

S50 非接触式 IC 卡性能简介（M1）

一、 主要指标

- 容量为 8K 位 EEPROM
- 分为 16 个扇区，每个扇区为 4 块，每块 16 个字节,以块为存取单位
- 每个扇区有独立的一组密码及访问控制
- 每张卡有唯一序列号，为 32 位
- 具有防冲突机制，支持多卡操作
- 无电源，自带天线，内含加密控制逻辑和通讯逻辑电路
- 数据保存期为 10 年，可改写 10 万次，读无限次
- 工作温度：-20℃~50℃(湿度为 90%)
- 工作频率：13.56MHZ
- 通信速率：106 KBPS
- 读写距离：10 cm 以内（与读写器有关）

二、 存储结构

1、M1 卡分为 16 个扇区，每个扇区由 4 块（块 0、块 1、块 2、块 3）组成，（我们也将 16 个扇区的 64 个块按绝对地址编号为 0~63，存贮结构如下图所示：

扇区 0	块 0		数据块	0
	块 1		数据块	1
	块 2		数据块	2
	块 3	密码 A 存取控制 密码 B	控制块	3
扇区 1	块 0		数据块	4
	块 1		数据块	5
	块 2		数据块	6
	块 3	密码 A 存取控制 密码 B	控制块	7
		⋮		
		⋮		
		⋮		
扇区 15	0		数据块	60
	1		数据块	61
	2		数据块	62
	3	密码 A 存取控制 密码 B	控制块	63

- 2、第 0 扇区的块 0（即绝对地址 0 块），它用于存放厂商代码，已经固化，不可更改。
- 3、每个扇区的块 0、块 1、块 2 为**数据块**，可用于存贮数据。

数据块可作两种应用：

- ★ 用作一般的数据保存，可以进行**读**、**写**操作。

★ 用作数据值，可以进行**初始化值、加值、减值、读值**操作。

4、每个扇区的块 3 为**控制块**，包括了密码 A、存取控制、密码 B。具体结构如下：

A0 A1 A2 A3 A4 A5	FF 07 80 69	B0 B1 B2 B3 B4 B5
-------------------	-------------	-------------------

密码 A（6 字节） 存取控制（4 字节） 密码 B（6 字节）

5、每个扇区的密码和存取控制都是独立的，可以根据实际需要设定各自的密码及存取控制。存取控制为 4 个字节，共 32 位，扇区中的每个块（包括数据块和控制块）的存取条件是由密码和存取控制共同决定的，在**存取控制**中每个块都有相应的**三个控制位**，定义如下：

块 0: C10 C20 C30
 块 1: C11 C21 C31
 块 2: C12 C22 C32
 块 3: C13 C23 C33

三个控制位以正和反两种形式存在于存取控制字节中，决定了该块的访问权限（如进行减值操作必须验证 KEY A，进行加值操作必须验证 KEY B，等等）。三个控制位在存取控制字节中的位置，以块 0 为例：

对块 0 的控制：

	bit 7	6	5	4	3	2	1	0
字节 6				C20_b				C10_b
字节 7				C10				C30_b
字节 8				C30				C20
字节 9								

（注： C10_b 表示 C10 取反）

存取控制（4 字节，其中字节 9 为备用字节）结构如下所示：

	bit 7	6	5	4	3	2	1	0
字节 6	C23_b	C22_b	C21_b	C20_b	C13_b	C12_b	C11_b	C10_b
字节 7	C13	C12	C11	C10	C33_b	C32_b	C31_b	C30_b
字节 8	C33	C32	C31	C30	C23	C22	C21	C20
字节 9								

（注： _b 表示取反）

6、**数据块**（块 0、块 1、块 2）的存取控制如下：


控制位 (X=0..2)			访问条件 (对数据块 0、1、2)			
C1X	C2X	C3X	Read	Write	Increment	Decrement, transfer, Restore
0	0	0	KeyA B	KeyA B	KeyA B	KeyA B
0	1	0	KeyA B	Never	Never	Never
1	0	0	KeyA B	KeyB	Never	Never
1	1	0	KeyA B	KeyB	KeyB	KeyA B
0	0	1	KeyA B	Never	Never	KeyA B
0	1	1	KeyB	KeyB	Never	Never
1	0	1	KeyB	Never	Never	Never
1	1	1	Never	Never	Never	Never

(KeyA|B 表示密码 A 或密码 B, Never 表示任何条件下不能实现)

例如：当块 0 的存取控制位 C10 C20 C30=1 0 0 时，验证密码 A 或密码 B 正确后可读；验证密码 B 正确后可写；不能进行加值、减值得操作。

7、**控制块**块 3 的存取控制与**数据块**（块 0、1、2）不同，它的存取控制如下：

			密码 A		存取控制		密码 B	
C13	C23	C33	Read	Write	Read	Write	Read	Write
0	0	0	Never	KeyA B	KeyA B	Never	KeyA B	KeyA B
0	1	0	Never	Never	KeyA B	Never	KeyA B	Never
1	0	0	Never	KeyB	KeyA B	Never	Never	KeyB
1	1	0	Never	Never	KeyA B	Never	Never	Never
0	0	1	Never	KeyA B	KeyA B	KeyA B	KeyA B	KeyA B
0	1	1	Never	KeyB	KeyA B	KeyB	Never	KeyB
1	0	1	Never	Never	KeyA B	KeyB	Never	Never
1	1	1	Never	Never	KeyA B	Never	Never	Never

