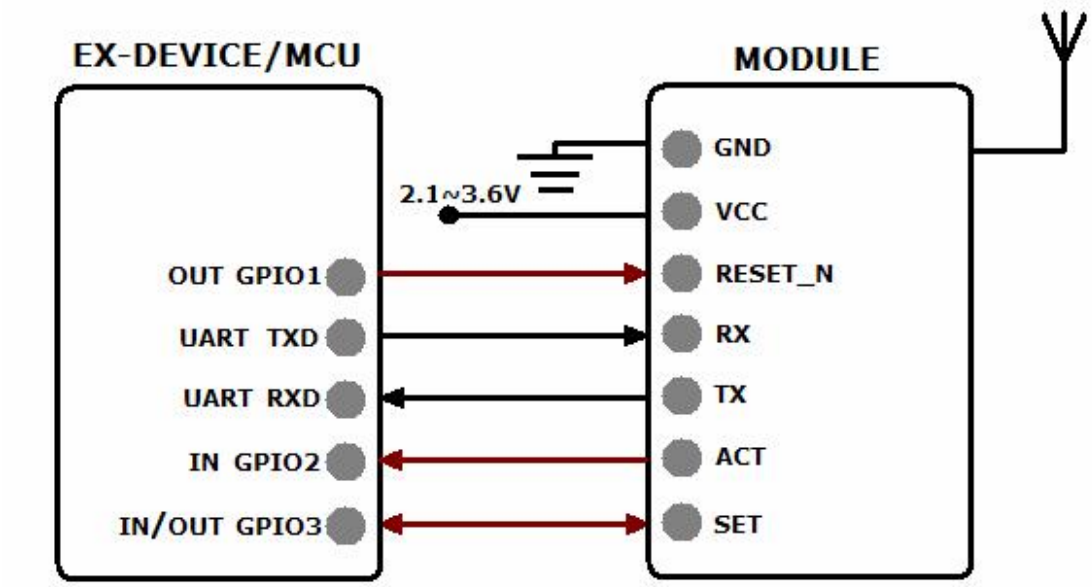
配置模式下模块控制块的读写

1．摘要

本文描述如何在配置模式下对 **Wave Mesh AMR** 协议系列模块的参数控制块进行读取和修改，通过这种方式可以不需要借助模块 **PC** 配置工具实现按照在线的方式对模块进行配置。模块控制块包含了模块所有可以配置的参数和模块的基本信息。通过在线控制块参数的修改可以实现远程无线、点对点红外等方式对无线网络中的节点进行管理的目的。

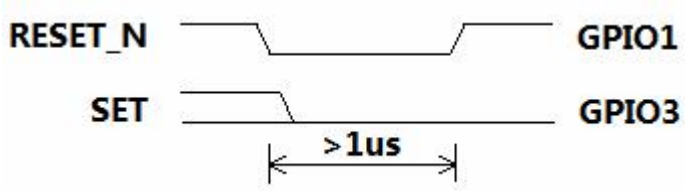
2．配置模式

模块在上电时通过检测 **SET** 引脚的电平的高低，若该引脚为逻辑低则会选择进入配置模式，否则会进入正常工作模式。配置模式主要是为了完成模块控制块的读取和修改，在配置模式下会关闭模块的射频部分等功能。无线模块和外设 **MCU** 的连接示意图如下所示：



可以通过控制无线模块的 **RESET_N** 和 **SET** 引脚来实现模块工作模式的选择。其中，**RESET_N** 为模块的输入复位引脚，低有效；**SET** 引脚会在模块上电、复位一瞬间自动设置为输入引脚并有弱上拉。若 **SET** 引脚此时被拉低模块会进入配置模式，进入配置模式后该引脚一直会保持为输入引脚；若 **SET** 引脚在模块上电、复位为高电平，则会进入正常工作模式，该引脚会被作为输出引脚使用用来指示模块的内部的缓冲区的空闲状态。

