SL-13.56MOD SFID 专用射频模块使用说明

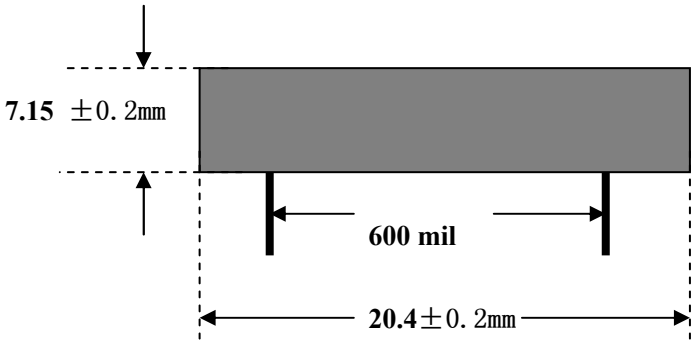
一. 概述

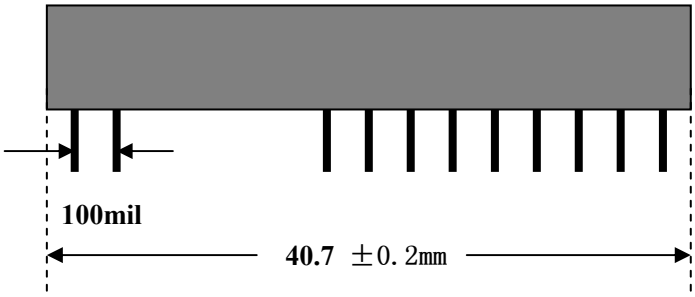
该模块可用于读取二代身份证和 M1 卡的 ID 号。模块与上位机之间可以使用多种接口连接（UART、SPI、TWI），方便用户使用。

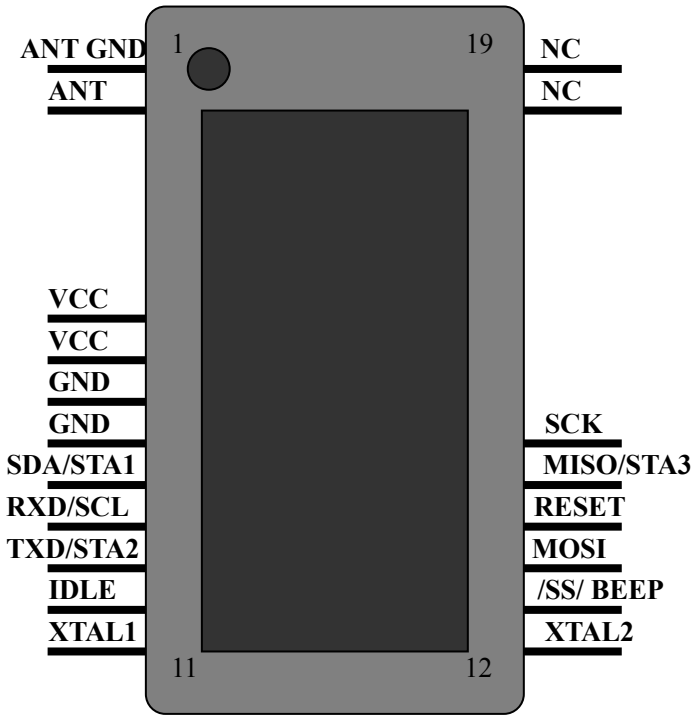
二. 特点

- 读取二代身份证 ID，MIFARE 1 卡的 ID 号，以及其他厂家的 13.56MHz 射频卡片。
- 提供多种外部通讯接口，方便用户开发.
  - UART：通用异步通讯接口。外接 RS232 芯片后可以方便的和 PC 机交换数据。固定波特率：115200Bps
  - SPI： 标准 SPI 硬件接口。最高速率可以达到 2.0MBps。使用标准 SPI 接口的模式 0 进行通讯。
  - TWI： 通用两线协议，可以使用最少的 I/O 数量建立模块与上位机的连接。最高可以达到 400 KPBS。
- I/O 管脚 CMOS 和 TTL 输入/输出电平兼容
- 内置上电复位电路。
- 自带看门狗电路。
- 尺寸：40.7mm*20.4mm*7.15mm。
- 工作频率：13.56MHz。
- 读写距离：读写距离和搭配天线尺寸有关。请参考天线设计的相关文档。
- 内置 256 字节 EEPROM。
- 工作电压：DC5V。、
- 直接驱动 **50 Ohm** 天线
- **卡片通讯最大 FIFO= 48 BYTE。**

