正常工作模式下模块控制块的修改

1．摘要

为了简化对模块控制块的修改，在模块 1.7 版本以后的固件增加了工作模式下修改模块控制块的 AT 指令。用户可以在不必切换到配置模式就可以在线修改模块的配置参数。值得注意的是工作模式下只能一次修改模块控制块的部分内容，然而控制块将通常需要在线修改的配置参数如频点、信道号、网络 ID、MAC 地址等放在控制块最开始的部分，因此一条 AT 指令就可以实现对控制块的关键参数修改。本文主要描述了在正常工作模式下通过 AT 指令对模块控制块参数的修改，对于在配置模式下模块控制块的修改请参考《应用笔记-配置模式下模块控制块的读写》。

2．AT 指令格式

在正常工作模式下，不管用户怎样配置其应用层数据帧解析结构和 UART 接口的波特率，AT 指令的格式是固定的：

"AT" 前缀 + ASCII 命令 + 空格（可选）+ 参数（可选，十六进制 ASCII 字符）+ 回车符<CR>

AT 指令通常为 4 个 ASCII 字符，如“ATCI”、“ATCC”；参数为 ASCII 字符表示的十六进制数，数字字符为‘0’~‘9’、‘A’~‘F’以及‘a’~‘f’；AT 指令的长度受到模块输入数据缓冲区大小 256 字节的限制，因此最大的 AT 指令总长度为 253 字节。

对于仅有 1 个参数的 AT 指令来说，参数可以添加十六进制数前缀“0x”，也可以省略该前缀，如“ATCS 0x2<CR>”；“ATCS 0x02<CR>”；“ATCS02<CR>”；“ATCS 2<CR>”。

对于多个参数的的 AT 指令来说，必须省略参数前的十六进制数前缀“0x”，如：“ATCC A B2 C3<CR>”；“ATCC 0A B2 C3<CR>”。为了使指令变得更加紧凑，可以省略参数之间的空格，这种格式的每个字节的参数必须由 2 个字节的 ASCII 表示，也就是说值为 0x2 和 0xA 的参数，必须在之前补 0 使其变成 2 个字节 02，0A：“ATCC**02**<CR>”，“ATCC**0A**B2C3<CR>”。 由于受到缓冲区 256 字节大小限制，模块能接收的 AT 指令长度最长为 253 字节，按照这种紧凑的命令格式最多能携带的参数个数为 124 字节（每 2 字节 ASCII 字符表示 1 个字节参数）。因此，在正常工作模式下利用 AT 指令一次性能最多能修改 124 个字节的控制块的内容。

