบบลบ

LAB 6 I2C (MASTER)

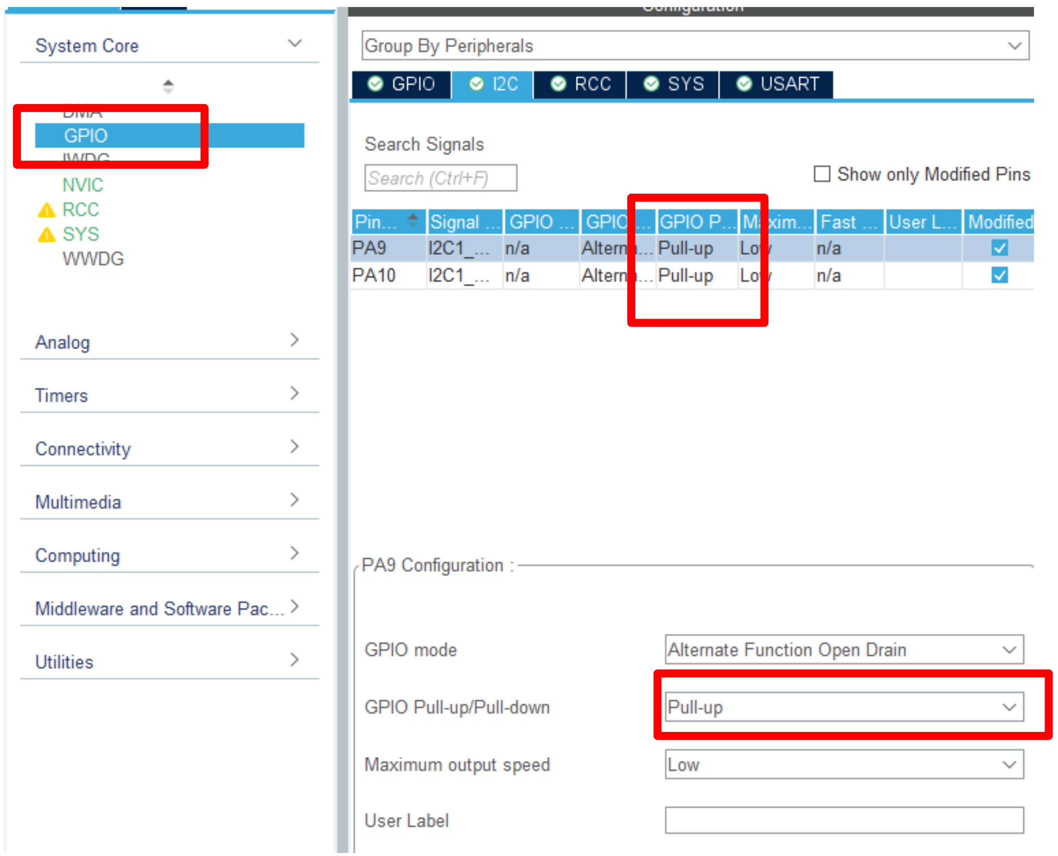




Enable I2C set standard frequency 100000 Hz
Set address to 0 (MASTER)
Set PC13 to Input

LAB 6 I2C (MASTER)

Set pullup PA9(SCL) , PA10(SDA)