三. 外观与管脚定义







序号	名称	描述
PIN1	**ANT GND**	天线地
PIN2	**ANT**	天线驱动（**50 Ohm antenna**）
PIN3	**VCC**	模块电源 **+5V**
PIN4	**VCC**	模块电源 **+5 V**
PIN5	**GND**	电源地
PIN6	**GND**	电源地
PIN7	**SDA/STA1**	**TWI** 接口数据 **I/O/UART** 或 **SPI** 状态指示
PIN8	**RXD/SCL**	**UART** 接口的 **RXD/TWI** 接口的时钟信号
PIN9	**TXD/STA2**	**UART** 接口的 **TXD/SPI** 或 **TWI** 接口状态指示
PIN10	**IDLE**	休眠模式控制 **0**：工作 **1**： 休眠（休眠以后不接收任何命令）
PIN11	**XTAL1**	外接 **13.56MHz 20Pf** 负载的晶体（输入）
PIN12	**XTAL2**	外接 **13.56MHz 20Pf** 负载的晶体
PIN13	**/SS**	**SPI** 接口片选信号，输入口，低有效 **(BEEP)**

PIN14	**MOSI**	**SPI 接口数据输入**
PIN15	**RESET**	**模块内部 MCU 复位控制，低有效**
PIN16	**MISO/STA3**	**SPI 接口数据输出/UART 或 TWI 模式为模块状态指示**
PIN17	**SCK**	**SPI 接口时钟信号**
PIN18	**NC**	保留未用
PIN19	**NC**	保留未用

复用管脚的说明：

- **PIN16（MISO/STA3）：**
  SPI 接口下为数据输出脚。
  UART 接口下，模块发送数据时 PIN16=0。模块空闲时 PIN16=1。
  TWI 接口下，模块执行命令过程中 PIN16=0。模块空闲时 PIN16=1。

- **PIN7（SDA/STA1）：**
  SPI 接口下，模块执行命令过程中 PIN7=0。模块空闲时 PIN7=1。
  UART 接口下，模块执行命令过程中 PIN7=0。模块空闲时 PIN7=1。
  TWI 接口下，数据输入/输出脚。

- **PIN8（RXD/SCL）：**
  SPI 接口下，没有定义功能。
  UART 接口下，数据接收端。
  TWI 接口下，时钟输入端。

- **PIN19（TXD/STA2）：**
  SPI 接口下，PIN19=0 表示模块准备好数据，等待 HOST 调取。发送完最后一个字节数据后复位为高电平。
  UART 接口下，数据发送端。
  TWI 接口下，PIN19=0 表示模块准备好数据，等待 HOST 调取。发送完最后一个字节数据后复位为高电平。
- **PIN13（SS/BEEP）：**
  SPI 接口下，设备选择端，输入状态，低有效。
  UART 和 TWI 接口：BEEP 信号的输出，可以直接驱动 5V 蜂鸣器。
- **PIN11/PIN12（XTAL1/XTAL2）：**
  可以直接连接 13.56MHz 的晶体振荡器。模块内置 20PF 电容器和起振电路，无需其他外围电路。

四. 通讯协议

该部分只介绍模块与上位机通讯时的数据祯结构，命令应答方式。具体的通讯的物理层协议本部分不做过多的描述，请参考相关的资料和文档。

模块和上位机之间采用命令应答的通讯模式。平时模块处在空闲状态，接收到上位机的命令后，进入命令执行状态（各种接口下的状态指示，请参考管脚定义部分）。模块命令执行完毕，准备好执行结果后会通过状态管脚给上位机指示（各种接口下的数据有效指示，请参考管脚定义部分）。上位机按照指示取走命令结果后，模块再次进入空

