Chungbuk National University - 충북대학교

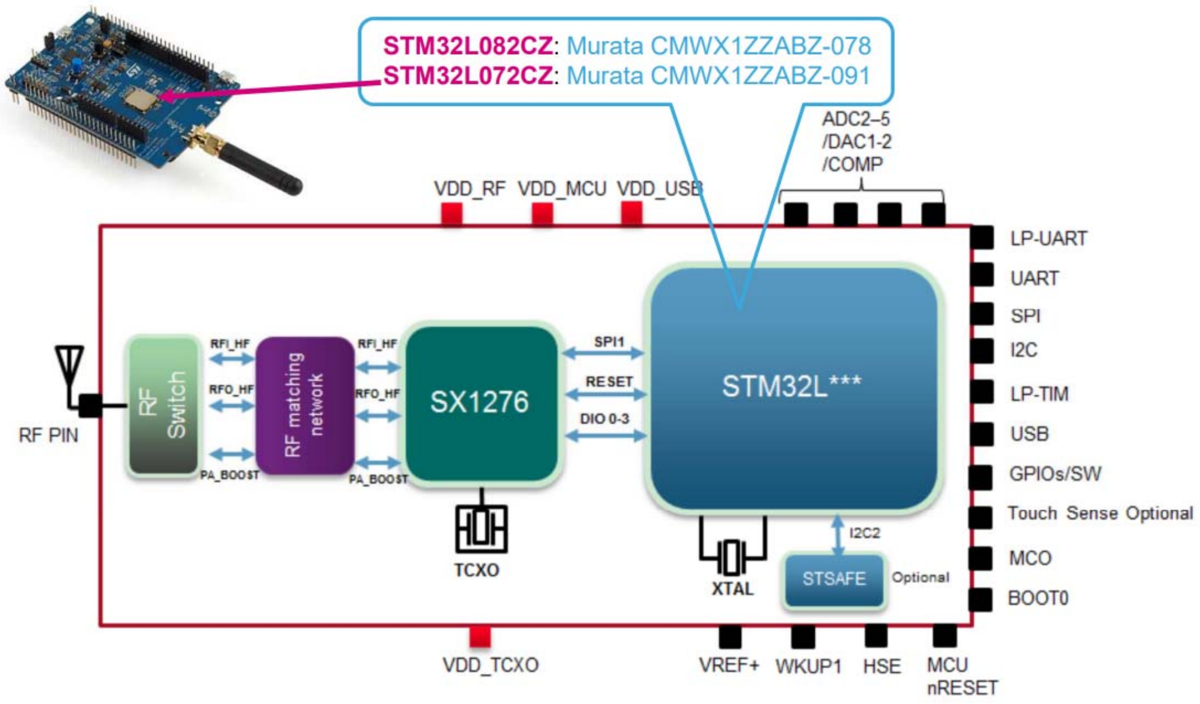
CMWX1ZZABZ-078 LoRa module

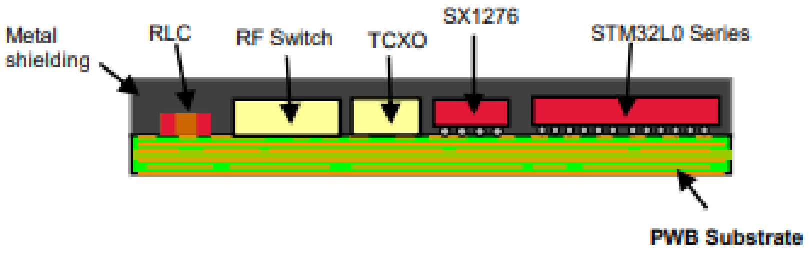
2022.03.11

Contents

1. Block diagram
2. Specifications
3. Principle of operation
4. Coding and Coding description

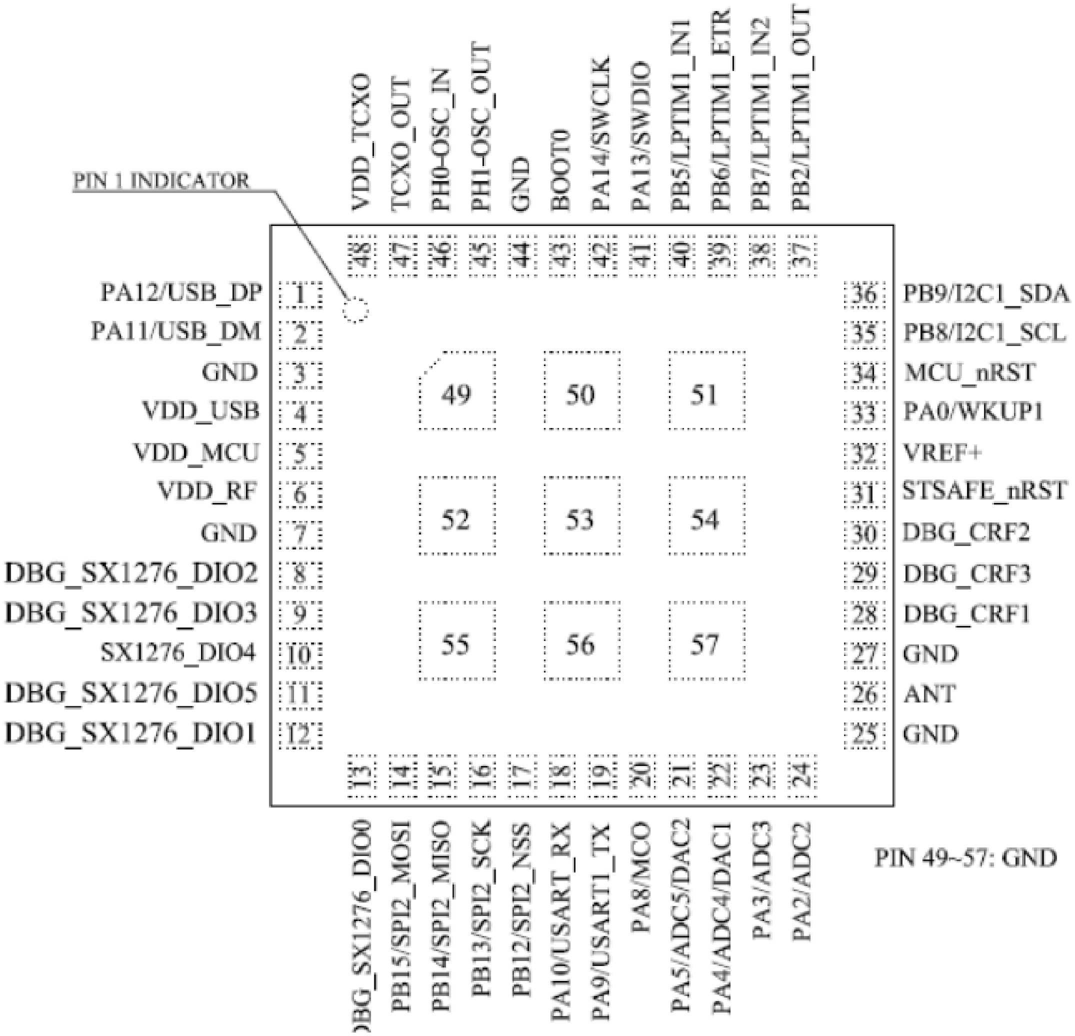
1. Block diagram





1. Pinout

- Pinout of CMWX1ZZABZ-078 / 091



2. Specifications

Specifications	
Product Category	RF Modules
Manufacturer	Murata
FR/BB chipset	SX1276
MCU chipset	STM32L082
Frequency	860 MHz to 930 MHz
Output Power	14 dBm/20 dBm
RF Sensitivity	-135dBm