设置模块为配置模式的时序示意图如下所示：



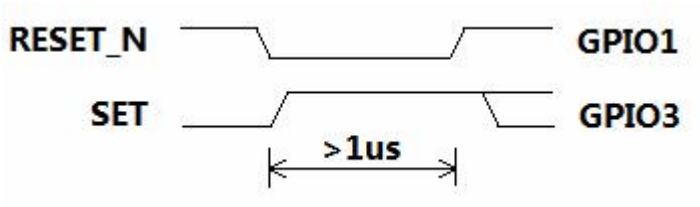
操作步骤如下：

1. 将连接 RESET_N 引脚的 GPIO1 拉低；

2. 将连接 SET 引脚的 GPIO3 设置为输出模式，并将其拉低；

3. 将连接 RESET_N 引脚的 GPIO1 拉高。

设置模块为正常工作模式的时序示意图如下所示：



操作步骤如下：

1. 将连接 RESET_N 引脚的 GPIO1 拉低；

2. 将连接 SET 引脚的 GPIO3 设置为带上拉的输入模式；

3. 将连接 RESET_N 引脚的 GPIO1 拉高。

说明：建议在配置模式下连接 SET 引脚的 GPIO3 设置为输出引脚并且拉低；在正常工作模式下连接 SET 引脚的 GPIO3 设置为带上拉的输入引脚。

3．模块控制块读写

在配置模式下可以通过模块的 UART 接口发送 AT 命令访问和修改模块的控制块。在配置模式下模块的 UART 口的波特率为 115200，没有校验位，1 比特停止位，每字节 8 比特位。注意，在配置模式下 AT 命令后的参数为二进制+CRC 校验的方式；而在正常工作模式下的 AT 指令后的参数为没有校验的 ASCII 字符。控制块的大小为 512 字节，读写时需要一次性读出和写入 512 字节的数据，数据的完整性由 CRC 校验来保证。目前该控制块仅有 300 多字节被赋予有意义的数据，剩余的为保留字节。为保证模块控制块数据的完整性，在控制块的写入时需要将读出的控制块保留字节的内容原封不动地写入控制块。

控制块的读取命令为“ATC0”没有参数；模块的响应为固定长度的“ATC0”+“512字节控制块”+“2字节CRC校验”。
控制块的写入命令为“ATW0”+“512字节控制块”+“2字节CRC校验”；模块的响应“ATOK”代表写入成功，“ATER”代表写入失败。

其中 CRC 校验仅对 512 字节的控制块进行计算，不包括数据帧头“ATC0”和“ATW0”；CRC 的生成多项式为 CRC16 ($x^{16} + x^{15} + x^2 + 1$)，CRC 寄存器的初始的各比特位全是 1。另外，CRC 的低字节在前，高字节在后。CRC 的实现代码如下：

```
uint16 CRCCalc (uint8* p, int len)
{
```

```
    uint16 crc = 0xFFFF;

    for (int i = 0; i < len; i++) {
        uint8 data = *p++;
        for (int j = 0; j < 8; j++) {
            if (((crc & 0x8000) >> 8) ^ (data & 0x80))
                crc = (crc << 1) ^ 0x8005;
            else
                crc <<= 1;
            data <<= 1;
        }
    }

    return crc;
}
```

由于某些硬件原因 UART 产生误码，会导致模块不响应的情况，为提高系统的稳定性，需要对控制块的读写增加超时处理。控制的写入响应时间会略大于读取的时间，建议将控制块读写命令的响应超时时间设置为 50ms，串口字符间隔最大时间为 1ms。

模块通过以下几个方面对写入的控制块完整性进行检查：

1. UART 的数据没有误码和错误；
2. 控制块帧头 4 个字节“ATW0”正确；
3. 控制块的长度正确；
4. 控制块的 CRC 正确；
5. 写入 FLASH 后的控制块逐个字节进行比对，确定 512 个字节都写入成功。

注意，模块并没有对控制块的实际参数的合理性进行检查。AT 命令的“ATC0”和“ATW0”为 4 个字节的 ASCII 字符，而其后的 512 字节的控制块和 2 字节的 CRC 校验为二进制数据。