闲状态。

上位机可以通过 PIN18（IDEL）管脚，控制模块的休眠和唤醒。在休眠状态（**PIN18=1**）下，射频场关闭，MCU 休眠。此时模块功耗最小。要唤醒模块（**PIN18=0**）需要 >5 毫秒的延时。唤醒模块后，需要附加的开场命令打开射频场。模块复位默认射频场状态为“关闭”。

1. 上位机发送命令给模块

数据结构：

数据头 + 长度 + 长度校验 + 命令代码 + 等待延时 +[命令参数]+ 校验和

HEAD+LENGTH+ LEN_CHK **+COMMAND+WAIT TIME+PARAMETER+CHECKSUM**

- **HEAD(数据头)：** 一个字节的“**0xA6**”。如果发送多个数据头字节，模块自动丢弃。
- **LENGTH(长度)：** 整个命令数据祯中除了 **HEAD(数据头)**、**LENGTH(长度)**、**LEN_CRC(长度校验)**以外的所有数据的字节数。

  对于没有 **PARAMETER(命令参数)** 的命令,LENGTH=3。

  对于有 **PARAMETER(命令参数)** 的命令 LENGTH=命令参数的长度 +3。该参数一个字节。
- **LEN_CHK(长度校验)：** **LEN_CHK** 是 LENGTH 取反以后的值。该参数一个字节。
- **COMMAND(命令代码)：** 需要模块执行的命令代码。详细的命令说明在第五部份介绍。该参数一个字节。
- **WAIT TIME(等待延时)：** 该命令预计执行时间。在卡片操作命令里面这个时间是模块将读写卡命令发送完毕后等待卡片应答的时间，对于不操作卡片的命令如关场、握手等该参数可以设置为“1”； 对于开场和检测卡的命令为开场的延时时间单位是毫秒；对于读写卡片的命令该参数的设置需要参考不同型号卡片的文档，根据实际需要设置。实际的等待时间是该参数的值乘以一个最小间隔单位。最小时间间隔约为“193 毫秒”。**注意该参数若设置为“0”则等待 TR0 的时间为 4969mS**。该参数一个字节。

  COMMAND=0 WAIT TIME 没有意义。
  COMMAND=1 WAIT TIME 单位是 19.3mS
  COMMAND=2 WAIT TIME 没有意义。
  COMMAND=3 WAIT TIME 单位是 1 mS
  COMMAND=4 WAIT TIME 没有意义。
  COMMAND=5 WAIT TIME 单位是 1 mS
  COMMAND=6 WAIT TIME 没有意义。
  COMMAND=7 WAIT TIME 没有意义。
  COMMAND=8 WAIT TIME 单位是 19.3Ms

COMMAND=9　WAIT TIME 存储器地址。
COMMAND=10 WAIT TIME 存储器地址。

- **PARAMETER(命令参数)：** 只有 SEND_RECEIVE 命令需要参数。该参数就是一个完整的 14443 命令或卡片定义的其他命令。模块在执行 SEND_RECEIVE 命令的时候，会自动计算该部分的 CRC，并连同 SOF、本数据、CRC、EOF（完整桢）发送给卡片然后等待卡片的应答，等待的超时界限就是参数“等待延时”指定的。（请参考 ISO14443 标准相关文档中关于 TR0 的定义）。
- **CHECKSUM(校验和)：** 命令校验和是从 **COMMAND**(命令代码)开始的所有字节的**带进位位**的累加和的**反码**。初始值为 0，进位位为 0 。