Interface Type	I2C, SPI, UART, USB
Operating Supply Voltage	2.2 V to 3.6 V
Maximum Operating Temperature	+ 85 C
Dimensions	12.5 mm x 11.6 mm x 1.76 mm
Minimum Operating Temperature	- 40 C
Packaging	Reel
Series	LoRa
Unit Weight	235 g

2. Specifications

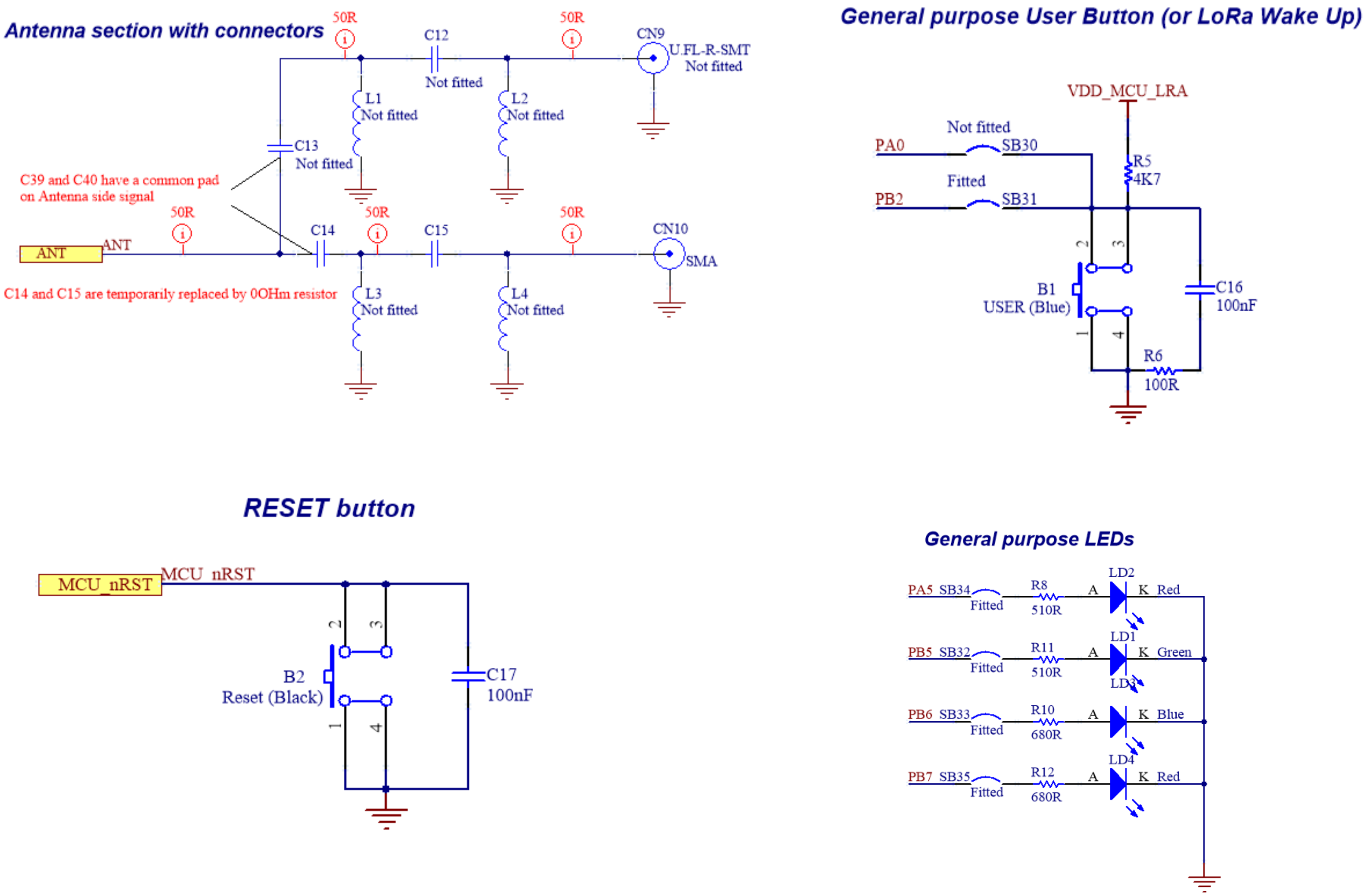
Power consumption:

Work Mode	Voltage	Current	Power Rate
Active Mode	3.3V	Receiving -22mA Transmit -44mA	72.6mW 145.2mW
Sleep Mode	3.3V	1.4uA	4.62uW

3. Principle of operation

Figure: Schematic of Lora module

3. Principle of operation



Antenna section with connectors

General purpose User Button (or LoRa Wake Up)

