CONFIQURATION OF LORA (AS32-TTL-100)

1. Pin definition

Pin No.	Pin item	Pin direction	Pin application
1	M0	Input (weak pull-up)	Work with M1 & decide the four operating modes. Floating is not allowed, can be ground.
2	M1	Input (weak pull-up)	Work with M0 & decide the four operating modes. Floating is not allowed, can be ground.
3	RXD	Input	TTL UART inputs, connects to external (MCU, PC) TXD output pin. Can be configured as open-drain or pull-up input.
4	TXD	Output	TTL UART outputs, connects to external RXD (MCU, PC) input pin. Can be configured as open-drain or push-pull output
5	AUX	Output	To indicate module' s working status & wakes up the external MCU. During the procedure of self-check initialization, the pin outputs low level. Can be configured as open-drain output or push-pull output (floating is allowed).
6	VCC		Power supply 2.3V-5.5V DC
7	GND		Ground

2. Operating mode

Mode (0-3)	M1	M0	Mode introduction	Remark
Mode 0 Normal	0	0	UART and wireless channel is opened, transparent transmission is on.	The receiver must works in mode 0 or mode 1
Mode 1 Wake-up	0	1	UART and wireless channel is opened. The difference between normal mode and wake-up mode is it will add preamble code automatically before data packet transmission so that it can awaken the receiver works in mode 2.	The receiver can works in mode 0, mode 1 or mode 2.
Mode 2 Power-saving	1	0	UART is disabled. Wireless module works at WOR mode (wake on radio). It will open the UART and transmit data after receive the wireless data.	1,the transmitter must works in mode 1 2,transmitting is not allowed in this mode
Mode 3 Sleep	1	1	Parameter setting.	

3. Instruction format

In sleep mode (mode 3: M1=1, M0=1), it supports below instruction on list.

(Only support 9600 and 8N1 format when setting)

No.	Instruction format	Illustration
1	C0 + working parameters	C0 + 5 bytes working parameters are sent in hexadecimal format. 6 bytes in total and must send in succession. (Save the parameters when power-down)
2	C1 C1 C1	Three C1 are sent in hexadecimal format. The module returns the saved parameters and must send in succession.
3	C2 + working parameters	C2 + 5 bytes working parameters are sent in hexadecimal format. 6 bytes in total and must send in succession. (Not save the parameters when power-down)
4	C3 C3 C3	Three C3 are sent in hexadecimal format. The module returns the version information and must send in succession.
5	C4 C4 C4	Three C4 are sent in hexadecimal format. The module will reset one time and must send in succession.

4. Parameter setting instruction

The difference between C0 command and C2 command is that C0 command will write parameters into the internal flash memory and can be saved when power down, while C2 command cannot be saved when power down, because C2 command is temporarily mend instruction.

C2 is recommend for the occasion that need to change the operating parameters frequently,

Like C2 00 00 1A 17 44.

No.	Item	Description	Remark
0	HEAD	Fix 0xC0 or 0xC2, it means this frame data is control command	Must be 0xC0 or 0xc2 C0: Save the parameter when power-down C2: not save the parameter when power-down
1	ADDH	High address byte of module (the default 00H)	00H-FFH
2	ADDL	Low address byte of module (the default 00H)	00H-FFH
3	SPEED	Rate parameter, including UART baud rate and air date rate 7,6 UART parity bit 00: 8N1 (default) 01: 8O1 10: 8E1 11: 8N1 (equal to 00) --- 5,4,3 TTL UART baud rate (bps) 000: 1200 001: 2400 010: 4800 011: 9600 (default) 100: 19200 101: 38400 110: 57600 111: 115200 --- 2,1,0 Air data rate (kbps)	• UART mode can be different between communication parties --- • UART baud rate can be different between communication parties • The UART baud rate has nothing to do with wireless transmission parameters & will not affect the wireless transmit / receive feature. ---

		000: 0.3 001: 1.2 010: 2.4 (default) 011: 4.8 100: 9.6 101: 19.2 110: 19.2 111: 19.2	• The lower air date rate the longer the transmitting distance, better anti-interence performance and longer transmitting time • Must keep the same for both communication parties.
4	CHAN	7,6,5 N / A --- 4-0: Communication channel, default 17H (434Hz)	• 0 recommended --- • 00H-1FH
5	OPTION	7, Fixed transmission (similar to MODBUS) 0: Transparent transmission mode 1: Fixe transmission mode --- 6, IO drive mode (default 1) 1: TXD and AUX push-pull output, RXD pull-up input 0: TXD, AUX open collector Output, RXD open-collector inputs --- 5,4,3 wireless wakes up time 000: 250ms (default) 001: 500ms 010: 750ms 011: 1000ms 100: 1250ms 101: 1500ms 110: 1750ms 111: 2000ms --- 2, FEC switch 0: turn off FEC 1; turn on FEC --- 1,0 transmission power 00: 20dBm 01: 17dBm 10: 14dBm 11: 10dBm	