2. 上位机接收命令执行结果

数据结构：

数据头 + 长度 + 长度校验 + 命令代码 + 执行状态 + [数据] + 校验和

HEAD+LENGTH+ LEN_CHK +COMMAND+STATUS+[DATA]+ CHECKSUM

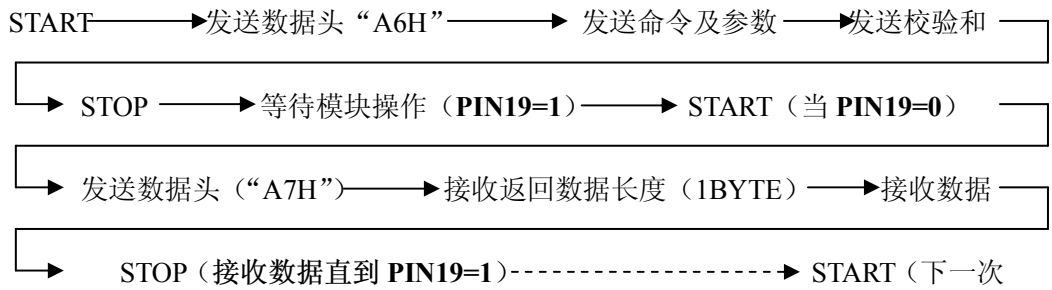
- **HEAD(数据头)：** 数据头固定为 0xA6。
- **LENGTH(长度)：** 整个数据祯中除了数据头、长度、长度校验以外的所有数据的字节数（包括数据校验和）。该参数一个字节。
- **LEN_CHK (长度校验)：** 数据长度字节取反。该参数一个字节。
- **COMMAND(命令代码)：** 模块返回数据所对应的命令的代码。该值应该和最近一次发送给模块的命令祯中的“命令代码”参数一致。详细的命令说明在第五部份介绍。该参数一个字节。
- **STATUS(执行状态)：** 该命令执行的结果。如果命令执行成果则状态为“0”。这个命令状态只标志模块的执行命令和通讯的状态，并不标志卡片的实际状态，卡片操作是否成功要在卡片的返回数据中表示。其他的错误代码在第五部分介绍。该参数一个字节。
- **DATA(数据)：** 只有 SEND RECEIVE 以及 GET VERSION 命令返回数据。对于 SEND RECEIVE 命令该部分的数据就是卡片的应答数据，不包括 14443 协议中的 CRC。对于 GET VERSION 命令该部分的数据就是模块的版本信息。
- **CHECKSUM (校验和)：** 返回数据校验和是从“命令代码”开始的所有字节的**带进位位的累积和**的反码。初始值为 0，进位位为 0 。

3. 数据校验方式

接收和发送两个过程中的数据校验方式都是采用带进位的字节累加和取反的方法计算。参加校验和运算的数据是数据祯中除了“HEAD”、“LENGTH”、“LEN_CRC”以外的所有数据。校验和的初始值为“0”。校验和的计算方法代码在附录中有说明。

4. TWI 接口的特殊说明

对于 TWI 接口上位机发送命令给模块的协议和 SPI 以及 UART 是一致的。但是由于模块被定义为 TWI 的从机，所以在命令执行结束不能主动发送数据给上位机。参考 I2C 协议的要求在上位机读取命令执行结果前需要先发送一个字节的命令头给模块，在此命令头发送完毕（接收到来自模块的 ACK 信号），开始读取数据。完整的通讯过程可以描述为：



TWI 的数据物理层协议是参考 I2C 协议定义的。可以参考相关的文档资料（硬件时序可以参考 **AT24C01 的技术文档**）。每个字节 9 个时钟。数据在 SCL 的上升沿有效。接收方每接收一个字节（8 BIT）的数据，需要在第九个时钟给发送方一个 ACK 应答。ACK=0 表示接收正确。上位机发送命令给模块，每个字节结束模块给上位机一个 ACK。上位机读取模块的执行结果时，每读取一个字节就给模块一个 ACK，读取最后一个字节时，上位机需要给模块一个反向的 ACK（NACK，逻辑“1”），然后以 STOP 结束通讯。**在用 TWI 接口时，模块返回的数据中没有数据头：0xA6**

五． 命令与应答

下列的命令和返回值是以串行通讯协议为样板给的例子。在 **TWI** 模式下，模块返回值里面没有前面的 **HEAD** 部分。

1．命令列表

- **0x10：握手命令（HAND SHAKE）**

  该命令实现上位机和模块的握手操作。没有任何实际的功能，只是用来测试模块和上位机的通讯是否正常。命令没有参数，调用该命令在命令祯中的“等待延时”可以设置为“1-5”。下面的 16 进制代码为完整的关场命令的数据祯内容：