RESET button

General purpose LEDs

Figure: Schematic of Lora module

3. Principle of operation

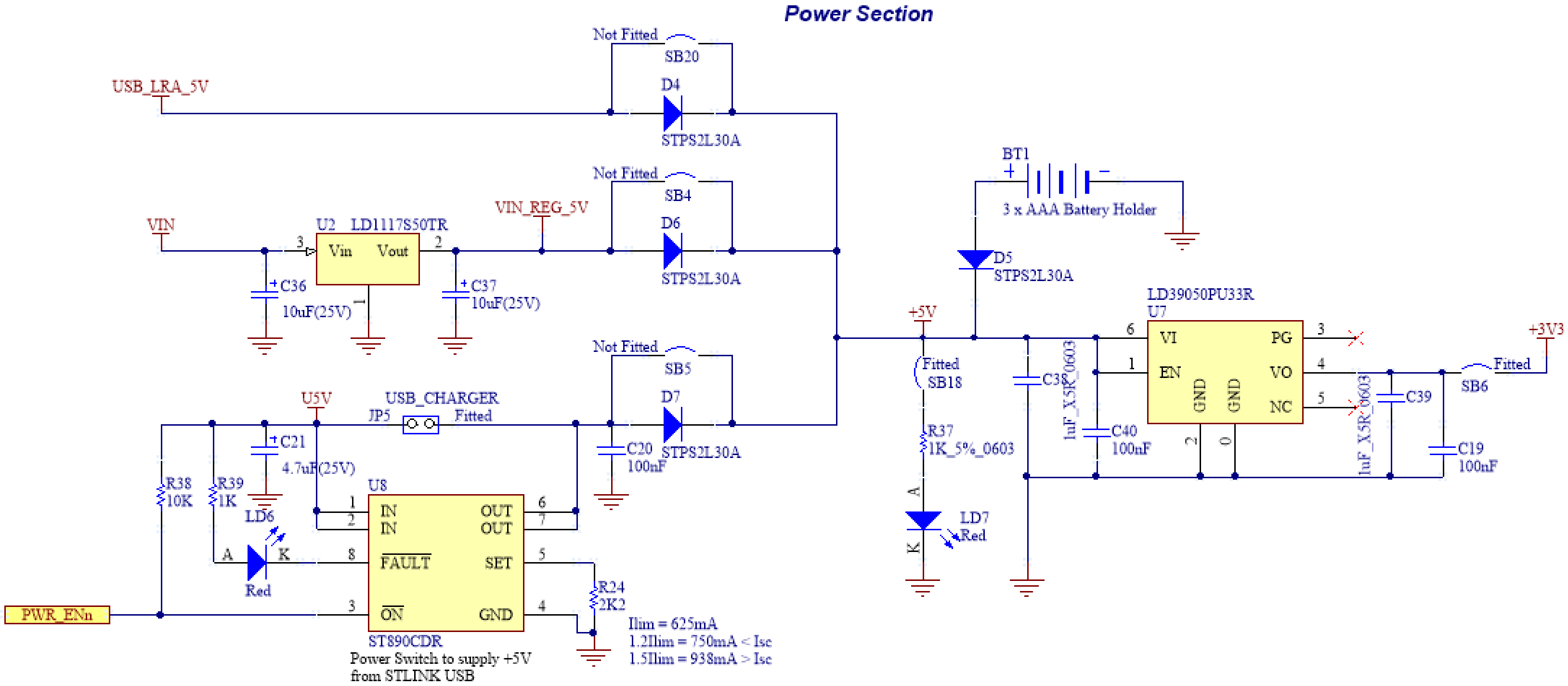
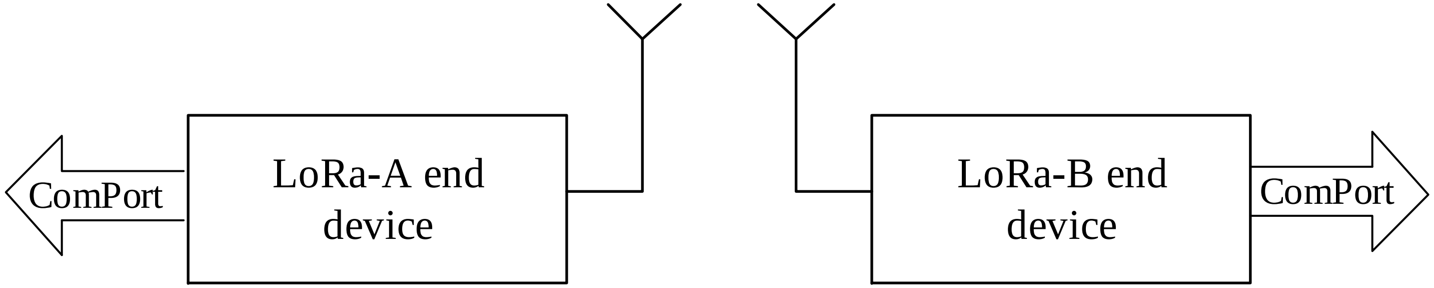