其中，<CR>是回车符，对应的字符十进制值为 13，十六进制的值为 0xD。

3．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。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。

```
typedef struct {
    /* module name - offset 0 */
    uint8 name[32];

    /* version - offset 32 */
    uint8 version0;
    uint8 version1;

    /* 6-byte MAC address - offset 34 */
    uint8 macID0;
    uint8 macID1;
    uint8 macID2;
    uint8 macID3;
    uint8 macID4;
    uint8 macID5;

    /* 3-byte subnet address - offset 40 */
    uint8 netID0;
    uint8 netID1;
    uint8 netID2;

    /* the frequency of channel 0 - offset 43 */
    uint8 freq0;
    uint8 freq1;
    uint8 freq2;

    /* the default RF baud - offset 46 */
    uint8 baud;

    /* the highest RF baud can be used - offset 47 */
    uint8 baudUp;

    /* the lowest RF baud can be used - offset 48 */
    uint8 baudDown;

    /* set the CS absolute threshold - offset 49 */
    int8 csAbsThreshold;

    /* the main channel number - offset 50 */
    uint8 chnMain;

    /* the start channel number for aux - offset 51 */
    uint8 chnAuxStart;
```

```
/* data channel interval - offset 52 */
uint8 chnInterval;

/* channel number mask - offset 53 */
uint8 chnMask;

/* output RF power - offset 54 */
uint8 paIndex;

/* sleep enabled or not - offset 55 */
uint8 sleepEnable;

/* how many devices controlled by the module - offset 56 */
uint8 devNum;

/* enable flow control if the speed of RF and Uart mismatch - offset 57 */
uint8 flowCtrl;

/* sleep-wake rate - offset 58 */
uint16 swRate;

/* host init time (uint:ms) - offset 60 */
uint16 hostInitTime;

/* command no response timeout (uint:ms) - offset 62 */
uint16 noResponse;

/* PWM and other condifgure - offset 64 */
uint8 pwmMode;

/* uart configuration - offset 65 */
uint8 uartBaud;
uint8 uartGCR;
uint8 uartUCR;

/* flags0 - offset 68 */
uint8 Flags0;

/* uart sync word - offset 69 */
uint8 uartStart;
uint8 uartEnd;

/* bin or ACII mode - offset 71 */
uint8 uartMode;

/* host address length - offset 72 */
uint8 hostIdLen;
```

```
/* host address offest - offset 73 */
uint8 hostIdOffD;
uint8 hostIdOffU;

/* wildcard of address used for broadcast and multicast - offset 75 */
uint8 idWildcard0;
uint8 idWildcard1;

/* packet length offset - offset 77 */
uint8 pktLenOff;

/* packet head and tail size - offset 78 */
uint8 pktLenAdd;

/* if it is necessary to check pkt end - offset 79 */
uint8 pktEndCheck;

/* tick configure - offset 80 */
uint8 t3CC0;
uint8 t3CTL;

/* wait ticks before going to sleep - offset 82 */
uint16 sleepWait;

/* Rts timeout before giveup - offset 84 */
uint16 rtsTimeout;

/* req send times before giveup - offset 86 */
uint16 reqTimes;

/* repeat time sending sync packets - offset 88 */
uint16 sycRepeat;

/* ticks of beacon stage - offset 90 */
uint16 beaconTicks;

/* ticks of wake slot - offset 92 */
uint16 wakeSlotTicks;

/* Event for CC1110 PM2 - offset 94 */
uint16 pm2Events;

/* WORCTRL for CC1110 PM2 - offset 96 */
uint8 pm2WORCTRL;

/* extern time unit - offset 97 */
```

```
    uint8 externUnit;

    /* interval time for sync mode period - offset 98 */
    uint16 intervalSync;

    /* enable periods auto-send or not - offset 100 */
    uint8 repeatEn;

    /* length of the ACK pcket - offset 101 */
    uint8 ackLen;

    /* length of cmd to get host address information - offset 102 */
    uint8 getAddCmdLen;

    /* length of cmd to read data from the host - offset 103 */
    uint8 cmd0Len;
    uint8 cmd1Len;

    /* the ack packet of handshake - offset 105 */
    uint8 ackCmd[48];

    /* the command to get host address information - offset 153 */
    uint8 getAddCmd[48];

    /* the command to read data from the host - offset 201 */
    uint8 cmd0[48];
    uint8 cmd1[48];
} cfgBlock;
```

4. 控制块头部 55 字节读取

ATCI<CR>

该命令目前没有参数，仅有 5 个字节，该命令返回模块的控制块的前 55 字节的信息，用 16 进制 ASCII 字符串表示，包括模块名称、软件版本、MAC 地址、网络 ID、频点、信道号、输出功率等重要基本参数。返回的信息。模块返回数据帧按照“ATCI”作为开始，回车符<CR>作为结束字符，参数没字节数据用 2 位 ASCII 字符表示，字节之间用空格隔开。命令具体参数的解析，请参见下面的实例：

ATCI<CR>
ATCI 57 61 76 65 4d 65 73 68 20 41 4d 52 20 53 74 64 20 43 43 31 31 31 30 20 52 20 31 37 30 30 36 00 06 01 30 30 30 30 30 30 00 bb 11 b1 13 12 03 03 03 00 00 02 02 03 00<CR>

数据帧解析如下：

1. ‘ATCI’四个字符之后紧挨着前 96 个字节“ 57 61 76 65 4d 65 73 68 20 41 4d 52 20 53 74 64 20 43 43 31 31 31 30 20 52 20 31 37 30 30 36 00”是用 ASCII 十六进制字符表示的 32 个字节的模块控制块，该内容对应模块的名称字段“WaveMesh AMR Std CC1110 R 17006”。

2. 接下来的 6 个字节“ 06 01”为 ASCII 十六进制字符表示的 2 个字节的模块控制块，该内容对应模块的版本号：1.6。

3. 接下来的 18 个字节“ 30 30 30 30 30 30”为 ASCII 十六进制字符表示的 6 个字节的模块控制块，该内容对应模块的 MAC 地址：000000。