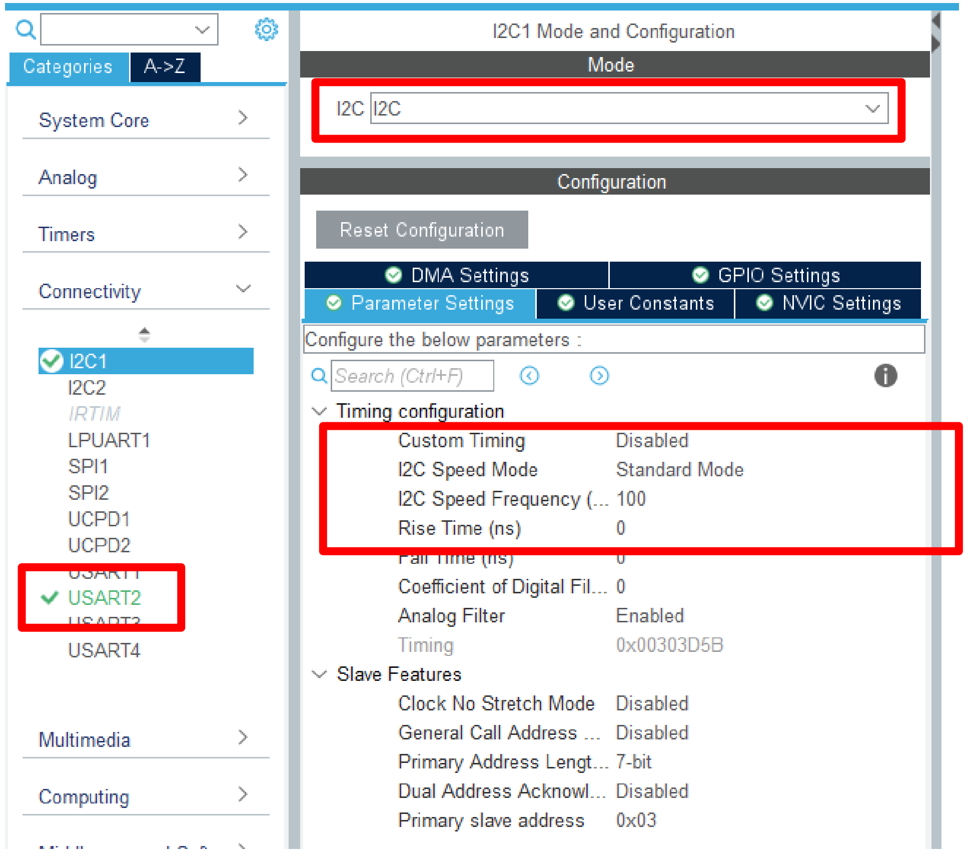


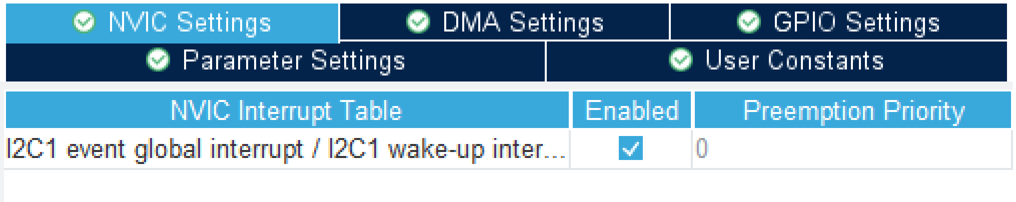
LAB 6 I2C (MASTER)