Figure: Schematic of power converters

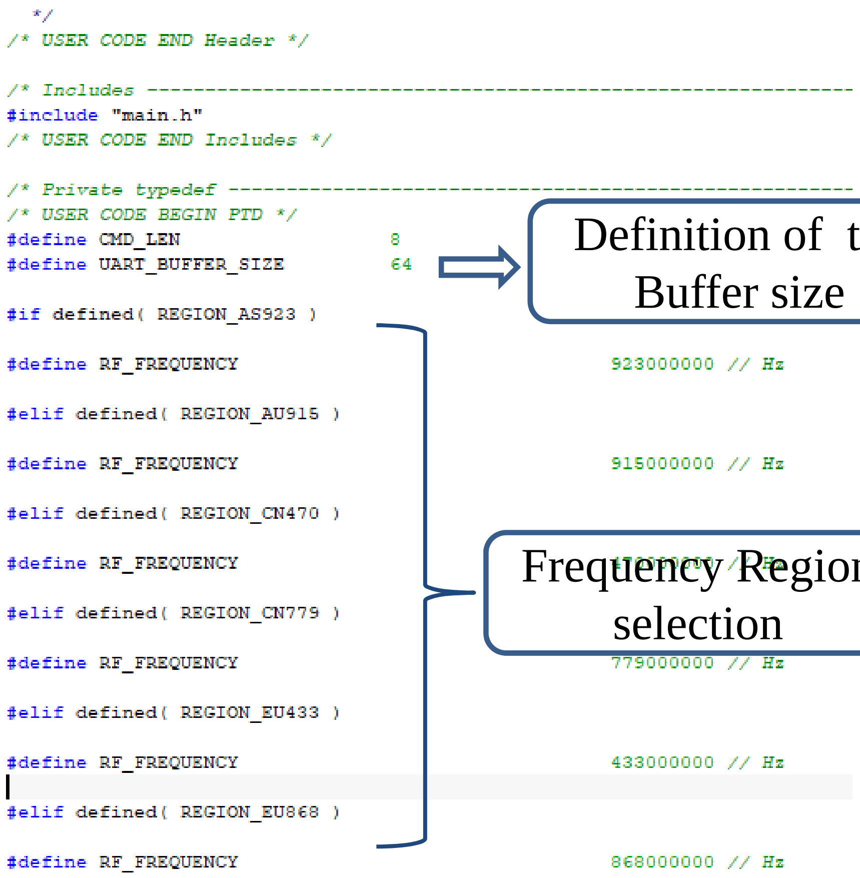
3. Principle of operation



This application is a simple Rx/Tx RF link between two LoRa end devices. By default, each LoRa end device starts as a master, transmits a DATA message, and waits for an answer. The first LoRa end device receiving a DATA2 message becomes a slave and an swers the master with a DATA1 message.

The module is powered by an STM32L072CZ microcontroller and SX1276 transceiver . The transceiver features the LoRa long-range modem, providing ultra-long-range spre ad-spectrum communication and high interference immunity, minimizing current cons umption. Since CMWX1ZZABZ-091 is an open module, the user has access to all ST M32L072CZ peripherals such as ADC, 16-bit timer, LP-UART, I2C, SPI, and USB 2.0 .

4. Coding and Coding description

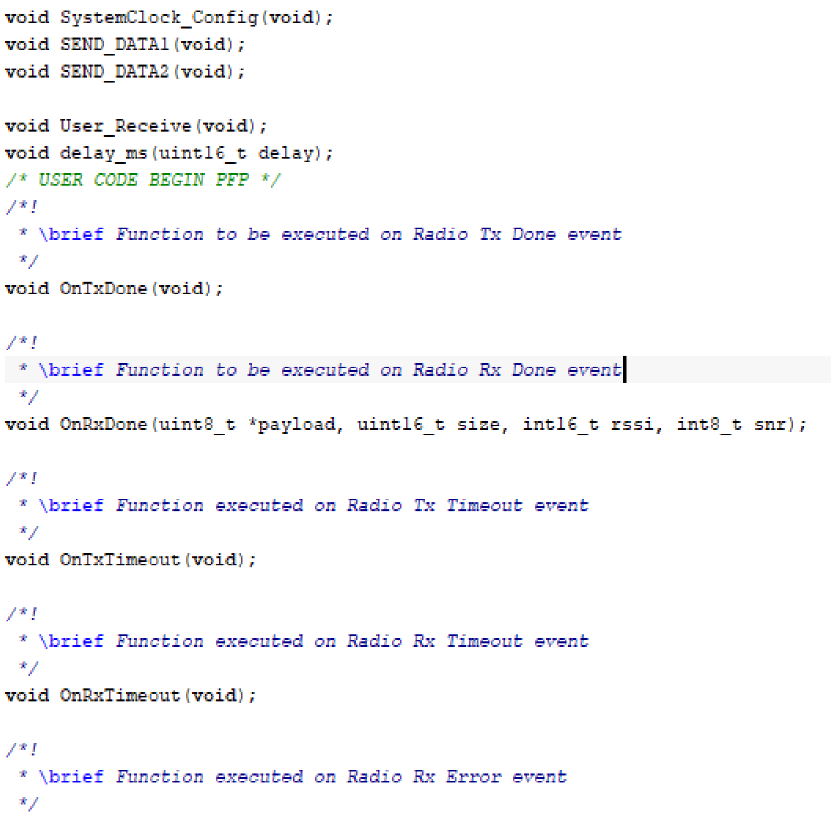


```
  */
/* USER CODE END Header */

/* Includes ------------------------------------------------------------------*/
#include "main.h"
/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */
#define CMD_LEN                 8
#define UART_BUFFER_SIZE        64

#if defined( REGION_AS923 )

#define RF_FREQUENCY                                923000000 // Hz

#elif defined( REGION_AU915 )

#define RF_FREQUENCY                                915000000 // Hz

#elif defined( REGION_CN470 )

#define RF_FREQUENCY                                470000000 // Hz

#elif defined( REGION_CN779 )

#define RF_FREQUENCY                                779000000 // Hz

#elif defined( REGION_EU433 )

#define RF_FREQUENCY                                433000000 // Hz

#elif defined( REGION_EU868 )

#define RF_FREQUENCY                                868000000 // Hz
```