例如：当块 3 的存取控制位 C13 C23 C33=1 0 0 时，表示  。

密码 A：不可读，验证 KEYA 或 KEYB 正确后，可写（更改）。

存取控制：验证 KEYA 或 KEYB 正确后，可读、可写。

密码 B：验证 KEYA 或 KEYB 正确后，可读、可写。

三、 工作原理

卡片的电气部分只由一个天线和 ASIC 组成。

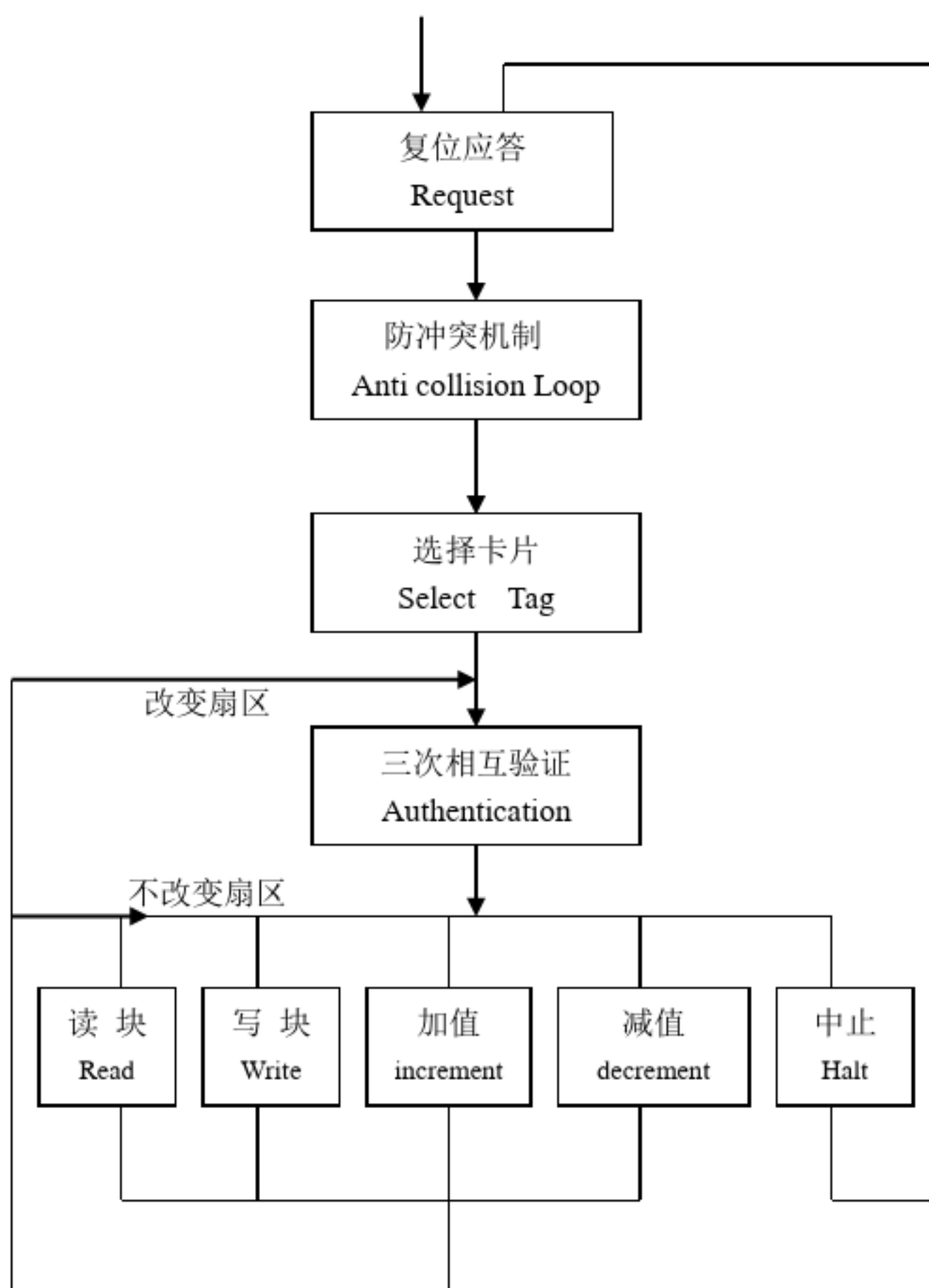
天线：卡片的天线是只有几组绕线的线圈，很适于封装到 ISO 卡片中。

ASIC：卡片的 ASIC 由一个高速（106KB 波特率）的 RF 接口，一个控制单元和一个 8K 位 EEPROM 组成。

工作原理：读写器向 M1 卡发一组固定频率的电磁波，卡片内有一个 LC 串联谐振电路，其频率与读写器发射的频率相同，在电磁波的激励下，LC 谐振电路产生共振，从而使

电容内有了电荷，在这个电容的另一端，接有一个单向导通的电子泵，将电容内的电荷送到另一个电容内储存，当所积累的电荷达到 2V 时，此电容可做为电源为其它电路提供工作电压，将卡内数据发射出去或接取读写器的数据。

四、 M1 射频卡与读写器的通讯



复位应答 (Answer to request)

M1 射频卡的通讯协议和通讯波特率是定义好的，当有卡片进入读写器的操作范围时，

读写器以特定的协议与它通讯，从而确定该卡是否为 M1 射频卡，即验证卡片的卡型。

防冲突机制 (Anticollision Loop)

当有多张卡进入读写器操作范围时，防冲突机制会从其中选择一张进行操作，未选中的则处于空闲模式等待下一次选卡，该过程会返回被选卡的序列号。

选择卡片(Select Tag)

选择被选中的卡的序列号，并同时返回卡的容量代码。

三次互相确认(3 Pass Authentication)

选定要处理的卡片之后，读写器就确定要访问的扇区号，并对该扇区密码进行密码校验，在三次相互认证之后就可以通过加密流进行通讯。（在选择另一扇区时，则必须进行另一扇区密码校验。）

对数据块的操作

读 (Read): 读一个块；

写 (Write): 写一个块；

加 (Increment): 对数值块进行加值；

减 (Decrement): 对数值块进行减值；

存储 (Restore): 将块中的内容存到数据寄存器中；

传输 (Transfer): 将数据寄存器中的内容写入块中；

中止 (Halt): 将卡置于暂停工作状态；

关于非接触式 IC 智能(射频)卡 及其读写设备内核技术的研究与应用开发

上海华东磁记录电子公司 张敏

前 言

当今的信息技术又到了一个关键的十字路口。单独的 PC 甚至局域网的时代已为“全球化网络”时代所代替。互联网 INTERNET 的产生和今天的飞速发展和广泛渗透就是典范。

IC（集成电路）智能卡正在并已经融入当今信息技术的主流。她以其高度的信息集成，高度的安全性，正日渐其辉煌和灿烂。

随着金融行业的不断发展，社会经济的日新月异，特别是公共交通行业，无线通信领域，卫生保健行业，封闭式场所管理，身份识别，电话通信，大楼保安系统等等，人们已愈来愈多地开始接受和使用 IC 智能卡。特别是银行服务系统，IC 智能卡替代古老的磁卡而服务于大众已日渐成熟。并且“一卡通”，一卡多用，给我们的生活质量带来了很大的提高。IC 智能卡自动电表抄表系统，煤气/自来水抄表系统，公交/地铁自动售票/检票系统，移动通信手机中 IC 智能 SIM 卡等等，IC 智能卡已愈来愈贴近我们的生活，成为我们生活的一步分。“刷卡”已成为人们日常生活中不可或缺的一部分。

近几年来，随着 IC 智能卡中的接触式 CPU 卡以及非接触式 IC 智能射频卡（内建 MCU，ASIC 等）的高度安全保密性，使之在 IC 智能卡领域中异军突起，成为当今 IC 智能卡中的流行宠物，应用前景十分广阔。

国外对 IC 卡的研究和应用较早，特别是在美国，欧洲国家等。IC 卡遍布社会各个方方面面。就象在美国，国民消费总额的 20%~30%是由“刷卡”消费完成的，由此可见 IC 智能卡流行和使用的程度。

尽管国外对 IC 卡已有多年的研究，但真正在 IC 智能卡中特别是 CPU 卡类，非接触式 IC 智能射频卡（内建 MCU，ASIC 等）方面的研究也仅有 1~2 年的成熟期。因为他们也看到了 CPU 卡，特别是非接触式 IC 智能射频卡（内建 MCU，ASIC 等）代表了整个“刷卡”领域的发展方向，前景光明。