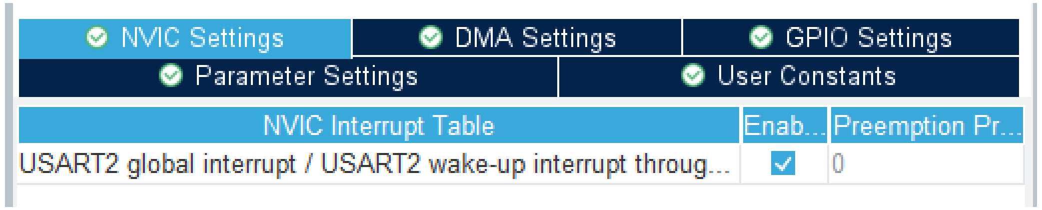
```c
/* USER CODE BEGIN 2 */
unsigned char data[10] = "haruhi";
/* USER CODE END 2 */

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

  /* USER CODE BEGIN 3 */
    if(HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13) == 0){
        HAL_GPIO_WritePin(LED_GREEN_GPIO_Port, LED_GREEN_Pin, 0);
        HAL_I2C_Master_Transmit(&hi2c1, 0x03 << 1, data, 6, 1000);
    }
    else{
        HAL_GPIO_WritePin(LED_GREEN_GPIO_Port, LED_GREEN_Pin, 1);
        HAL_I2C_Master_Transmit(&hi2c1, 0x04 << 1, data, 6, 1000);
    }
    HAL_Delay(2000);
    data[0]++;
    if(data[0] > 'z'){
        data[0] = 'a';
    }
}
/* USER CODE END 3 */
}
```


```c
/* USER CODE BEGIN 2 */
unsigned char data[10] = "haruhi";
/* USER CODE END 2 */

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

/* USER CODE BEGIN 3 */
if(HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13) == 0){
HAL_GPIO_WritePin(LED_GREEN_GPIO_Port, LED_GREEN_Pin, 0);
HAL_I2C_Master_Transmit(&hi2c1, 0x03 << 1, data, 6, 1000);
}
else{
HAL_GPIO_WritePin(LED_GREEN_GPIO_Port, LED_GREEN_Pin, 1);
HAL_I2C_Master_Transmit(&hi2c1, 0x04 << 1, data, 6, 1000);
}
HAL_Delay(2000);
data[0]++;
if(data[0] > 'z'){
data[0] = 'a';
}
}
/* USER CODE END 3 */
```