Definition of the Buffer size

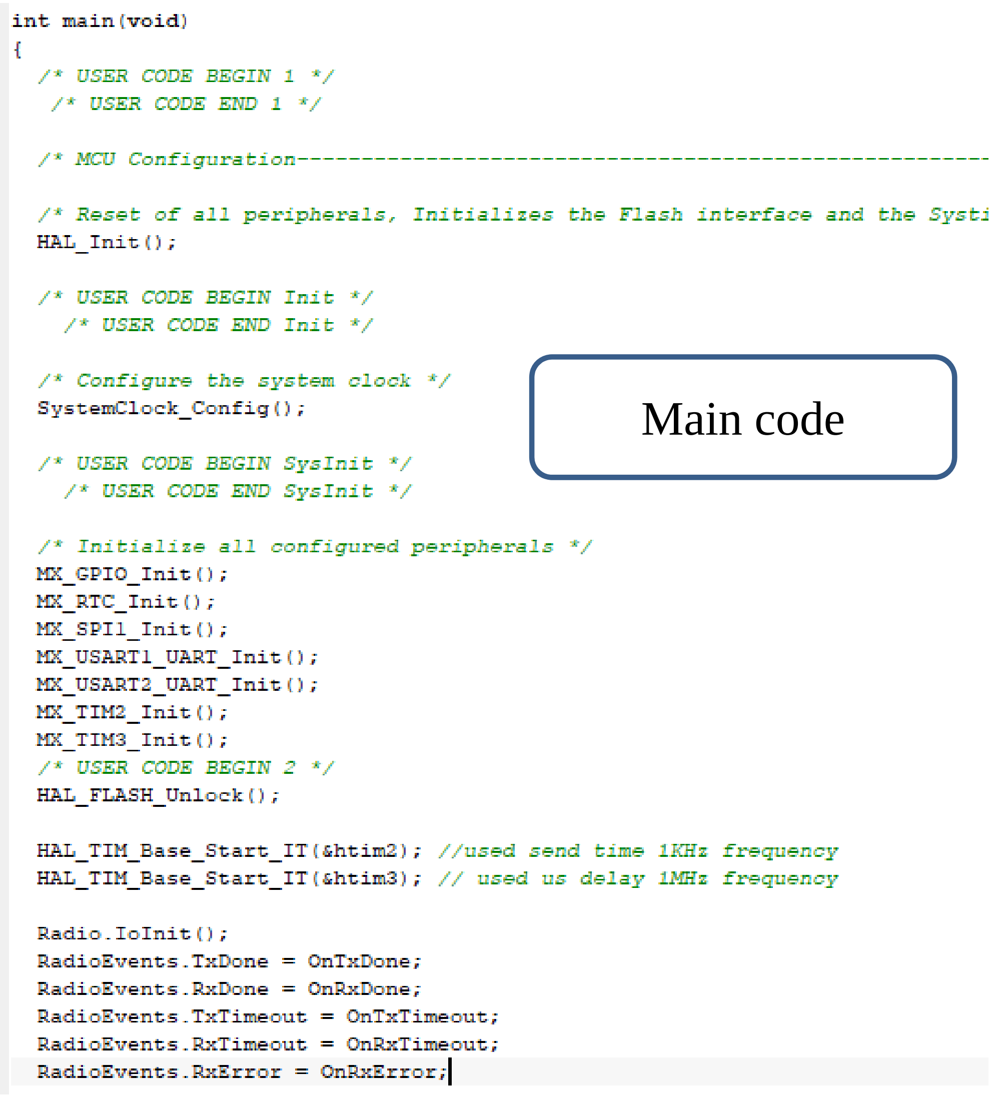
Frequency Region selection

Definition of SUBCODE



```
void SystemClock_Config(void);
void SEND_DATA1(void);
void SEND_DATA2(void);

void User_Receive(void);
void delay_ms(uint16_t delay);
/* USER CODE BEGIN PFP */
/*!
 * \brief Function to be executed on Radio Tx Done event
 */
void OnTxDone(void);

/*!
 * \brief Function to be executed on Radio Rx Done event
 */
void OnRxDone(uint8_t *payload, uint16_t size, int16_t rssi, int8_t snr);

/*!
 * \brief Function executed on Radio Tx Timeout event
 */
void OnTxTimeout(void);

/*!
 * \brief Function executed on Radio Rx Timeout event
 */
void OnRxTimeout(void);

/*!
 * \brief Function executed on Radio Rx Error event
 */
```

4. Coding and Coding description

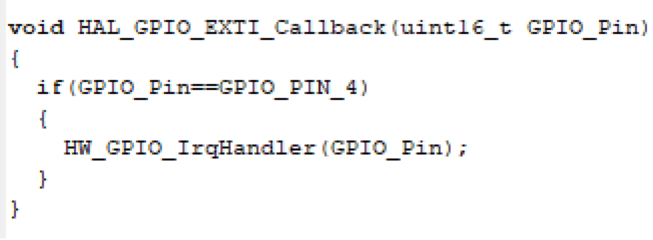


```
int main(void)
{
  /* USER CODE BEGIN 1 */
   /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------

  /* Reset of all peripherals, Initializes the Flash interface and the Systi
  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_RTC_Init();
  MX_SPI1_Init();
  MX_USART1_UART_Init();
  MX_USART2_UART_Init();
  MX_TIM2_Init();
  MX_TIM3_Init();
  /* USER CODE BEGIN 2 */
  HAL_FLASH_Unlock();

  HAL_TIM_Base_Start_IT(&htim2); //used send time 1KHz frequency
  HAL_TIM_Base_Start_IT(&htim3); // used us delay 1MHz frequency

  Radio.IoInit();
  RadioEvents.TxDone = OnTxDone;
  RadioEvents.RxDone = OnRxDone;
  RadioEvents.TxTimeout = OnTxTimeout;
  RadioEvents.RxTimeout = OnRxTimeout;
  RadioEvents.RxError = OnRxError;
```

Main code

External Interrupt
EXTI line interrupt detected

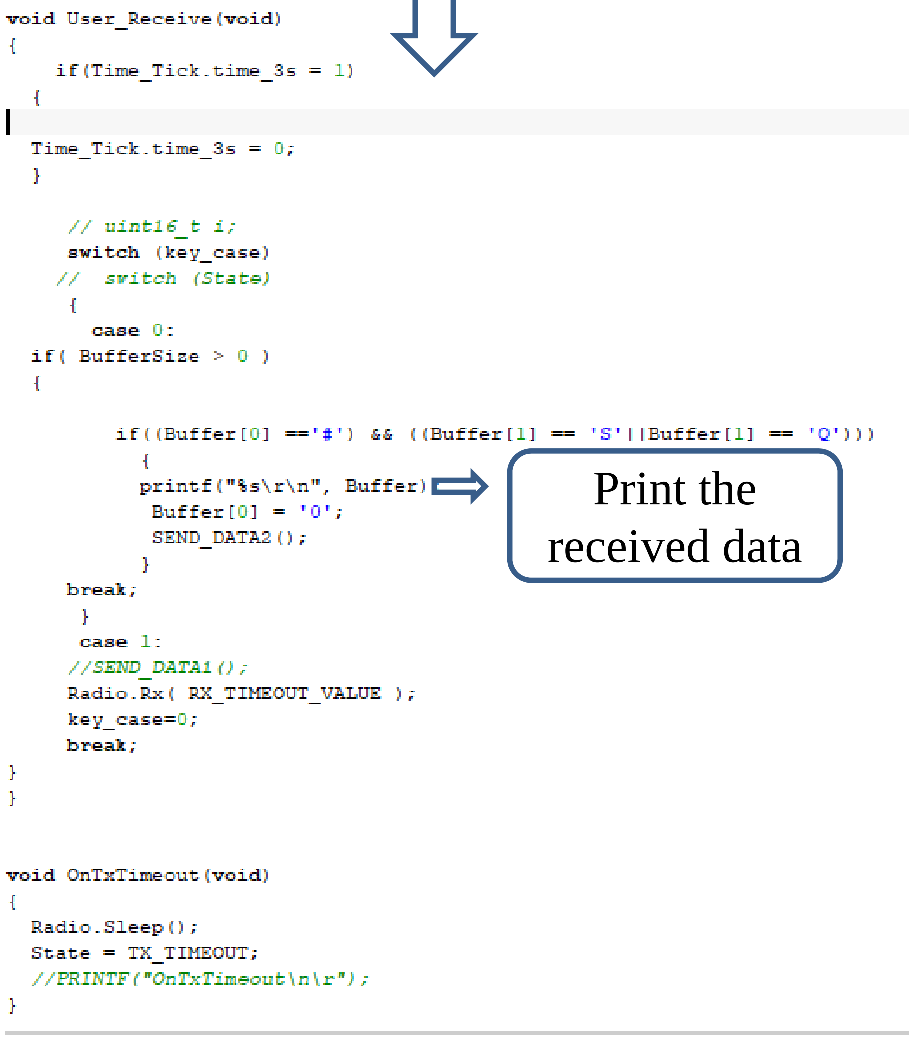
⇩



```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
  if(GPIO_Pin==GPIO_PIN_4)
  {
    HW_GPIO_IrqHandler(GPIO_Pin);
  }
}
```