我国对 IC 卡行业的发展始于 1993 年左右，当时的中央领导特别是江泽民同志高度重视 IC 卡行业，高瞻远瞩，指示要发展我国自己的 IC 卡事业，建立“金卡工程”。虽然至今也只有短短的 6 年左右时间，但已取得了不小的成就。已研制成功我国自主知识产权的较大容量的存储卡，逻辑加密卡等，但是对于尖端的 CPU 卡及非接触式 IC 智能射频卡（内建 MCU，ASIC 等）的制卡技术及其相应的读/写卡设备技术，仍处于落后状态。这极大地制约了 IC 卡行业的发展，特别是在要求有极高安全性和保密性的金融行业中，制约了我国金卡工程的实施和发展。

为了加快 IC 卡行业的发展，特别是尖端的 CPU 卡及非接触式 IC 智能射频卡（内建 MCU，ASIC 等）的推广应用，让 IC 卡行业及其他行业的同行尽快掌握 CPU 卡及非接触式 IC 智能射频卡（内建 MCU，ASIC 等）的核心技术及其读写设备的软硬件技术及应用设计，跟上国外的 IC 卡研究和应用步伐，甚至走在其前面，在上海华东计算机研究所，上海计算机协会等领导同志的关怀下，在上海华东磁记录设备电子公司吴文总经理的直接支持下，经过一段时间全身心投入的研发之后，笔者已基本比较全面地掌握了 PHILIPS 公司的非接触式 IC 智能射频卡（内建 MCU，ASIC 等）MIFARE 1 卡（S50 系列）及其卡片读写设备核心模块 MCM（Mifare Core Module）的软硬件技术。使我们发展，应用，甚至超越 PHILIPS 公司的非接触式 IC 智能射频卡（内建 MCU，ASIC 等）技术及其读写设备技术成为可能。

当今世界上非接触式 IC 智能射频卡（内建 MCU，ASIC 等）中的主流主要为 PHILIPS 公司的 MIFARE 技术，已经被制定为国际标准：**ISO/IEC 14443 TYPE A 标准**。欧洲一些较大的 IC 卡片制造商以及 IC 卡片读写器制造商以及 IC 卡片软件设计公司等（例如法国的 GEMPLUS 公司）大都以 MIFARE 技术为标准，而发展和推进 IC 卡行业。

通过阅读本文，相信有一定 IC 卡经验，技术及具有一定的计算机软/硬件技术（尤其是微处理单片机 MCU 技术）的同行定能将本文消化之后而能设计出诸多应用场合的产品，例如，门禁控制系统（Door Access System），高速公路不停车收费系统，停车场收费管理系统，地铁非接触式 IC 智能射频卡的“刷卡”读写器，机关/企业内部考勤管理系统，医疗保险管理系统，公共交通收费管理系统，出租车收费管理系统，银行服务“一卡通”等等。

第一章 Mifare 1 非接触式 IC 智能(射频)卡

一. Mifare 1 非接触式 IC 智能射频卡特点

Mifare 1 IC 智能(射频)卡的核心是 Philips 公司的 Mifare 1 IC S50（-01，-02，-03，-04）系列微模块（微晶片）。它确定了卡片的特性以及卡片读写器的诸多性能。

Mifare 1 IC 智能(射频)卡采用先进的芯片制造工艺制作。内建有高速的 CMOS EEPROM，MCU 等。卡片上除了 IC 微晶片及一副高效率天线外，无任何其他元件。

卡片上无源（无任何电池），工作时的电源能量由卡片读写器天线发送无线电载波信号耦合到卡片上天线而产生电能，一般可达 2V 以上，供卡片上 IC 工作。工作频率 13.56MHZ。

Mifare 1 射频卡所具有的独特的 MIFARE RF（射频）非接触式接口标准已被制定为国际标准：ISO/IEC 14443 TYPE A 标准。

射频卡标准操作距离为 100mm（由 MCM500 作为读写器核心模块）和 25 mm（由 MCM200 作为读写器核心模块）。与卡片读写器的通信速率高达 106Kbit/s。

Mifare 1 IC 智能(射频)卡上具有先进的数据通信加密并双向验证密码系统；且具有防重叠功能：能在同一时间处理重叠在卡片读写器天线的有效工作距离内的多张重叠的卡片。

Mifare 1 IC 智能(射频)卡与读写器通信使用握手式半双工通信协议；卡片上有高速的CRC 协处理器，符合 CCITT 标准。

卡片制造时具有唯一的卡片系列号，没有重复的相同的两张 MIFARE 卡片。

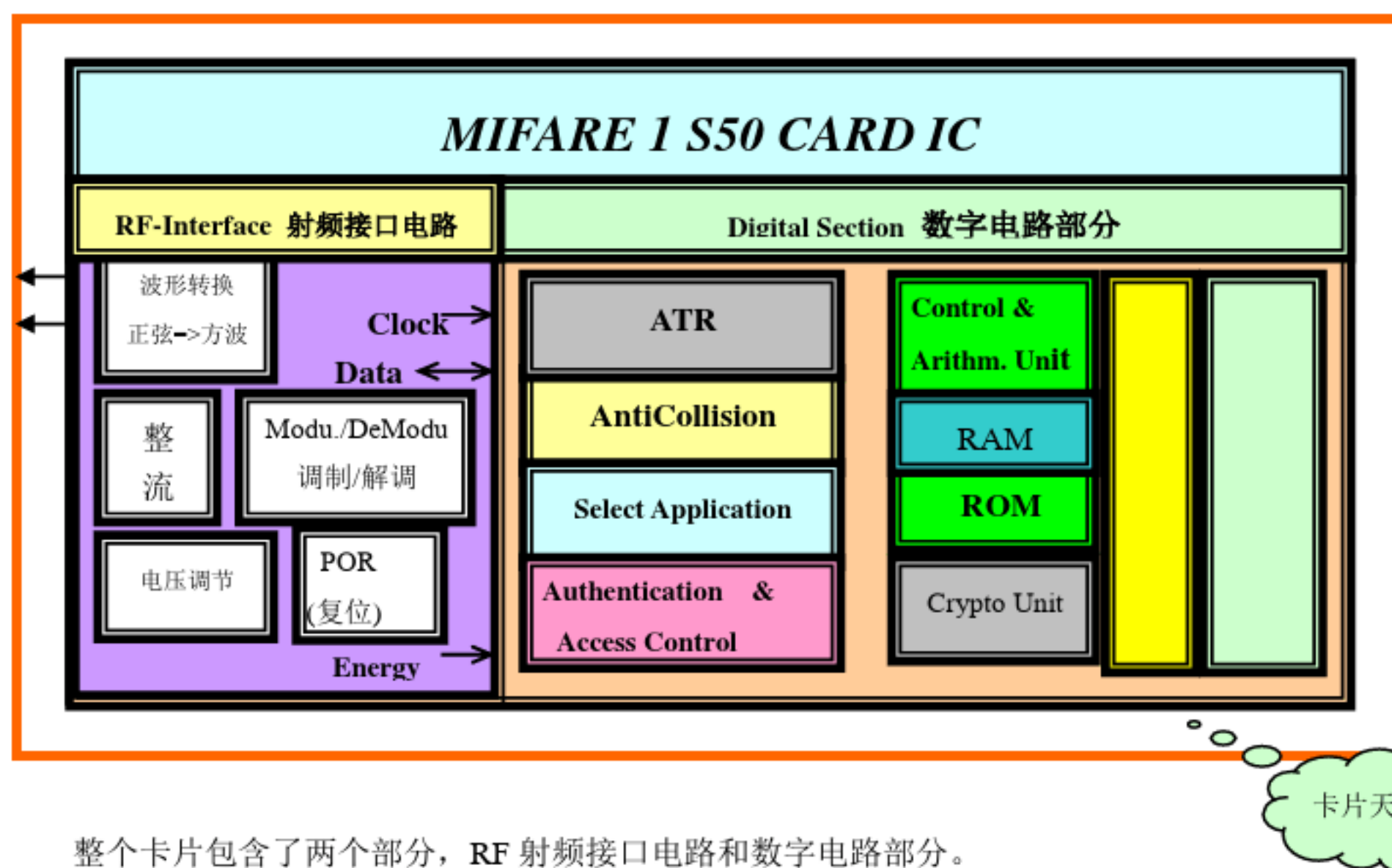
卡片上内建 8K (bit) EEPROM 存储容量并划分为 16 个扇区，每个扇区划分为 4 个数据存储块，每个扇区可由多种方式的密码管理。