LAB 7 I2C Slave



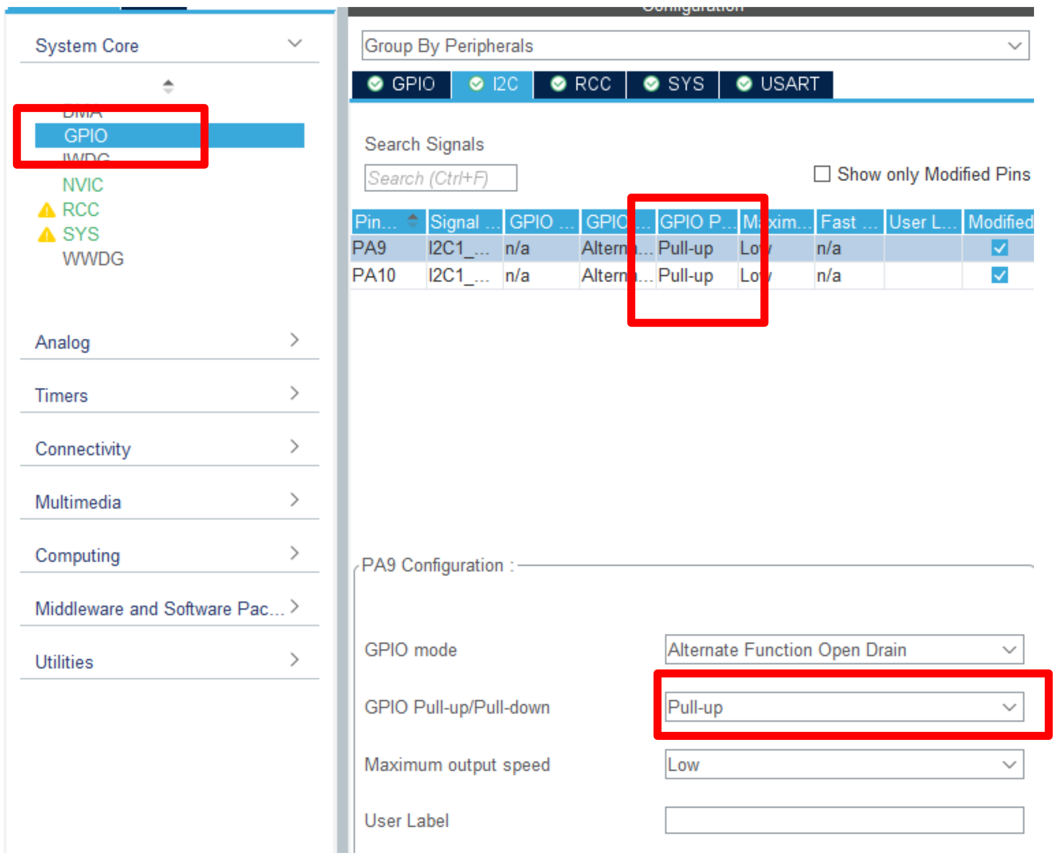




Enable I2C set standard frequency 100000 Hz
Set address to 0x03 and 0x04
Enable I2C1 and UART2 interrupt

LAB 7 I2C Slave

Set pullup PA9(SCL) , PA10(SDA)