4. Coding and Coding description

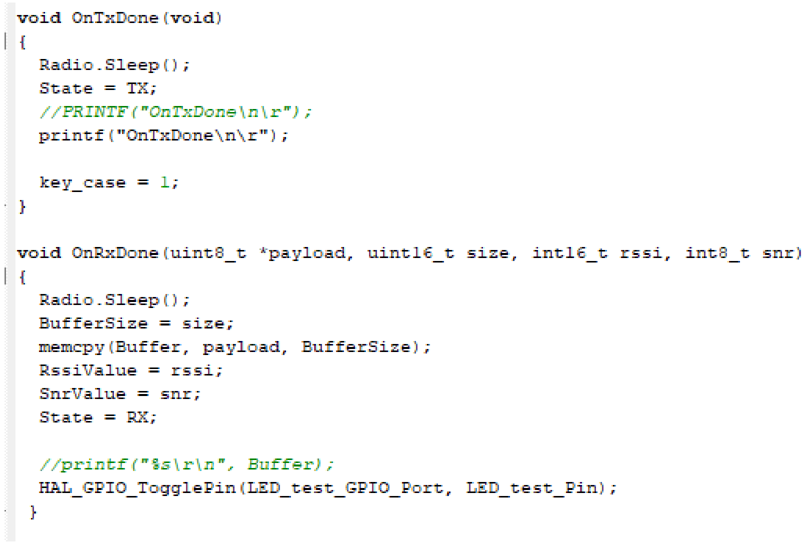
Transmitting and Receiving Data



```
void User_Receive(void)
{
    if(Time_Tick.time_3s = 1)
  {

  Time_Tick.time_3s = 0;
  }

     // uint16_t i;
     switch (key_case)
    //  switch (State)
     {
       case 0:
  if( BufferSize > 0 )
  {

         if((Buffer[0] =='#') && ((Buffer[1] == 'S'||Buffer[1] == 'Q')))
           {
           printf("%s\r\n", Buffer)
            Buffer[0] = '0';
            SEND_DATA2();
           }
     break;
      }
      case 1:
     //SEND_DATA1();
     Radio.Rx( RX_TIMEOUT_VALUE );
     key_case=0;
     break;
}
}


void OnTxTimeout(void)
{
  Radio.Sleep();
  State = TX_TIMEOUT;
  //PRINTF("OnTxTimeout\n\r");
}
```

Print the received data

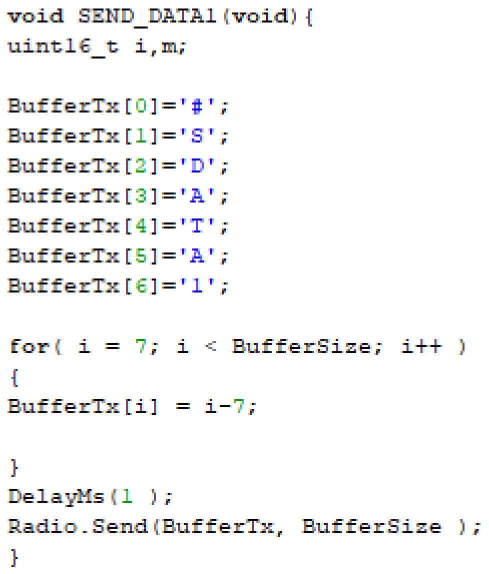
Setting the Transmitting and Receiving mode



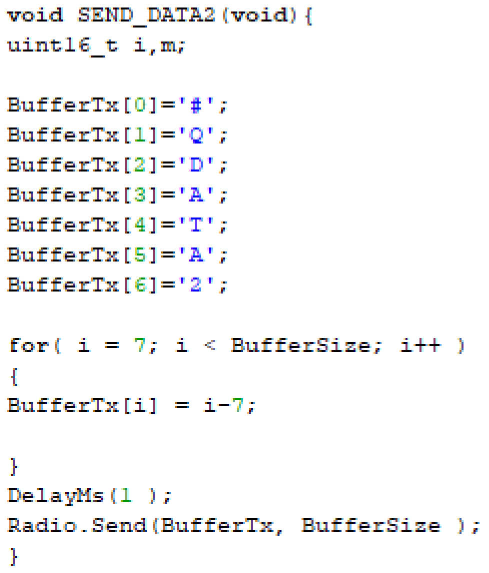
```
void OnTxDone(void)
{
  Radio.Sleep();
  State = TX;
  //PRINTF("OnTxDone\n\r");
  printf("OnTxDone\n\r");

  key_case = 1;
}

void OnRxDone(uint8_t *payload, uint16_t size, int16_t rssi, int8_t snr)
{
  Radio.Sleep();
  BufferSize = size;
  memcpy(Buffer, payload, BufferSize);
  RssiValue = rssi;
  SnrValue = snr;
  State = RX;

  //printf("%s\r\n", Buffer);
  HAL_GPIO_TogglePin(LED_test_GPIO_Port, LED_test_Pin);
 }
```