卡片上还内建有增值/减值的专项的数学运算电路，非常适合公交/地铁等行业的检票/收费系统。典型的检票交易时间最长不超过 100ms(0.1 秒) (包括卡片的认证，6 个扇区的读 (768bit,2 个扇区的认证)，2 个扇区的写操作 (256bit))。

卡片上的数据读写可超过 10 万次以上；数据保存期可达 10 年以上，且卡片抗静电保护能力达 2KV 以上。

二. Mifare 1 非接触式 IC 智能射频卡功能组成

如图所示为 MIFARE 1 S50 非接触式 IC 智能射频卡的功能组成图。



整个卡片包含了两个部分，RF 射频接口电路和数字电路部分。

(一). RF 射频接口电路

在 RF 射频接口电路中，主要包括有**波形转换模块**。它可将卡片读写器上的 13.56MHZ 的无线电调制频率接收，一方面送**调制/解调模块**，另一方面进行波形转换，将正弦波转换为方波，然后对其整流滤波，由**电压调节模块**对电压进行进一步的处理，包括稳压等，最终输出供给卡片上的各电路。

POR 模块主要是对卡片上的各个电路进行 POWER-ON-RESET (上电复位)，使各电路同步启动工作。

(二)。在数字电路部分模块中：

1. **ATR 模块：Answer to Request(“请求之应答”)**

当一张 Mifare 1 卡片处在卡片读写器的天线的工作范围之内时，程序员控制读写器向卡片发出 REQUEST all(或 REQUEST std) 命令后，卡片的 ATR 将启动，将卡片 Block 0 中的卡片类型 (TagType) 号共 2 个字节传送给读写器，建立卡片与读写器的第一步通信联络。

如果不进行第一步的 ATR 工作，读写器对卡片的其他操作 (Read/Write 等) 将不会进行。

卡片的类型 (TagType) 号共 2 个字节，可能为：0004H

2. **AntiCollision 模块：防止 (卡片) 重叠功能**

如果有多张 Mifare 1 卡片处在卡片读写器的天线的工作范围之内时，AntiCollision 模块的防重叠功能将被启动工作。在程序员控制下的卡片读写器将会首先与每一张卡片进行通信，取得每一张卡片的系列号。由于 Mifare 1 卡片每一张都具有其唯一的系列号，决不会相同，因此卡片读写器根据卡片的序列号来识别，区分已选的卡片，卡片读写器中的 MCM 中的 AntiCollision 防重叠功能配合卡片上的防重叠功能模块，由程序员来控制读写器，根据卡片的序列号来选定一张卡片。被选中的卡片将直接与读写器进行数据交换，未被选择的卡片处于等待状态，随时准备与卡片读写器进行通信。

AntiCollision 模块 (防重叠功能) 启动工作时，卡片读写器将得到卡片的序列号 Serial Number。序列号 Serial Number 存储在卡片的 Block 0 中，共有 5 个字节，实际有用的为 4 个字节，另一个字节为序列号 Serial Number 的校验字节，这在以后章节中详细论述，包括对序列号 Serial Number 的校验方法等。

序列号 Serial Number 中实际有用的 4 个字节，可能为：007e0a42h。

3. **Select Application 模块：主要用于卡片的选择。**

当卡片与读写器完成了上述的二个步骤，程序员控制的读写器要想对卡片进行读写操作，必须对卡片进行 “Select” 操作。以使卡片真正地被选中。

被选中的卡片将卡片上存储在 Block 0 中的卡片的容量 “Size” 字节传送给读写器。当读写器收到这一字节后，将明确可以对卡片进行深一步的操作了。例如，可以进行密码验证等等。

读写器收到的 “Size” 字节可能为：88h

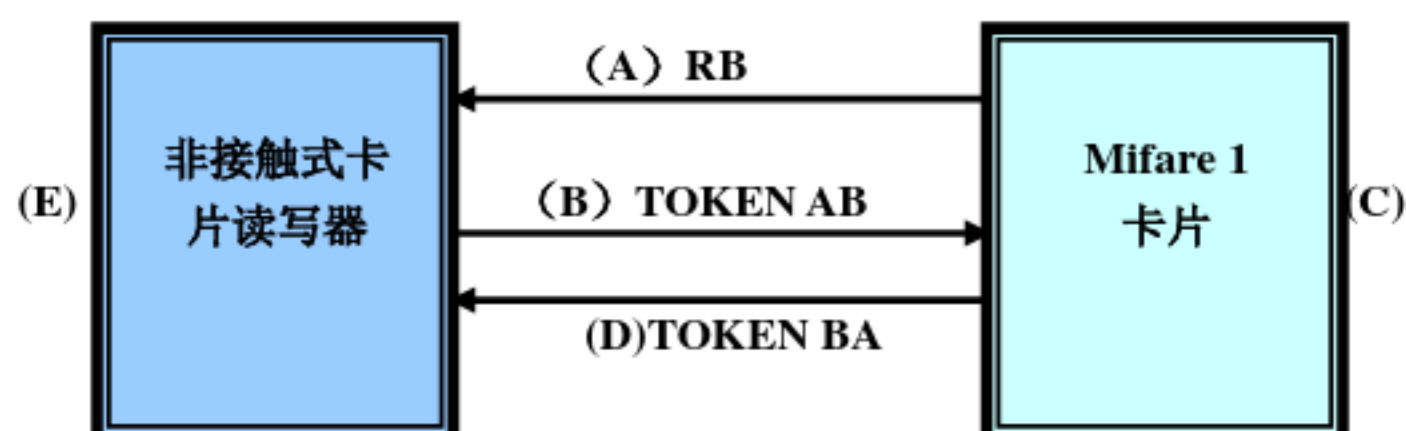
4. **Authentication & Access Control 模块：认证及存取控制模块**