  A6 03 FC 10 02 75

  HEAD=A6

  LENGTH=03

  LEN_CHK =FC

  COMMAND=10

  WAITTIME=02

  CHECKSUM =75

  模块返回的数据为：

  A6 03 FC 10 00 77

  HEAD=A6（如果是 UART 模式下，模块返回 3 个字节的数据头）。

  LENGTH=03

  LEN_CHK =FC

  COMMAND=10

  STATUS=00

  CHECKSUM =77

- **0x11：发送接收（SEND RECEIVE）**

  该命令实现上位机和卡片的数据交换操作，上位机发送给卡片的命令在参数 PARAMETER(命令参数)中，其中不包括 IS014443 规定的 CRC 校验字节。模块在发送这些信息前会自动计算 CRC。调用该命令的时候要特别注意 WAIT TIME(等待延时)的设置。该参数为一个字节。模块发送命令以后等待卡片的应答的超时界限就是这个参数乘以“193”毫秒。应当严格按照不同型号卡片的技术文档的要求，设置这个参数。下面是 ISO14443 TYPE B –3 中的 REQB 命令的实际数据祯内容供参考：

  A6 06 F9 11 05 05 00 00 F7

  HEAD=A6

  LENGTH=06

  LEN_CHK =F9

  COMMAND=11

  WAIT TIME=05

  PARAMETER=”050000”（标准卡片指令，参考卡片资料和 14443 标准）

  CHECKSUM =F7

  模块返回信息：

  A6 0F F0 11 00 50 00 00 00 01 FF FF FF 64 00 30 51 3F

  HEAD=A6

LENGTH=0F

LEN_CHK =F0

COMMAND=11

STATUS=00

DATA=" 50 00 00 00 01 FF FF FF 64 00 30 51"（具体的数据意义请参考卡片和 ISO14443 标准文档）。

CHECKSUM =3F

- **0x12：关场命令（CLOSE FIELD）**

该命令使模块关闭射频场。该命令和开场命令配合可以实现给卡片硬复位的功能，另外关场以后模块的功耗会大大降低。命令没有参数，调用该命令在命令祯中的"等待延时"可以设置为"1-5"。下面的 16 进制代码为完整的关场命令的数据祯内容：

A6 03 FC 12 05 70

HEAD=A6

LENGTH=03

LEN_CHK =FC

COMMAND=12

WAIT TIME=05

CHECKSUM =70

模块返回的数据为：

A6 03 FC 12 00 75

HEAD=A6

LENGTH=03

LEN_CHK =FC

COMMAND=12

STATUS=00

CHECKSUM =75

- **0x13：开场命令（OPEN FIELD）**

该命令使模块打开射频场，准备读写卡片。命令没有参数，调用该命令在命令祯中的"等待延时"可以设置为"1-5"。下面的 16 进制代码为完整的开场命令的数据祯内容：

A6 03 FC 13 05 6F

HEAD=A6

LENGTH=03

LEN_CHK =FC

COMMAND=13

WAIT TIME=05

CHECKSUM =6F

模块返回的数据为：

A6 03 FC 13 00 74

HEAD=A6

LENGTH=03

LEN_CHK =FC

COMMAND=13

STATUS=00

CHECKSUM =74

- **0x14：选择 ISO14443 协议（SELECT PROTOCL）**

该命令选择模块与卡片之间通讯需要符合的标准类型。ISO14443 目前有两个主要的分支，一个是 TYPE A 标准（MIFARE 1 以及兼容卡片符合的标准）。另一个是 TYPE B 标准（ATMEL,ST 等公司的卡片遵循的标准，主要的卡片有 AT88RF020,AT88SC0104CRF 系列，ST_SRI176、ST_SRIX4K 等）。使用该命令设置卡片的协议类型，针对某一种类型的卡片的操作，只需要在操作前设置一下该命令就可以了。如果要同时操作多种协议的卡片，则在更换协议的时候，需要调用这个命令。调用该命令的时候 WAIT TIME 参数作为一个命令参数使用。WAIT TIME=0A（进入 TYPE A 模式），WAIT TIME=0B（进入 TYPE B 模式）。下面的 16 进制代码为完整的选择 TYPE A 协议命令的数据祯内容：