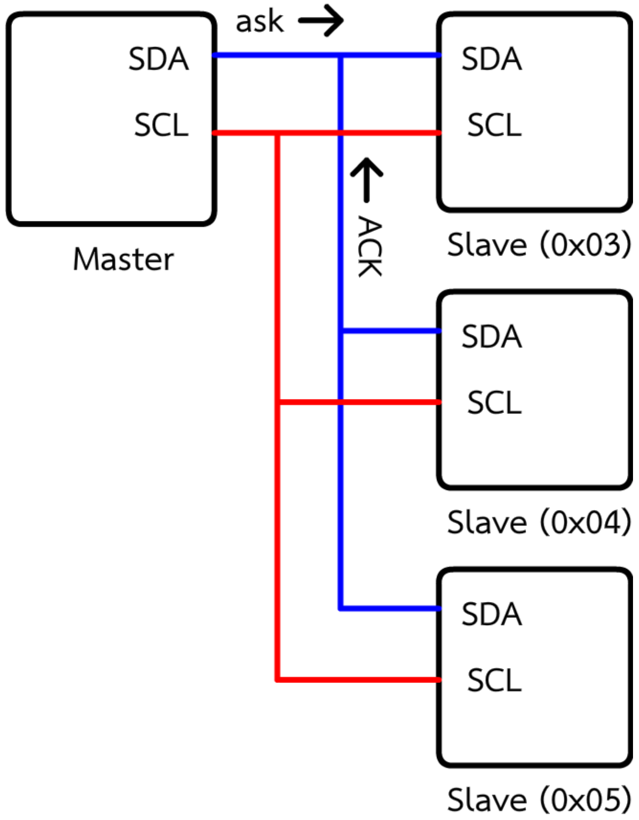
LAB 7 I2C Slave

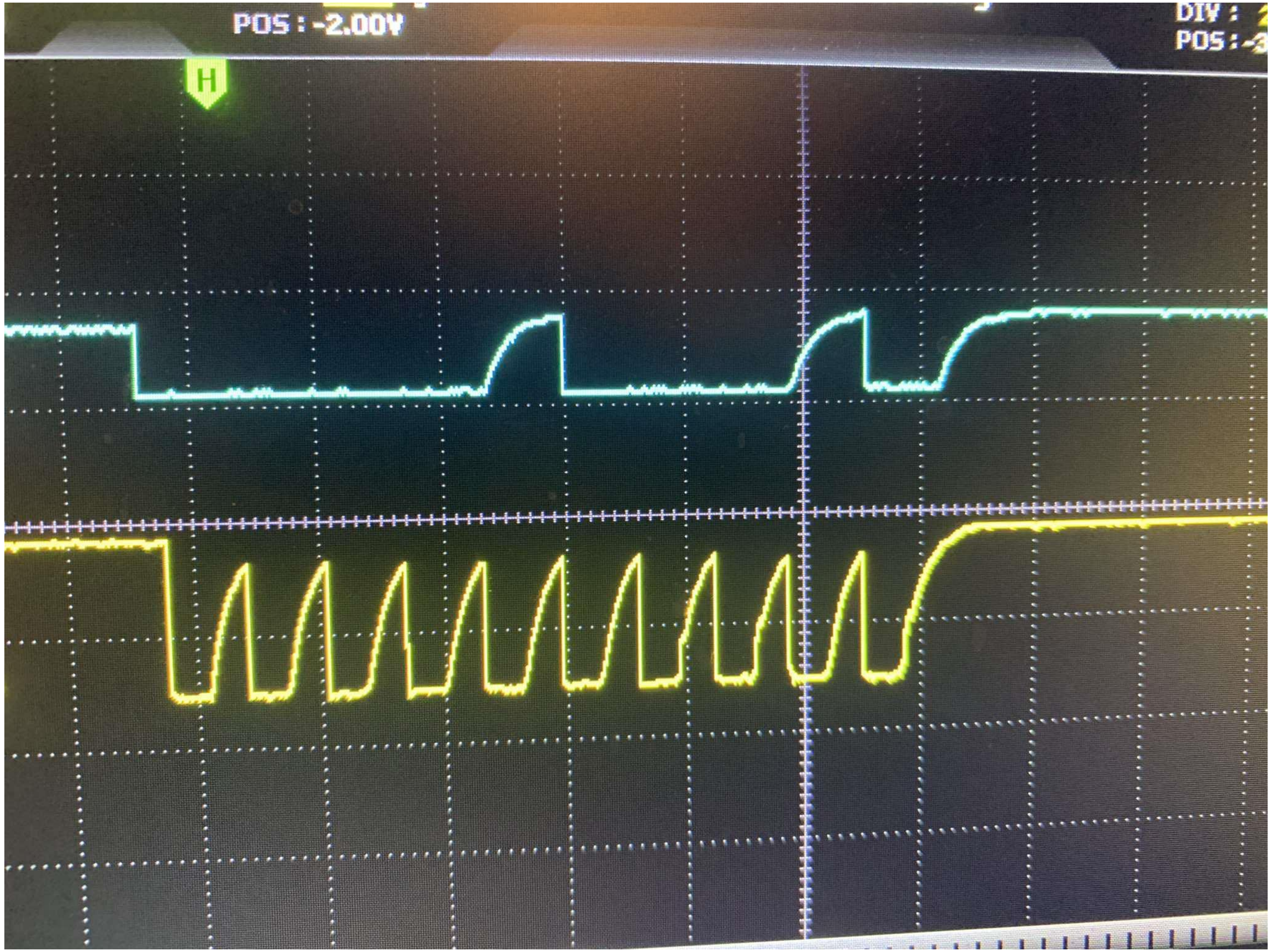
```
/* USER CODE BEGIN 0 */
unsigned char data[10] = "A\r\n";
/* USER CODE END 0 */
```

```
/* USER CODE BEGIN 2 */
HAL_I2C_EnableListen_IT(&hi2c1);
/* USER CODE END 2 */
```

```
/* USER CODE BEGIN 4 */
void HAL_I2C_ListenCpltCallback (I2C_HandleTypeDef *hi2c)
{
HAL_I2C_EnableListen_IT(hi2c);
}

extern void HAL_I2C_AddrCallback(I2C_HandleTypeDef *hi2c, uint8_t TransferDirection, uint16_t AddrMatchCode)
{
if(TransferDirection == I2C_DIRECTION_TRANSMIT) // if the master wants to transmit the data
{
HAL_I2C_Slave_Sequential_Receive_IT(hi2c, data, 6, I2C_FIRST_AND_LAST_FRAME);
}
else // master requesting the data is not supported yet
{
Error_Handler();
}
}

void HAL_I2C_SlaveRxCpltCallback(I2C_HandleTypeDef *hi2c)
{
if(hi2c == &hi2c1){
HAL_UART_Transmit_IT(&huart2, data, 6);
}
}
void HAL_I2C_ErrorCallback(I2C_HandleTypeDef *hi2c)
{
HAL_I2C_EnableListen_IT(hi2c);
}

/* USER CODE END 4 */
```

I2C topology




POS:-2.00V
H