在确认了上述的三个步骤，确认已经选择了一张卡片时，程序员对卡片进行读写操作之前，必须对卡片上已经设置的密码进行认证，如果匹配，则允许进一步的 Read/Write 操作。

Mifare 1 卡片上有 16 个扇区，每个扇区都可分别设置各自的密码，互不干涉。因此每个扇区可独立地应用于一个应用场合。整个卡片可以设计成 “一卡通” 形式来应用。

三遍认证：

如图所示为三遍认证的令牌原理框图。



认证过程是这样进行的：

- (A) 环：由 Mifare 1 卡片 向读写器 发送一个随机数据 RB；
- (B) 环：由读写器收到 RB 后向 Mifare 1 卡片发送一个令牌数据 TOKEN AB，其中包含了读写器发出的一个随机数据 RA；
- (C) 环：Mifare 1 卡片收到 TOKEN AB 后，对 TOKEN AB 的加密的部分进行解密，并校验第一次由 (A) 环中 Mifare 1 卡片 发出去的随机数 RB 是否与 (B) 环中接收到的 TOKEN AB 中的 RB 相一致；
- (D) 环：如果 (C) 环校验是正确的，则 Mifare 1 卡片 向读写器 发送令牌 TOKEN BA 给读写器；
- (E) 环：读写器 收到令牌 TOKEN BA 后，读写器将对令牌 TOKEN BA 中的 RB（随机数）进行解密；并校验第一次由 (B) 环中读写器发出去的随机数 RA 是否与 (D) 环中接收到的 TOKEN BA 中的 RA 相一致；

如果上述的每一个环都为“真”，都能正确通过验证，则整个的认证过程将成功。读写器将能对刚刚认证通过的卡片上的这个扇区可以进入下一步的操作（READ/WRITE 等操作）。

卡片中的其他扇区由于有其各自的密码，因此不能对其进行进一步的操作。如想对其他扇区进行操作，必须完成上述的认证过程。

认证过程中的任何一环出现差错，整个认证将告失败。必须从新开始。

如果事先不知卡片上的密码，则由于密码的变化可以极其复杂，因此靠猜测密码而想打开卡片上的一个扇区的可能性几乎为零。

这里提醒一下程序员和卡片的使用者，必须牢记卡片中的 16 个扇区的每一个密码，否则，遗忘某一扇区的密码，将使该扇区中的数据不能读写。没有任何办法可以挽救这种低级错误。但是，卡片上的其他扇区可以照样使用。

上述的叙述已经可以充分地说明了 Mifare 1 卡片的高度安全性，保密性，及卡片的应用场合多样性，一卡多用（一卡通）。

5. Control & Arithmetic Unit 控制及算术运算单元：

这一单元是整个卡片的控制中心，是卡片的“头脑”。它主要进行对整个卡片的各个单位进行微操作控制，协调卡片的各个步骤；同时它还对各种收/发的数据进行算术运算处理，递增/递减处理，CRC 运算处理，等等。是卡片中内建的中央微处理机（MCU）单元。

6. RAM/ROM 单元：

RAM 主要配合控制及算术运算单元，将运算的结果进行暂时存储，如果某些数据需要存储到 EEPROM，则由控制及算术运算单元取出送到 EEPROM 存储器中；如果某些数据需要传送给读写器，则由控制及算术运算单元取出，经过 RF 射频接口电路的处理，通过卡片上的天线传送给卡片读写器。RAM 中的数据在卡片失掉电源后（卡片离开读写器天线的有效工作范围内）将被清除。

同时，ROM 中还固化了卡片运行所需要的必要的程序指令，由控制及算术运算单元取出去对每个单元进行微指令控制。使卡片能有条不紊地与卡片的读写器进行数据通信。

7. Crypto Unit 数据加密单元：

该单元完成对数据的加密处理及密码保护。加密的算法可以为 DES 标准算法或其他。

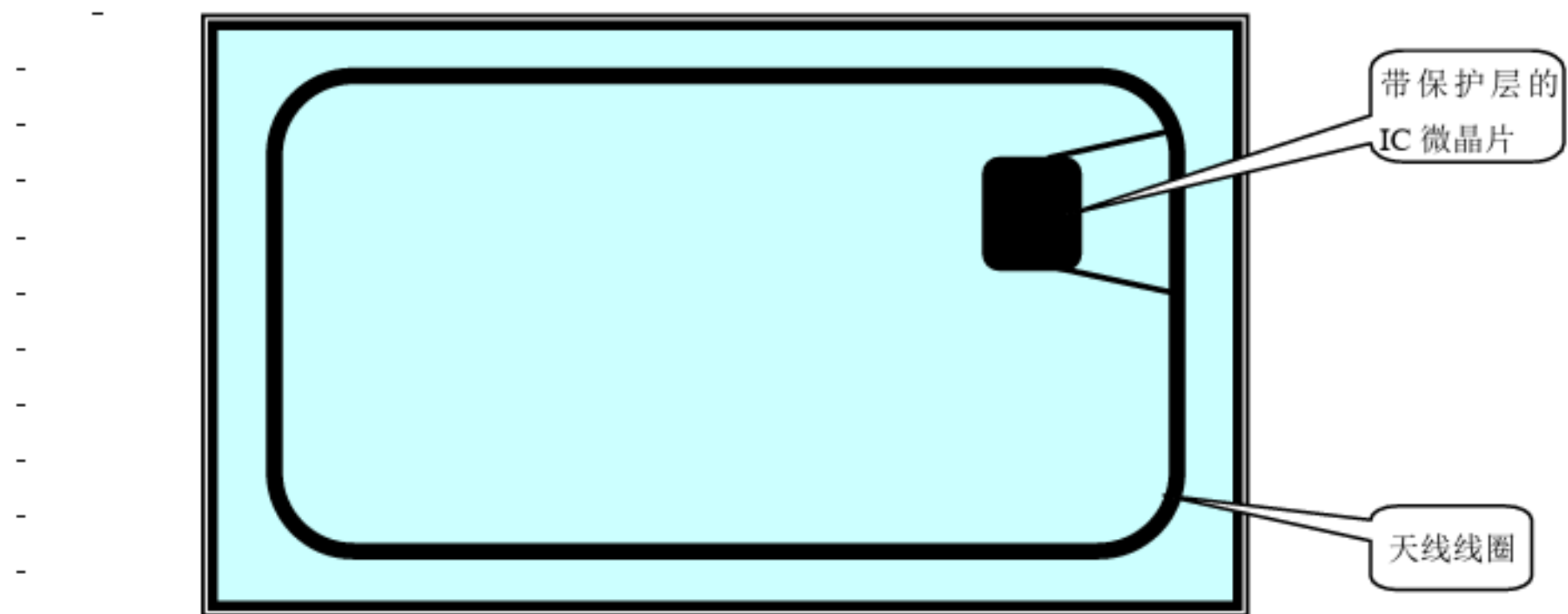
8. EEPROM INTERFACE/EEPROM MEMORY EEPROM 存储器及其接口电路：

