Standard station：

Baud rate：2400bps

Data length: 8 bit

Parity check bit: no

Stop bit: 1 bit

RTU mode

The controller is working with RTU mode, 8 bites and binary

Modbus-RTU air conditioner protocol

①data transmission method：

Async 10 bits——1 bit starting，8 bits data bits，1 bit stop，no parity check bit.

②data baud rate:

2400BPS。

③address：

1～220，user could set the air conditioner address number by setting remote control, it would be remembered in EEPROM.

④ read data

A： PC master request to read data format：

Master sending	byte	Sending info（HEX）	remark
Slave address	1	XX	Request the data from slave address XX
Function code	1	03	Read register
Starting address	2	0000	Starting address 0000（high byte is front, and low byte is back）
Data length	2	0005	Read 5 data（total 10 bytes）（high byte is front, and low byte is back）
CRC code	2	XXXX	Got CRC code from master, first is low byte, later is high byte

B: Air conditioner slave feedback format:

Slave feedback	byte	Feedback info (HEX)	remark
Slave address	1	XX	From the slave address XX
Function code	1	03	Read register
Data length	1	0A	10 bytes,
Register data 1	2	DAT1	Data 1（high byte is front, and low byte is back）
…	…	…	…
Register data 5	2	DAT5	Data N （high byte is front, and low byte is back）
CRC code	2	XXXX	Got CRC code from slave, first is low byte, later is high byte

C: Air conditioner slave' s feedback is unnormal:

Slave feedback	byte	Feedback info (HEX)	remark
Slave address	1	XX	From the slave address XX
Function code	1	83	Read register
Data code	1	EX	Error code
CRC code	2	XXXX	Got CRC code from slave, first is low byte, later is high byte

Data during monitoring:

Register no.	content	
0000H	Slave setting temperature	
0001H	Slave room temperature	
0002H	keep	
0003H	keep	
0004H	BIT0	1: slave open 0: slave close
	BIT1	1: slave compressor open 0: slave compressor close
	BIT2	1: slave electrical heater open 0: slave electrical heater close
	BIT3	Slave motor speed indication
	BIT4	00 – 01: low 10: mid 11: high
	BIT5	Current slave working mode
	BIT6	000 auto 001: cool 010: dry
	BIT7	011: heat 100: fan
	BIT8	Timer on function: 1: yes 0: no
	BIT9	Timer off function: 1: yes 0: no
	BIT10	Lock setting: 1: lock 0: unlock
	BIT11	keep
	BIT12	keep
	BIT13	keep
	BIT14	keep
	BIT15	keep

⑤ write data (when it is in broadcast method, slave address is 00; refer standard MODBUS protocol) ;

A: PC master request to write data format:

Master sending	byte	Sending info (HEX)	remark
Slave address	1	XX	Write the data to slave address XX
Function code	1	10	Write register
Starting address	2	0000	Starting address 0000 (high byte is front, and low byte is back)
Data length	2	000A	5 data (total 10 bytes) (high byte is front, and low byte is back)
Register 1	2	DAT1	Data 1 (high byte is front, and low byte is back)
…	…	…	…
Register 5	2	DAT5	Data N (high byte is front, and low byte is back)
CRC code	2	XXXX	Got CRC code from master, first is low byte, later is high byte

B: Air conditioner slave feedback format:

Slave feedback	byte	Feedback info (HEX)	remark
Slave address	1	XX	From the slave address XX
Function code	1	10	write register
Starting address	2	0000	Starting address 0000 (high byte is front, and low byte is back)
Data length	2	0005	5 data (total 10 bytes) (high byte is front, and low byte is back)
CRC code	2	XXXX	Got CRC code from slave, first is low byte, later is high byte

C: Air conditioner slave' s feedback is unnormal:

Slave feedback	byte	Feedback info (HEX)	Remark
Slave address	1	XX	From the slave address XX
Function code	1	90	write register
Data code	1	EX	Error code
CRC code	2	XXXX	Got CRC code from slave, first is low byte, later is high byte

Write command data structure:

Register no.	content	
0000H	master setting temperature	
0001H	Master PC Timer on setting with Minute. No Timer on setting is 0;	
0002H	Master PC Timer off setting with Minute. No Timer off setting is 0;	
0003H	keep	
0004H	BIT0	1: open 0: close
	BIT1	keep
	BIT2	1: Open electrical heater 0: close electrical heater
	BIT3	Motor speed setting
	BIT4	00 – 01: low 10: mid 11: high
	BIT5	Working mode setting
	BIT6	000 auto 001: cool 010: dry
	BIT7	011: heat 100: fan
	BIT8	Timer on function: 1: yes 0: no
	BIT9	Timer off function: 1: yes 0: no
	BIT10	Lock setting: 1:lock 0:unlock
	BIT11	keep
	BIT12	keep
	BIT13	keep
	BIT14	keep
	BIT15	keep

⑥ error code:

E1: room temperature sensor problem E2: indoor coil sensor problem

E4: indoor PG motor problem E4: lack refrigerant

E5: No feedback overtime E6: Master send wrong info

E7: outdoor coil sensor problem

● CRC code calculation method：

Simple CRC function is as below ：

```
unsigned short CRC16(puchMsg, usDataLen)
unsigned char *puchMsg ; /* 要进行CRC校验的消息 */
unsigned short usDataLen ; /* 消息中字节数 */
{
unsigned char uchCRCHi = 0xFF ; /* 高CRC字节初始化 */
unsigned char uchCRCLo = 0xFF ; /* 低CRC 字节初始化 */
unsigned uIndex ; /* CRC循环中的索引 */
while (usDataLen--) /* 传输消息缓冲区 */
{
uIndex = uchCRCHi ^ *puchMsgg++ ; /* 计算CRC */
uchCRCHi = uchCRCLo ^ auchCRCHi[uIndex} ;
uchCRCLo = auchCRCLo[uIndex] ;
}
return (uchCRCHi << 8 | uchCRCLo) ;
}
/* CRC 高位字节值表 */
static unsigned char auchCRCHi[] = {
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
```

```
0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40
} ;
/* CRC低位字节值表*/
static char auchCRCLo[] = {
0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06,
0x07, 0xC7, 0x05, 0xC5, 0xC4, 0x04, 0xCC, 0x0C, 0x0D, 0xCD,
0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09,
0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A,
0x1E, 0xDE, 0xDF, 0x1F, 0xDD, 0x1D, 0x1C, 0xDC, 0x14, 0xD4,
0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,
0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3,
0xF2, 0x32, 0x36, 0xF6, 0xF7, 0x37, 0xF5, 0x35, 0x34, 0xF4,
0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A,
0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38, 0x28, 0xE8, 0xE9, 0x29,
0xEB, 0x2B, 0x2A, 0xEA, 0xEE, 0x2E, 0x2F, 0xEF, 0x2D, 0xED,
0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60,
0x61, 0xA1, 0x63, 0xA3, 0xA2, 0x62, 0x66, 0xA6, 0xA7, 0x67,
0xA5, 0x65, 0x64, 0xA4, 0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F,
0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68,
0x78, 0xB8, 0xB9, 0x79, 0xBB, 0x7B, 0x7A, 0xBA, 0xBE, 0x7E,
0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,
0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71,
0x70, 0xB0, 0x50, 0x90, 0x91, 0x51, 0x93, 0x53, 0x52, 0x92,
0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C,
0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B,
0x99, 0x59, 0x58, 0x98, 0x88, 0x48, 0x49, 0x89, 0x4B, 0x8B,
0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42,
0x43, 0x83, 0x41, 0x81, 0x80, 0x40 } ;
```