➢ **Example of configuration**

Value speed 9600,8N1, 2,4 KBps	1A
ADDRESS 25, CH 15, 17dBm ADDRESS 10, CH 10, 11dBm	C0 00 19 1A 1F 45 C0 00 10 1A 0A 47
Value speed 9600,8N1, 2,4 KBps	1B
ADDRESS 25, CH 15, 14dBm ADDRESS 10, CH 10, 20dBm	C0 00 19 1B 1F 46 C0 00 10 1B 0A 44

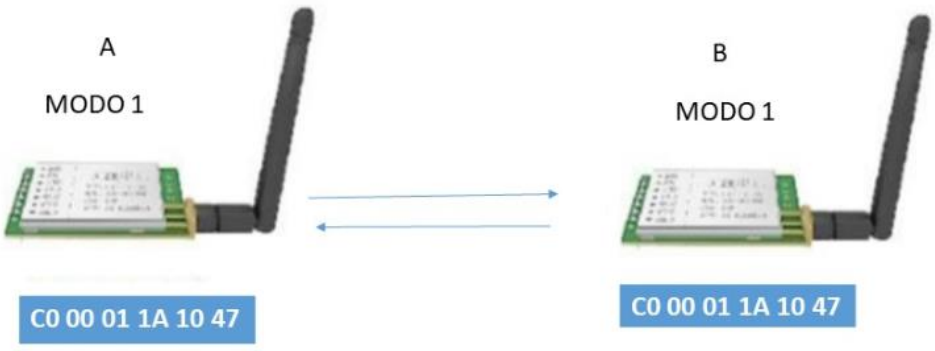
➢ **Experiment**

Parameter Configuration in STM32cubeide

```
/* USER CODE BEGIN PV */
uint8_t HEAD         = 0xC0;
uint8_t ADDH         = 0x00;
uint8_t ADDL         = 0x01;
uint8_t SPEED        = 0x1A;
uint8_t CHAN         = 0x10;
uint8_t OPTION       = 0x47;
uint8_t cmd_txBuffer[6];
uint8_t LoRa_rx[6]   = {0};
/* USER CODE END PV */

/* Private function prototypes -----------------------------------*/
void SystemClock_Config(void);
/* USER CODE BEGIN PFP */
void cmd_send(){
cmd_txBuffer[0] = HEAD;
cmd_txBuffer[1] = ADDH;
cmd_txBuffer[2] = ADDL;
cmd_txBuffer[3] = SPEED;
cmd_txBuffer[4] = CHAN;
cmd_txBuffer[5] = OPTION;
HAL_UART_Transmit(&huart1, cmd_txBuffer, sizeof(cmd_txBuffer), 100);
}
/* USER CODE END PFP */

/* USER CODE BEGIN 2 */
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_8, 1);
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_9, 1);
cmd_send();
HAL_Delay(2);
HAL_UART_Receive(&huart1, &LoRa_rx, sizeof(LoRa_rx), 100);
/* USER CODE END 2 */
```

Note: If the module response is **OK,** the configuration is successful.

Transparent transmission mode



Transmitter (transparent transmission mode)

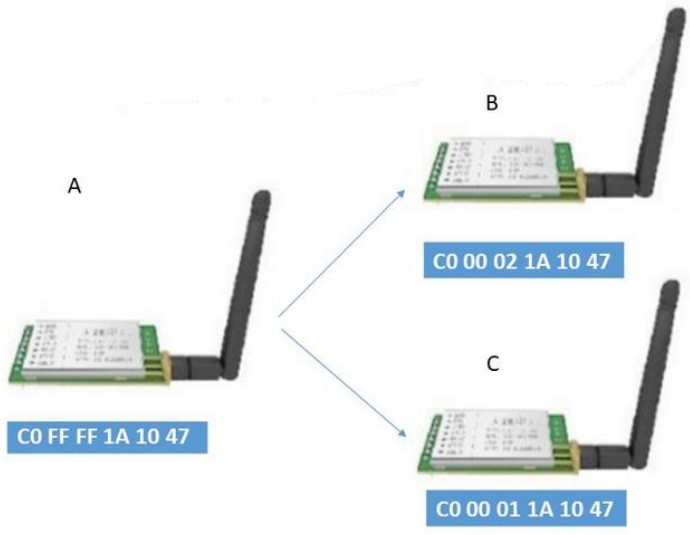
```
/* USER CODE BEGIN PV */
uint8_t ADDH         = 0x00;
uint8_t ADDL         = 0x01;
uint8_t CHAN         = 0x10;
uint8_t DATA1        = 10;
uint8_t DATA1        = 11;
uint8_t DATA1        = 12;
uint8_t cmd_txBuffer[6];
/* USER CODE END PV */

/* Private function prototypes -----------------------------------
------------*/
void SystemClock_Config(void);
/* USER CODE BEGIN PFP */
void cmd_send(){
cmd_txBuffer[0] = ADDH;
cmd_txBuffer[1] = ADDL;
cmd_txBuffer[2] = CHAN;
cmd_txBuffer[3] = DATA1;
cmd_txBuffer[4] = DATA2;
cmd_txBuffer[5] = DATA3;
HAL_UART_Transmit(&huart1, cmd_txBuffer,6, 100);
}
/* USER CODE END PFP */
```