该单元主要用于存储数据。EEPROM 中的数据在卡片失掉电源后（卡片离开读写器天线的有效工作范围内）仍将被保持。用户所要存储的数据被存放在该单元中。Mifare 1 卡片中的这一单元容量为 8196bit(1 Kbyte)。分为 16 个扇区。

三. Mifare 1 IC 智能(射频)卡的物理组成及卡片上天线的研究

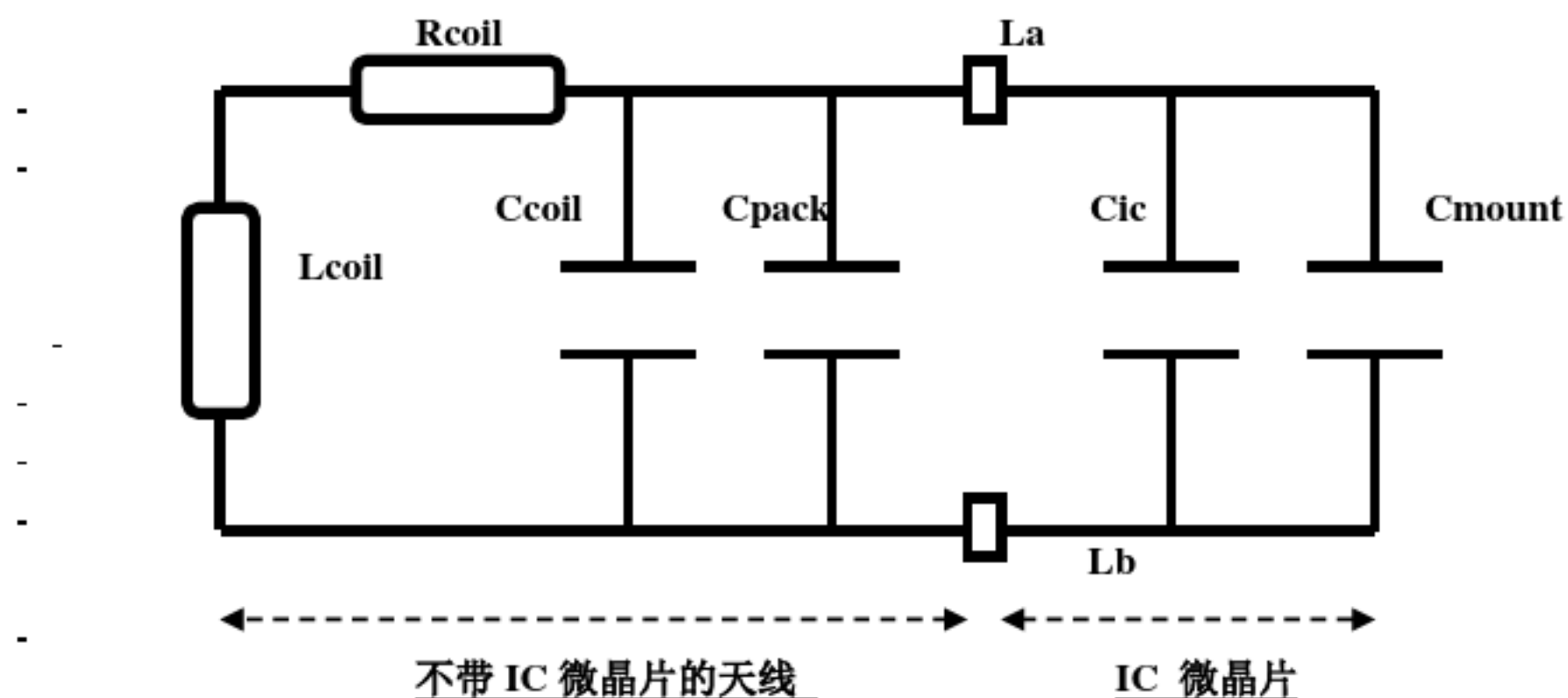
在卡片上的微晶片外面一般封装了保护层。保护层可以防止微晶片被折叠，扭曲等众多对卡片实施非正常的物理性损坏。同时也防止微晶片受到紫外线的辐射，使卡片能长久地被使用。

但从电性能的角度来看，由于加入了保护层，使 IC 与卡片上的天线组成的振荡回路的频率将发生变化。因为，保护层给 IC 微晶片增加了一个输入回路电容 C_{mount} 。尽管这个电容只有几个 pf 至几十个 pf，但对于要求频率精度，稳定度等都很高的非接触式 IC 智能射频卡来说，也将是很重要的。



Mifare 1 IC 智能(射频)卡

卡片上的等效电路如下：



整个卡片的自由振荡频率计算方法：

$$F_{res} = \frac{1}{2 * 3.1415926 * (Lcoil * (Ccoil + Cpack + Cic + Cmount))^{1/2}}$$

说明：

- F_{res} : 卡片的振荡频率，应为 13.56MHz

- Rcoil : 天线线圈的电阻, 约 6.07 欧姆
- Lcoil : 天线线圈的电感, 约 3.6 μH
- Ccoil : 天线线圈的电容, 约 5pf
- Cpack : 天线线圈的封装后引入的电容, 约 5pf
- Cic : IC 微晶片的电容, 约 16pf
- Cmount : IC 微晶片的安装后引入的电容, 约 几个 pf~ 几十个 pf
- La : 天线线圈与 IC 微晶片的接触点 a
- Lb : 天线线圈与 IC 微晶片的接触点 b

其中,

天线线圈的电感:

$$L_{\text{coil}} = 2 * L [\text{cm}] * (\ln (L [\text{mm}] / D [\text{mm}] - 1.04) * N^P$$

- L : 天线线圈一圈的长度
- N : 天线线圈圈数, 一般为 4 圈
- D : 天线线圈直径或导体的宽度
- P : 由天线线圈的技术而定的 N 的指数因子, 如下表所示:

P 值:	天线线圈结构:
1.8	环绕线圈
1.7	Etched(蚀刻)线圈
1.5 ~ 1.7	印刷电路板线圈

上述天线线圈的电感的公式只能作为首次估测之用, 实际的天线线圈的电感必须通过仪器测量而定, 但偏差不会很大。

一般 天线线圈的电感 : $L_{\text{coil}} < 4.2 \mu\text{H}$, 实际中推荐在 $3.6 \mu\text{H}$ 左右为最优。