4. Coding and Coding description

SEND THE DATA STRING



```
void SEND_DATA1(void){
uint16_t i,m;

BufferTx[0]='#';
BufferTx[1]='S';
BufferTx[2]='D';
BufferTx[3]='A';
BufferTx[4]='T';
BufferTx[5]='A';
BufferTx[6]='1';

for( i = 7; i < BufferSize; i++ )
{
BufferTx[i] = i-7;

}
DelayMs(1 );
Radio.Send(BufferTx, BufferSize );
}
```

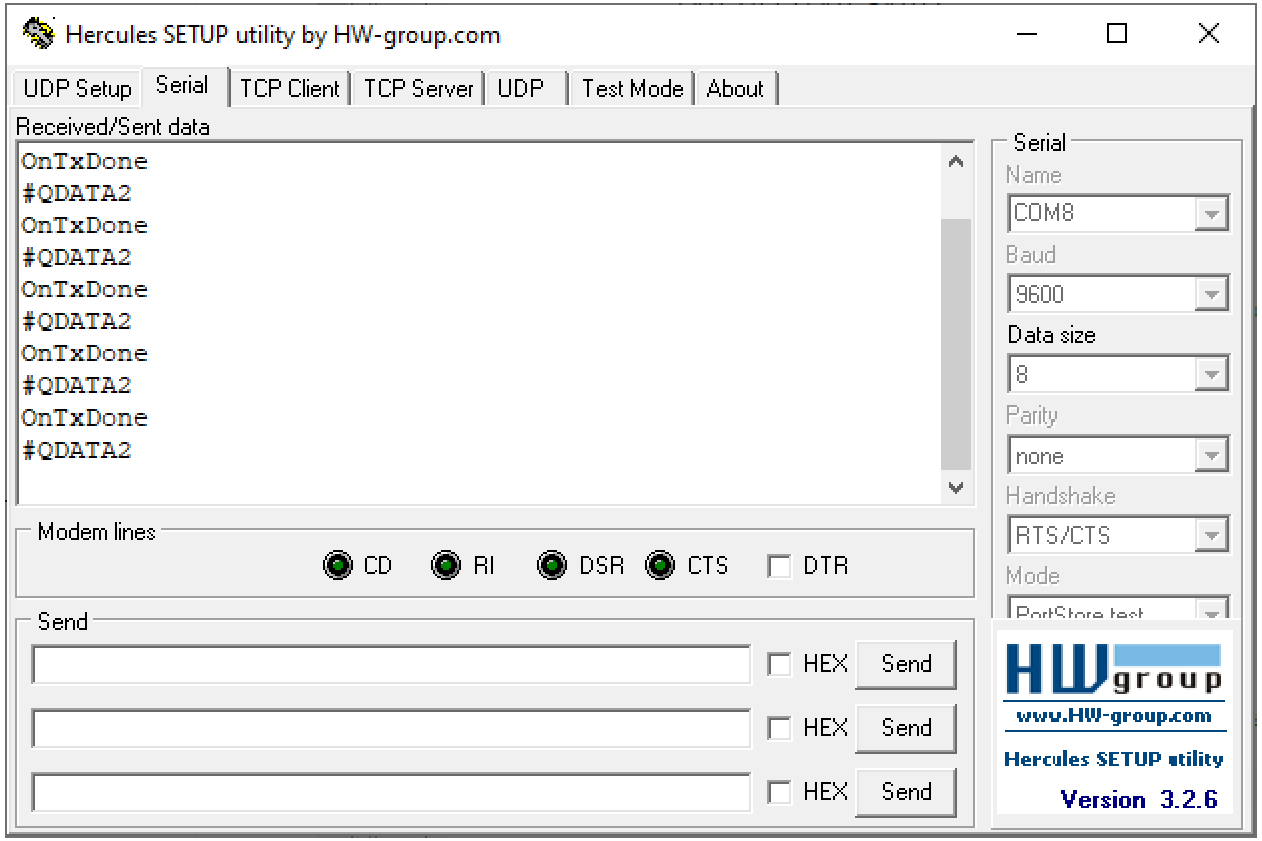


```
void SEND_DATA2(void){
uint16_t i,m;

BufferTx[0]='#';
BufferTx[1]='Q';
BufferTx[2]='D';
BufferTx[3]='A';
BufferTx[4]='T';
BufferTx[5]='A';
BufferTx[6]='2';

for( i = 7; i < BufferSize; i++ )
{
BufferTx[i] = i-7;

}
DelayMs(1 );
Radio.Send(BufferTx, BufferSize );
}
```

We fill the buffer with numbers for the payload

5. Test Result

Hardware and software set-up environment To set up the Lora board, connect it or the B-L072Z-LRWAN1 board to the computer with a Type-A to Mini-B USB cable to the CN1 ST-LINK connector. Ensure that the CN2 ST-LINK connector jumpers are ON. Refer to Figure for a representation of the data setup.