A6 04 FB 14 0A 0A 5F

HEAD=A6

LENGTH=04

LEN_CHK =FB

COMMAND=14

WAIT TIME=0A

PARAMETER=”0A”

CHECKSUM =5F

模块返回的数据为：

A6 03 FC 14 00 73

HEAD=A6

LENGTH=03

LEN_CHK =FC

COMMAND=14

STATUS=00

CHECKSUM =73

- **0x15：蜂鸣器操作命令（BEEP OPTION）**

该命令使模块的 BEEP 管脚（PIN13）发送一个制定宽度的脉冲，可以驱动外部的蜂鸣器。调用该命令在命令祯中的 WAIT TIME(等待延时)参数表示 PIN13 置低的时间。可以设置为“0-255”。下面的 16 进制代码为完整命令的数据祯内容：

A6 03 FC 15 FF 72

HEAD=A6

LENGTH=03

LEN_CHK =FC

COMMAND=15

WAIT TIME=FF

CHECKSUM =72

模块返回的数据为：

A6 03 FC 15 00 72

HEAD=A6

LENGTH=03

LEN_CHK =FC

COMMAND=15

STATUS=00

CHECKSUM =72

- **0x16：获取版本信息（GET VERSION）**

该命令返回当前使用模块的版本信息。命令没有参数，调用该命令在命令祯中的“等待延时”可以设置为“1-5”。下面的 16 进制代码为完整的命令祯内容：

A6 03 FC 16 05 6C

HEAD=A6

LENGTH=03

LEN_CHK =FC

COMMAND=16

WAIT TIME=05

CHECKSUM =6C

模块返回的数据为：

A6 2F D0 16 00 53 74 61 6E 74……. 00 66 A4

HEAD=A6

LENGTH=24

LEN_CHK =D0

COMMAND=16

STATUS=00

DATA=” 53 74 61 6E 74……. 31 36”

CHECKSUM =A4

完整的版本信息是一串 ASCII 码字符串

“Standard ISO14443 RF Reader,V1.02,2006,04,16”

- **0x17: 读取 MIFARE 1 卡 ID 号（READ MIFARE 1）**

该命令用来读取 MIFARE 1 卡片的 ID 号。读写器发现卡片就锁定卡片，并返回卡片的 ID 信息。调用该命令的时候要特别注意 WAIT TIME(等待延时)的设置：该参数为卡片从上电复位到可以接受命令的时间，单位为毫秒。具体采用多长的延时时间，可以参考卡片的技术文档。一般设置为 05 对大多数片就足够了。

A6 03 FC 17 05 6B

HEAD=A6

LENGTH=03

LEN_CHK =FC

COMMAND=17

WAIT TIME=05

CHECKSUM =6B

返回信息为：

A6 0C F3 17 00 0A 04 00 A6 A2 FA 69 97 08 15

HEAD=A6

LENGTH=0C

LEN_CHK =F3

COMMAND=17
STATUS=00
DATA=” 0A 04 00 A6 A2 FA 69 97 08 ”
TYPE A ID BCC
CHECKSUM =15

- **0x18：读取第二代身份证卡 ID 号（READ PERSONAL CARD）**

该命令只用来读取第二代居民身份证卡的 ID 号。读写器发现卡片就锁定卡片，并返回卡片的 ID 信息。调用该命令的时候要特别注意 WAIT TIME(等待延时)的设置：该参数为卡片从上电复位到可以接受命令的时间，单位为毫秒。具体采用多长的延时时间，可以参考卡片的技术文档。一般设置为 05 对大多数片就足够了。