天线线圈的品质因数 Qcoil :

$$Q_{\text{coil}} = \frac{\omega * L_{\text{coil}}}{R_{\text{coil}}} = \frac{2\pi f_{\text{res}}}{R_{\text{coil}}} = \frac{2\pi * 13.56}{R_{\text{coil}}}$$

一般天线线圈的品质因数 $30 < Q_{\text{coil}} < 60$ 。

实际中, 品质因数 Qcoil 在>30 后的增加量, 对卡片的操作距离的增加无明显帮助; 品质因数 Qcoil 必须 < 60, 以确保数据通信稳定, 可靠, 否则天线的有效工作距离内有死区, 而不能可靠地进行数据通信。

天线线圈的矩形面积 S 总:

$$S_{\text{总}} \text{ 必须 } > 11200 \text{ mm}^2 \quad S_{\text{总}} = S_{\text{平均}} * N_{\text{圈}}$$

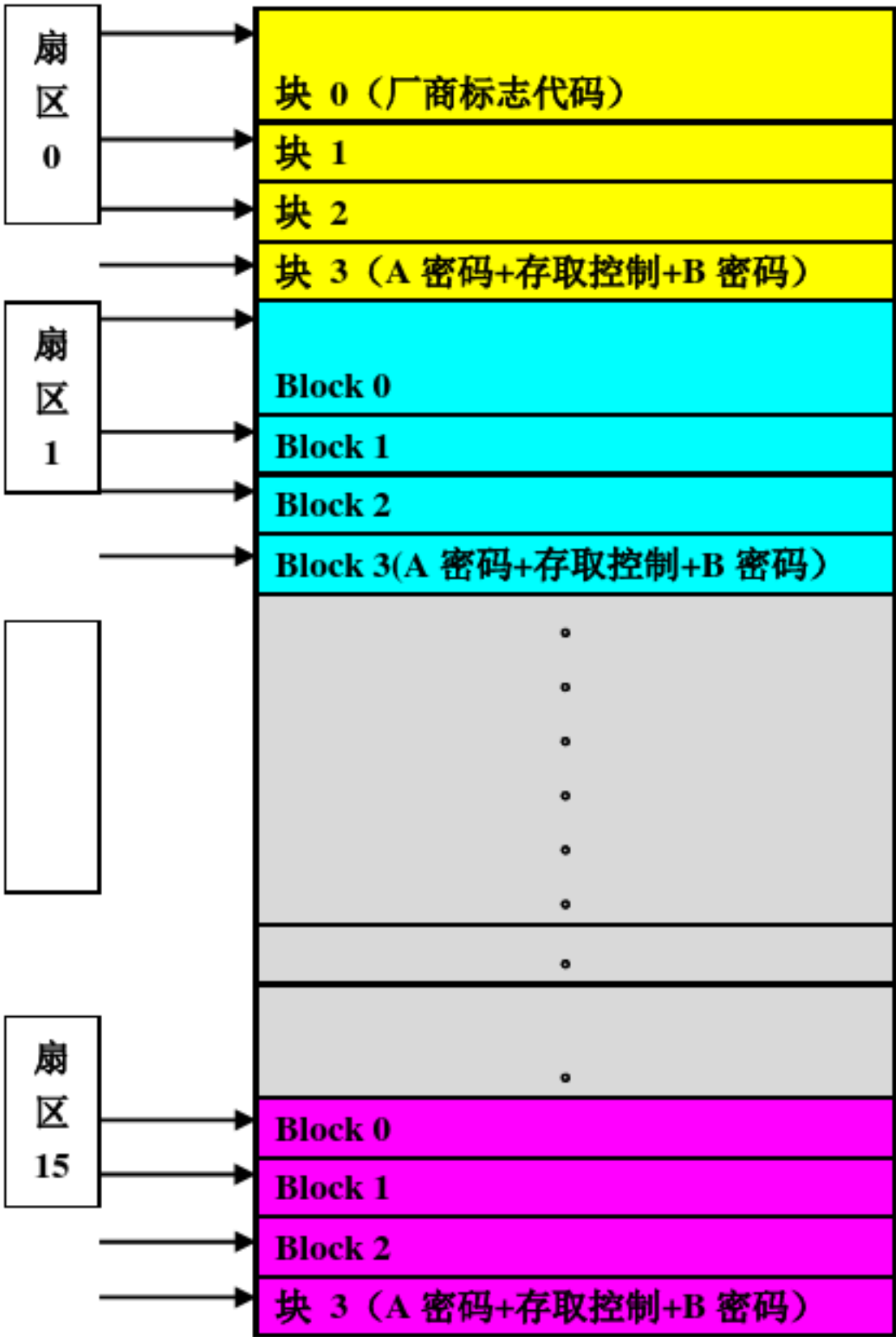
所以 $S_{\text{平均}}$ 必须 $> 11200 \text{ mm}^2 / N_{\text{圈}}$

一般, $S_{\text{平均}} \geq 2778 \text{ mm}^2$ (当卡片上的天线线圈的矩形面积的 长 X 宽 = 40mm X 70 mm 时)。

实际中, 推荐 $S_{\text{平均}}$ 在 3330 mm^2 左右。即, 实际设计时, 卡片上的天线线圈的矩形面积的长和宽应为: 74mm 和 45mm, 天线线圈的圈数 N 为 4 圈, 则这样制作出的卡片将能保证通信的距离。

四. Mifare 1 卡的存储结构

Mifare 1 卡的存储容量为 8192 BIT X 1 位字长（即 1K X 8 位字长），采用 EEPROM 作为存储介质，整个结构划分为 16 个扇区，编为扇区 0~~15。每个扇区有 4 个块（Block），分别为块 0,块 1,块 2 和块 3。每个块有 16 个字节。一个扇区共有 16 Byte X 4 = 64 Byte。如下图：



每个扇区的块 3(即第四块) 包含了该扇区的密码 A(6 个字节)、存取控制(4 个字节)、密码 B(6 个字节)，是一个特殊的块。其余三个块是一般的数据块。

但扇区 0 的块 0 是特殊的，是厂商代码，已固化，不可改写。
其中： 第 0~4 个字节为卡的序列号，第 5 个字节为序列号的校验码；
第 6 个字节为卡的容量“SIZE”字节；
第 7，8 个字节为卡的类型号字节，即 Tagtype 字节；
其他字节由厂商另加定义。

Mifare 1 卡的扇区 0 中的块 0 (Block 0) 存储的 16 个字节的内容可能为：
420a7e00368804004481740630373937h。

下面将对密码 A，密码 B，存取控制与数据区的关系加以说明：

存取控制的结构如下：(四个字节中的各位依次命名为)

位: 7	6	5	4	3	2	1	0
<i>C2X3_b</i>	<i>C2X2_b</i>	<i>C2X1_b</i>	<i>C2X0_b</i>	<i>C1X3_b</i>	<i>C1X2_b</i>	<i>C1X1_b</i>	<i>C1X0_b</i>
C1X3	C1X2	C1X1	C1X0	<i>C3X3_b</i>	<i>C3X2_b</i>	<i>C3X1_b</i>	<i>C3X0_b</i>
C3X3	C3X2	C3X1	C3X0	<i>C2X3</i>	<i>C2X2</i>	<i>C2X1</i>	<i>C2X0</i>
BX7	BX6	BX5	BX4	BX3	BX2	BX1	BX0