4. 接下来的 9 个字节“ 00 bb 11”为 ASCII 十六进制字符表示的 3 个字节的模块控制块，该内容对应模块的网络 ID，其中后 2 个字节需要查表进行配置，不能任意修改，请参考 PC 配置工具进行设置。

5. 接下来的 9 个字节“ b1 13 12”为 ASCII 十六进制字符表示的 3 个字节的模块控制块，该内容对应模块的 RF 的信道号为 0 的信道的使用的频率，实际物理意义请参见 PC 配置工具。

6. 接下来的 9 个字节“ 03 03 03”为 ASCII 十六进制字符表示的 3 个字节的模块控制块，该内容对应模块的 RF 的默认波特率、自适应波特率上限和自适应波特率的下限，实际物理意义请参见 PC 配置工具。

7. 接下来的 3 个字节“ 00”为 ASCII 十六进制字符表示的 1 个字节的模块控制块，该内容对应模块的载波检测绝对门限值(dB)，实际物理意义请参见 PC 配置工具。

8. 接下来的 12 个字节“ 00 02 02 03”为 ASCII 十六进制字符表示的 4 个字节的模块控制块，该内容对应模块的主信道号、起始辅助信道号、辅助信道间隔和辅助信道数量的掩码值，实际物理意义请参见 PC 配置工具。

9. 接下来的 3 个字节“ 00”为 ASCII 十六进制字符表示的 1 个字节的模块控制块，该内容对应模块的输出功率，实际物理意义请参见 PC 配置工具。

5．控制块的修改

“ATCC”+ 十六进制 ASCII 参数 + 回车<CR>

该命令以‘ATCC’四个字节作为帧的开始，参数为 16 进制的 ASCII 字符，参数的个数可以为 0 到多个，仅受到缓冲区 256 字节大小限制；最后以回车符<CR>即 0x0D 作为结束。值得注意的是，模块控制块的模块名称和版本字段即前 34 个字节是只读信息，不能被修改，该命令后的第一个参数对应模块控制块的“macID0”，以此顺序对应之后的参数。由于该命令是从控制块的固定偏移位置顺序修改参数，需要将不需要修改的参数值读取，再将其原封不动地写入。另外，对于该命令未涉及到的参数模块自动保持原来的参数值。

该命令的返回值为：
ATOK<CR> – 成功
ATER<CR> – 失败

举例说明：
修改 MAC 地址 macID0 和 macID1 两个字节为 0x30 0x31:
ATCC**3031**<CR>

修改 MAC 地址为 0x30 0x31 0x32 0x33 0x34 0x35:
ATCC**303132333435**<CR>

修改网络 ID 为 0x34 0xE2 0x3D，并且假设 MAC 地址为 0x30 0x31 0x32 0x33 0x34 0x35:
ATCC303132333435**34E23D**<CR>
注：该参数会同时修改网络 ID 和 MAC 地址，需要事先知道 MAC 地址，再写入才能保证 MAC 地址不变。

修改模块的输出功率参数值为 0x1：
ATCC 57 61 76 65 4d 65 73 68 20 41 4d 52 20 53 74 64 20 43 43 31 31 31 30 20 52 20 31 37 30 30 36 00 06 01 30 30 30 30 30 30 00 bb 11 b1 13 12 03 03 03 00 00 02 02 03 **01**<CR>
注：该参数会同时修改 macID0 字段之后、输出功率 paIndex 字段之前的所有参数，必须保证该命令后所有的参数都是修改之前的值。

注意：
- **该命令会擦除无线模块片内控制块所在 FLASH 空间，FLASH 擦写次数为 1000 次左右，尽可能的避免频繁擦写 FLASH 空间，以免造成无线模块硬件报废。**
- **该命令在执行期间，应保证无线模块的供电可靠稳定，避免 FLASH 空间在数据擦除后数据写入失败。**
- **该命令对所设置的参数没有做任何正确性检查和校验，在该命令执行后建议采用 ATCI 命令重新读取模块的控制块，以确保写入的参数正确无误。**
- **在该命令执行完毕后，模块会自动重启，新的参数将会使能，请谨慎修改诸如 UART 波特率之类的参数，如果修改失败，就会导致模块串口无法正常工作，而且在正常工作模式下不能修复。**