A6 06 F9 18 05 05 00 00 65
HEAD=A6
LENGTH=06
LEN_CHK =F9
COMMAND=18
WAIT TIME=05
DATA=“05 00 00”（该字段固定为 “05 00 00” ）
CHECKSUM =65
返回信息为：
A6 37 C8 18 00 50 00 00 00 00 D1 03 86 05 00 80 80 20 B3 20 2C A4 24 68 78 00 01 42 4A 01 06 03 14 00 00 00 00 00 06 B2 10 70 74 4B EA 8B 6F 47 AA 34 86 18 B9 93 16 21 A5 7A

HEAD=A6
LENGTH=37
LEN_CHK =C8
COMMAND=18
STATUS=00
DATA=” 50 00 00 00 00 ** 03 ** 05 00 ** ** 20 ** 20 ** ** 24 ** 78 00
身份证卡 ID 号
01 ** ** 01 06 03 14 00 00 00 00 00 06 ** 10 70 ** 4B ** 8B ** ** AA **

86 18 ** 93 ** ** ** ”
（* 号部分为隐去的身份信息。返回的数据信息里面从第 **13** 个字节开始的 **8** 个字节的数据是身份证卡片的 **ID** 号）。
CHECKSUM =7A

- **0x19：写 EEPROM（WRITE EEPROM）**

  该命令向模块中的 EEPROM 写入数据。注意：模块中的 EEPROM 的容量为：256 字节。在实际使用的时候，最前面的一个字节不要使用。另外该命令的延时时间按照下面的单位进行计算。每个字节 4mS。命令中的 WAIT TIME 参数在这里作为写入字节的地址使用。下面的 16 进制代码为完整的写 EEPROM 命令的数据祯内容：

  A6 0A F5 19 00 12 34 56 78 00 00 00 59

  HEAD=A6

  LENGTH=0A

  LEN_CHK =F5

  COMMAND=19

  WAIT TIME=00（从 00H 开始写入数据）

  PARAMETER=” 12 34 56 78 00 00 00”

  CHECKSUM =59

  模块返回的数据为：

  A6 03 FC 19 00 6E

  HEAD=A6

  LENGTH=03

  LEN_CHK =FC

  COMMAND=19

  STATUS=00

  CHECKSUM =6E

- **0x1A：读 EEPROM（READ EEPROM）**

  该命令从模块的 EEPROM 中读取数据。注意：模块中的 EEPROM 的容量为：256 字节。在实际使用的时候，最前面的一个字节不要使用。另外该命令的延时时间按照下面的单位进行计算。每个字节 1 mS。命令中的 WAIT TIME 参数在这里作为字节的地址使用。下面的 16 进制代码为完整的读 EEPROM 命令的数据祯内容：

  A6 04 FD 1A 00 04 69

  HEAD=A6

  LENGTH=04

  LEN_CHK =FD

  COMMAND=1A

  WAIT TIME=00（从 00H 开始写入数据）

  PARAMETER=” 04”（读取 4 个字节的数据）

  CHECKSUM =69

  模块返回的数据为：

  A6 07 F8 1A 00 12 34 56 78 58

  HEAD=A6

  LENGTH=07

  LEN_CHK =F8

  COMMAND=1A

  STATUS=00

  DATA=” 12 34 56 78”

CHECKSUM =6E

2．返回状态码

模块返回的数据祯中参数 STATUS(执行状态)表示模块执行命令的状态。所有的命令代码如下：

- 00H　　操作成功
- 01H-77H　监测 TYPE A 卡片的时候出现冲突。
- 78H　　接收卡片数据，CRC 校验错。
- 80 H　　读写卡片时，没有开场
- 81H　　卡片无应答
- 82H　　卡片返回数据错误
- 83H　　调制常开
- 84H　　通讯命令校验和错误
- 85H　　未知错误
- 86H　　命令长度错误
- 87H　　EEPROM 操作错误

附录

1．参考天线设计

可参考 PHILIPS 关于天线设计的相关文档。

2．UART 通讯流程描述

int IssuCommand_For_020(unsigned char Length,unsigned char Command_Code,unsigned char Command_Para,unsigned char Wait_Time,int *RLength,char *Data,int TimeOut)