Receiver

```
HAL_UART_Receive_DMA(&huart1, LoRa_rx, 10);
HAL_Delay(250);
```

Note: For the receiver, we used **DMA** to get all data corrections.

Broadcast in Point-to-point Transmission



Transmitter (Broadcast in Point-to-point Transmission)

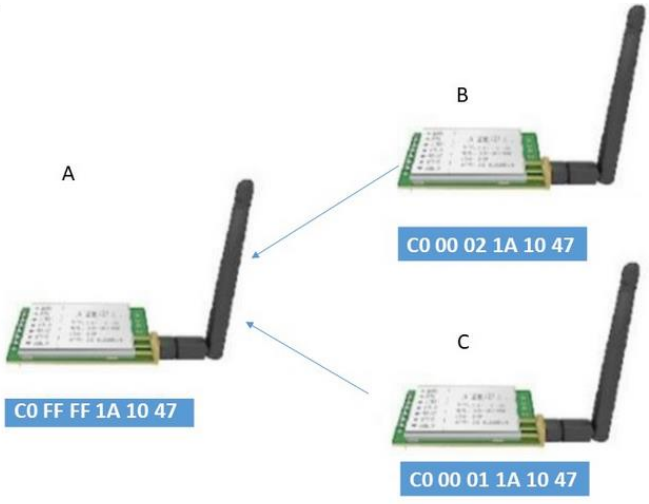
```
/* USER CODE BEGIN PV */
uint8_t ADDH         = 0xFF;
uint8_t ADDL         = 0xFF;
uint8_t CHAN         = 0x10;
uint8_t DATA1        = 10;
uint8_t DATA1        = 11;
uint8_t DATA1        = 12;
uint8_t cmd_txBuffer[6];
/* USER CODE END PV */

/* Private function prototypes -----------------------------------
------------*/
void SystemClock_Config(void);
/* USER CODE BEGIN PFP */
void cmd_send(){
cmd_txBuffer[0] = ADDH;
cmd_txBuffer[1] = ADDL;
cmd_txBuffer[2] = CHAN;
cmd_txBuffer[3] = DATA1;
cmd_txBuffer[4] = DATA2;
cmd_txBuffer[5] = DATA3;
HAL_UART_Transmit(&huart1, cmd_txBuffer,6, 100);
}
/* USER CODE END PFP */
```

Receiver

```
HAL_UART_Receive_DMA(&huart1, LoRa_rx, 10);
HAL_Delay(250);
```

Note: The first 3 bytes of the transmitter's data must be 0xFF + 0xFF + receiver channel.

Monitor in Point-to-point Transmission



Transmitter (Monitor in Point-to-point Transmission)

```
/* USER CODE BEGIN PV */
uint8_t ADDH        = 0x68;
uint8_t ADDL        = 0x76;
uint8_t CHAN        = 0x10;
uint8_t DATA1       = 10;
uint8_t DATA1       = 11;
uint8_t DATA1       = 12;
uint8_t cmd_txBuffer[6];
/* USER CODE END PV */

/* Private function prototypes -----------------------------------
------------*/
void SystemClock_Config(void);
/* USER CODE BEGIN PFP */
void cmd_send(){
cmd_txBuffer[0] = ADDH;
cmd_txBuffer[1] = ADDL;
cmd_txBuffer[2] = CHAN;
cmd_txBuffer[3] = DATA1;
cmd_txBuffer[4] = DATA2;
cmd_txBuffer[5] = DATA3;
HAL_UART_Transmit(&huart1, cmd_txBuffer,6, 100);
}
/* USER CODE END PFP */
```

Receiver

```
HAL_UART_Receive_DMA(&huart1, LoRa_rx, 10);
HAL_Delay(250);
```

➢ **Note**

- The address of the monitor module must be set to 0xFFFF.
- Normal mode.
- The channel of the monitor module and the transmitter must be the same.
- The first 3 bytes of the sending data must be 0xXX+0xXX+ monitor channel.