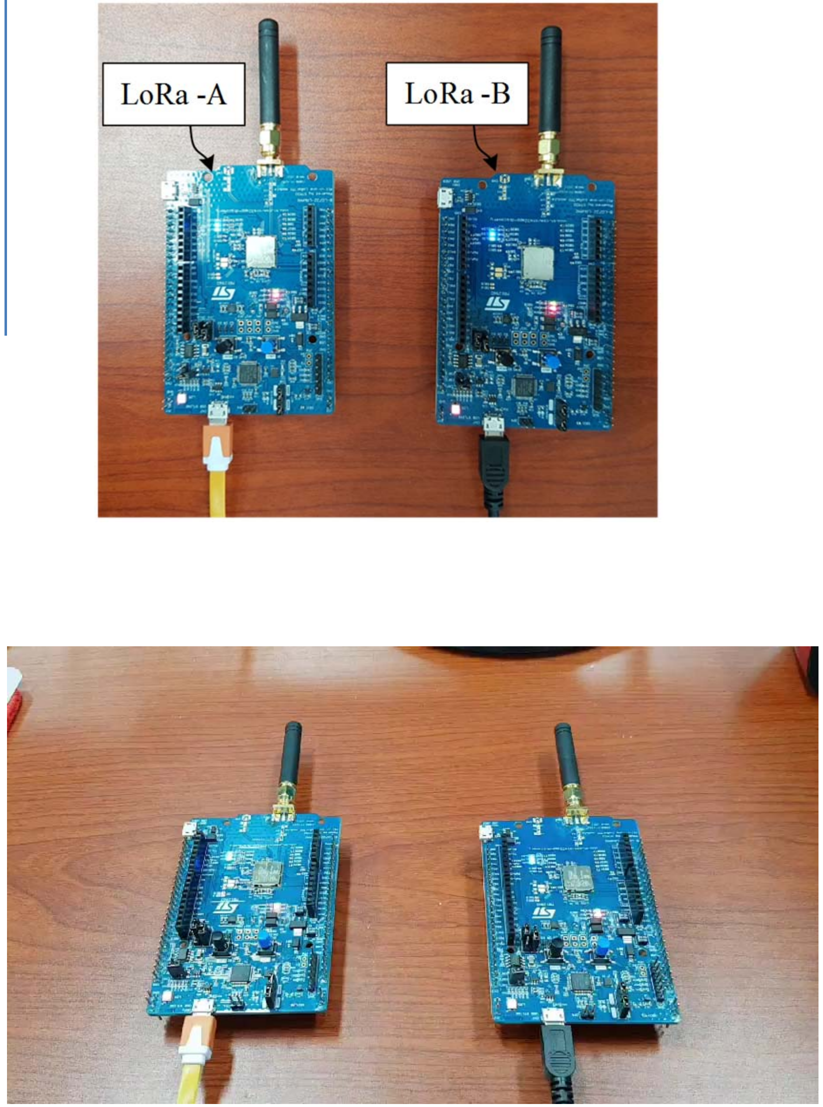
Received/Sent data

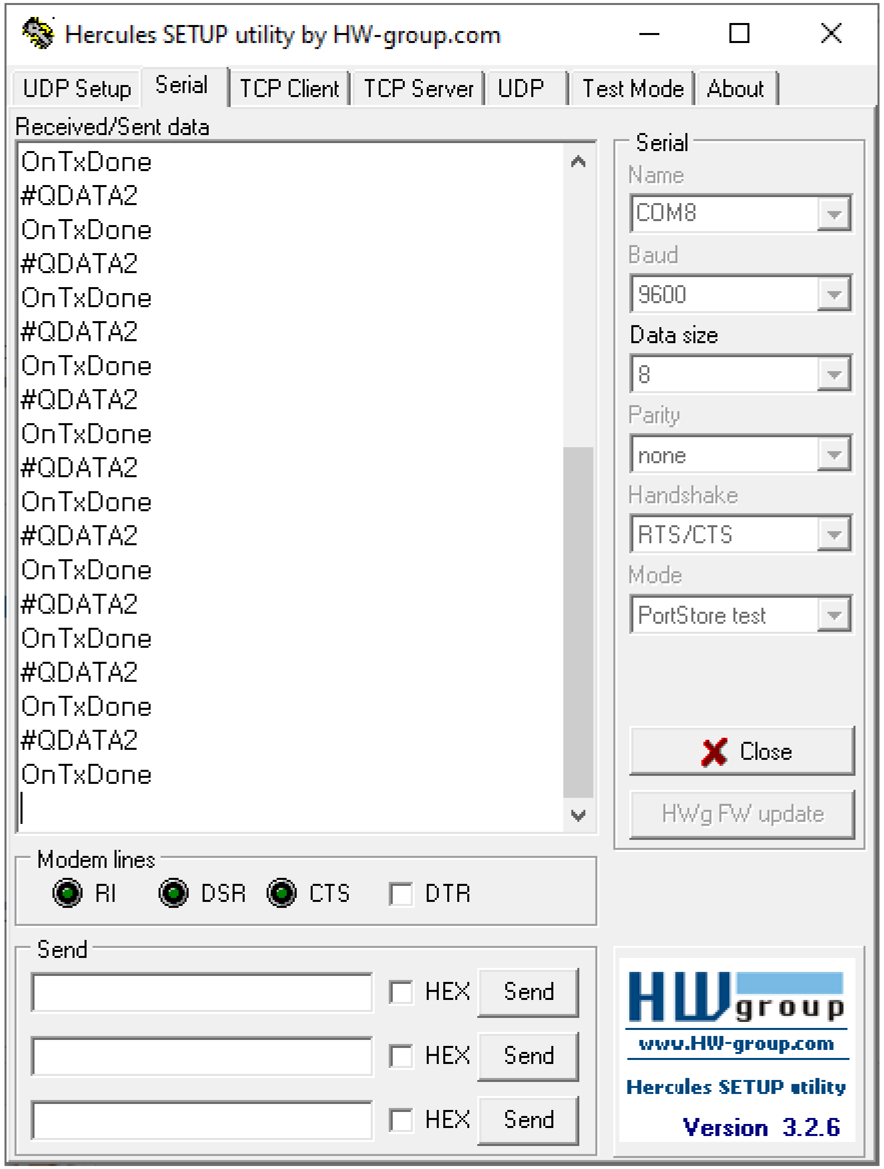
```
OnTxDone
#QDATA2
OnTxDone
#QDATA2
OnTxDone
#QDATA2
OnTxDone
#QDATA2
OnTxDone
#QDATA2
```

Serial
Name: COM8
Baud: 9600
Data size: 8
Parity: none
Handshake: RTS/CTS
Mode: PortStore test

Figure: Result of the Receiving data

5. Test Result





Received/Sent data

```
OnTxDone
#QDATA2
OnTxDone
#QDATA2
OnTxDone
#QDATA2
OnTxDone
#QDATA2
OnTxDone
#QDATA2
OnTxDone
#QDATA2
OnTxDone
#QDATA2
OnTxDone
#QDATA2
OnTxDone
#QDATA2
OnTxDone
```

Serial

Name: COM8

Baud: 9600

Data size: 8

Parity: none

Handshake: RTS/CTS

Mode: PortStore test

Modem lines: ● RI ● DSR ● CTS □ DTR

Send

□ HEX Send

□ HEX Send

□ HEX Send

Figure: Result of the Receiving data from LoRa-A