{

// Length：要发送给卡片的数据长度（命令参数的长度）

// Command_Code：命令代码（0-5）

// Wait_Time：等待延时

// Command_Para：没有使用

// Rlength：返回数据的长度

// Data：发送给卡片的数据（命令参数）

// TimeOut：上位机等待模块应答的超时界限

// Check_Sum（Data,Length,&CheckSum)计算校验和,计算的结果在 CheckSum 中。

// Send(Length,(char*)Comm)发送指定长度的数据

// Receive(Length,(char*)Receive_Data)接收指定长度的数据，内部的延时为 20 毫秒

```
unsigned char Comm[64];
unsigned char CheckSum=0;
int Cur_Len;
int status,i=0;
int Wait_Timeout=0;
unsigned char Temp_Byte=0;
unsigned char Receive_Data[64];
PurgeComm(hCom,PURGE_RXABORT | PURGE_TXABORT | PURGE_TXCLEAR |
```

```
PURGE_RXCLEAR);                 //清除串口缓冲区数据
    Cur_Len=Length+3;
    Comm[0]=0xA6;
    Comm[1]=0xA6;
    Comm[2]=0xA6;               //数据头
    Comm[3]=Cur_Len;            //长度
    Comm[4]=Comm[3]^0xff;       //长度校验
    Comm[5]=Command_Code;       //命令代码
    Comm[6]=Wait_Time;          //等待延时
    if(Length>0)
    {
        for(int i=0;i<Length;i++)
        {
            Comm[7+i]=(char)*(Data+i);          //命令参数
        }
    }
    Check_Sum(Comm+5,2+Length,&CheckSum);   //计算校验和
    Comm[7+Length]=CheckSum;                //校验和
    status=Send(8+Length,(char*)Comm);      //发送命令
    if (status!=OPOK)
    {
        return(status);
    }
    if(TimeOut<20)
        TimeOut=20;
    Wait_Timeout=(TimeOut/20) +1;
    do  //循环等待数据头直到收到不是数据头的数据或超时。
    {
        status=Receive(1,(char*)Receive_Data);
        if(status!=0)
        {
            Wait_Timeout--;
        }
        else
        {
            Temp_Byte=Receive_Data[0];
            Wait_Timeout=(TimeOut/20)+1;
        }
    }while(Wait_Timeout!=0 && ((status!=0)||(Temp_Byte==0xa6)));
    if (status!=0)
        return(WAIT_RESPOND_TIMEOUT);
    status=Receive(1,(char*)Receive_Data);          //长度校验
    if (status!=0)
        return(status);
```

```
    CheckSum=Temp_Byte^Receive_Data[0];

    Receive_Data[0]^=0xff;
    if(Receive_Data[0]!=Temp_Byte)
        return(CLIENT_SEND_WRONG_RESULT);
    else
        Cur_Len=Receive_Data[0];
    status=Receive(Cur_Len,(char*)Receive_Data);          //接收数据
    if (status!=0)
        return(status);
    if (Receive_Data[0]!=Command_Code)
        return(RECEIVE_DATA_ERROR);
    Check_Sum(Receive_Data,Cur_Len-1,&CheckSum);          //数据校验

    if(CheckSum!=Receive_Data[Cur_Len-1])
    {
        return(RECEIVE_DATA_ERROR);
    }
    *RLength=Cur_Len-1;
    for(i=0;i<Cur_Len-1;i++)
        *(Data+i)=Receive_Data[i];                //保存数据
    return (0);
}
```

3．通讯命令校验和计算方法

```
void Check_Sum(unsigned char *Data,short int Length,unsigned char *Checksum)
//DATA:原始数据
//LENGTH:计算校验和的数据长度
//CHECKSUM:计算结果
{
    int temp=120;
    char *Pb;
    for(int i=0;i<Length;i++)
    {
        temp+=*(Data+i);
        if((temp==255)||(temp==510))
            temp=255;
        else
            temp%=255;
    }
    Pb=(char*)&temp;
    *Checksum=(*Pb)^0xff;
}
```