4. 控制块数据结构

模块控制块的数据结构（V1.7）如下所示，该数据结构在今后的升级版本中可能会有变化，但会保证和之前的版本兼容。

```
typedef struct {
    /* module name */
    uint8 name[32];

    /* version */
    uint8 version0;
    uint8 version1;
```

```
/* 6-byte MAC address */
uint8 macID0;
uint8 macID1;
uint8 macID2;
uint8 macID3;
uint8 macID4;
uint8 macID5;

/* 3-byte subnet address */
uint8 netID0;
uint8 netID1;
uint8 netID2;

/* the frequency of channel 0 */
uint8 freq0;
uint8 freq1;
uint8 freq2;

/* the default RF baud */
uint8 baud;

/* the highest RF baud can be used */
uint8 baudUp;

/* the lowest RF baud can be used */
uint8 baudDown;

/* set the CS absolute threshold */
int8 csAbsThreshold;

/* the main channel number */
uint8 chnMain;

/* the start channel number for aux */
uint8 chnAuxStart;

/* data channel interval */
uint8 chnInterval;

/* channel number mask */
uint8 chnMask;

/* output RF power */
uint8 paIndex;

/* sleep enabled or not */
uint8 sleepEnable;
```

```
/* how many devices controlled by the module */
uint8 devNum;

/* enable flow control is the speed of RF and Uart mismatch */
uint8 flowCtrl;

/* sleep-wake rate */
uint16 swRate;

/* host init time (uint:ms) */
uint16 hostInitTime;

/* command no response timeout (uint:ms) */
uint16 noResponse;

/* PWM and other condifgure */
uint8 pwmMode;

/* uart configuration */
uint8 uartBaud;
uint8 uartGCR;
uint8 uartUCR;

/* flags0 */
uint8 Flags0;

/* uart sync word */
uint8 uartStart;
uint8 uartEnd;

/* bin or ACII mode */
uint8 uartMode;

/* host address length */
uint8 hostIdLen;

/* host address offest */
uint8 hostIdOffD;
uint8 hostIdOffU;

/* wildcard of address used for broadcast and multicast */
uint8 idWildcard0;
uint8 idWildcard1;

/* packet length offset */
uint8 pktLenOff;

/* packet head and tail size */
```

```
uint8 pktLenAdd;

/* if it is necessary to check pkt end */
uint8 pktEndCheck;

/* tick configure */
uint8 t3CC0;
uint8 t3CTL;

/* wait ticks before going to sleep */
uint16 sleepWait;

/* Rts timeout before giveup */
uint16 rtsTimeout;

/* req send times before giveup */
uint16 reqTimes;

/* repeat time sending sync packets */
uint16 sycRepeat;

/* ticks of beacon stage */
uint16 beaconTicks;

/* ticks of wake slot */
uint16 wakeSlotTicks;

/* Event for CC1110 PM2 */
uint16 pm2Events;

/* WORCTRL for CC1110 PM2 */
uint8 pm2WORCTRL;

/* extern time unit */
uint8 externUnit;

/* interval time for sync mode period */
uint16 intervalSync;

/* enable periods auto-send or not */
uint8 repeatEn;

/* length of the ACK pcket */
uint8 ackLen;

/* length of cmd to get host address information */
uint8 getAddCmdLen;
```

```
    /* length of cmd to read data from the host */
    uint8 cmd0Len;
    uint8 cmd1Len;

    /* the ack packet of handshake */
    uint8 ackCmd[48];

    /* the command to get host address information */
    uint8 getAddCmd[48];

    /* the command to read data from the host */
    uint8 cmd0[48];
    uint8 cmd1[48];
} cfgBlock;
```

由于控制块中的很多参数需要计算才能得到正确的值，而且有些参数是相关的，可以参考配置工具的源码，在此就不做太多说明。**对于，控制块中的某些参数的修改，建议先采用 PC 配置工具对参数进行修改写入模块，将不同配置下的模块控制块读取出来。通过比较的方式记录配置块差异的部分，在需要在线修改控制块时将事先得到的相应的控制块写入即可。**这样可以不需要对控制块的具体参数意义进行了解，也可以避免由于对参数的计算错误和误差导致的模块不能正常工作。