_b 表示取反，如 C2X3_b 即 C2X3 取反；X 表示扇区号；Y 表示第几块；C 表示控制位；B 表示备用位；

存取控制对块 3 的控制如下：(X=0-15)

			密码 A	密码 A	存取控制	存取控制	密码 B	密码 B
C1X3	C2X3	C3X3	read	Write	read	write	read	write
0	0	0	never	KEYA B	KEYA B	never	KEYA B	KEYA B
0	1	0	never	Never	KEYA B	never	KEYA B	never
1	0	0	never	KEYB	KEYA B	never	never	KEYB
1	1	0	never	Never	KEYA B	never	never	never
0	0	1	never	KEYA B	KEYA B	KEYA B	KEYA B	KEYA B
0	1	1	never	KEYB	KEYA B	KEYB	never	KEYB
1	0	1	never	Never	KEYA B	KEYB	never	never
1	1	1	never	Never	KEYA B	never	never	never

KEYA|B 表示密码 A 或密码 B；never 表示没有条件实现。

对数据块的控制如下表：

(X=0-15 扇区、Y=每个扇区的 0-2 块)

C1XY	C2XY	C3XY	Read	Write	Increment	Decr,Transfer,restore
0	0	0	KEYA B	KEYA B	KEYA B	KEYA B
0	1	0	KEYA B	never	never	never
1	0	0	KEYA B	KEYB	never	never
1	1	0	KEYA B	KEYB	KEYB	KEYA B
0	0	1	KEYA B	never	never	KEYA B
0	1	1	KEYB	KEYB	never	never
1	0	1	KEYB	never	never	never
1	1	1	Never	never	never	never

块 3 的初始化值为: a0,a1,a2,a3,a4,a5,ff,07,80,69,b0,b1,b2,b3,b4,b5 共 16 个字节,其中 KEYA 是 {a0,a1,a2,a3,a4,a5}，KEYB 是 {b0,b1,b2,b3,b4,b5}，控制存取的四个字节为 {0xff,0x07,0x80,0x69}。

存取控制位表示如下(请用户去对照上表以确定初始化状态):

位: # 7	6	5	4	3	2	1	0	Ff h
								07h
								80h
								69h

1	1	1	1	1	1	1	1
0	0	0	0	0	1	1	1
1	0	0	0	0	0	0	0
0	1	1	0	1	0	0	1

程序员可以根据自己应用的具体情况，对不同的扇区可选用不同的存取控制，不同的密码，但应注意其每一位的格式，以免误用！

数据块有两种应用方法，一种是用作一般的数据保存用，直接读写。另一种用法是用作数值块，可以进行初始化值、加值、减值、读值的运算。系统配用相应的函数完成相应的功能。

MCM 内核技术说明

第二章 MCM 的硬件内核电路

一. MCM200/MCM500 读写模块说明

Mifare Core Module 是 MCM 的全称，意为 Mifare 核心模块。

Philips 公司的 MCM 主要有两种产品型号，为 MCM200 和 MCM500。这两种智能模块均被用于读写 Mifare 1 非接触式 IC 智能射频卡的读写器中，负责读写器中对非接触式 IC 智能射频卡片的读写等功能，一般在读写器中还必须有 MCU（微处理单片机）来对 MCM 进行控制，及对读写器的其他方面进行控制，例如对键盘，显示，通信等部分的控制等等。

MCM200 模块主要应用于对卡片操作距离在 25mm 的卡片读写器中；

MCM500 模块主要应用于对卡片操作距离在 100mm 的卡片读写器中。

Mifare 所具有的独特 MIFARE RF（射频）非接触式接口标准已被制定为国际标准：ISO/IEC 14443 TYPE A 标准。

（一）。MCM200 模块说明

A. 特性：

1. 标准的双列直插 32 引脚
2. 工作频率：13.56MHZ
3. 标准的+5V 电源供电，供电范围 4.75V 至 5.25V
4. （典型）电流消耗 40MA，最大不超过 80MA，最小 10MA 左右
5. 读写卡片距离可达 25MM 以上
6. 与卡片的通信速率可达 106KBps
7. 模块与卡片通信时，数据加密
8. 每个扇区设有 3 套密码及其认证和密码存储器
9. 有防卡片重叠功能
10. 16 个字节的 FIFO（先进先出）队列接收/发送缓冲寄存器
11. 在模块与卡片通信时自动侦查错误，自动对数据流分析
12. 对 RF（射频）通道自动监控
13. 内建 8 位/16 位的 CRC 协处理器，提供 CRC，PARITY 等数据校验
14. 支持多种方式的活动的天线，并且不需“天调系统”（天线调节系统）对天线进行补偿调节
15. 标准的 MIFARE 并行接口与 MCM500 100%全兼容
16. MCM200 的软件与 MCM500 模块 100%全兼容
17. 可控制，级联 MCM500 模块
18. 工作温度范围在：-20 ~ +70 度

B. MCM200 模块引脚说明

如图所示是该模块的引脚排列示意图：

D7	1	32	KPROG
D6	2	31	NCS
D5	3	30	NWR
D4	4	29	NRD
D3	5	28	A0
D2	6	27	A1
D1	7	26	A2
D0	8	25	A3
BP	9	24	ALE
NPAUSE1	10	23	USEALE
KOMP1	11	22	MODE
NRST	12	21	NANT
DVDD	13	20	ANT
DGND	14	19	AVDD
NIRQ	15	18	AGND
TP	16	17	RX

MCM200 模块

MCM200 引脚说明如下：

引脚	引脚名称	引脚类型	将来使用类型	引脚	引脚名称	引脚类型	将来使用类型
1	D7	I/O(PU)	I/O(PU)	17	RX	I(analog)	I(analog)
2	D6	I/O(PU)	I/O(PU)	18	AGND	SUPPLY	SUPPLY
3	D5	I/O(PU)	I/O(PU)	19	ACDD	SUPPLY	SUPPLY
4	D4	I/O(PU)	I/O(PU)	20	ANT	O	O
5	D3	I/O(PU)	I/O(PU)	21	NANT	O	O
6	D2	I/O(PU)	I/O(PU)	22	MODE	I(PU)	I(PU)
7	D1	I/O(PU)	I/O(PU)	23	USEALE	I(PU)	I(PU)
8	D0	I/O(PU)	I/O(PU)	24	ALE	I(PU)	I(ST,PU)
9	Bp	SUPPLY	SUPPLY	25	A3	I(PU)	I(ST,PU)
10	NPAUSE1	O	O	26	A2	I(PU)	I(ST,PU)
11	KOMP1	I(PU)	I (PU)	27	A1	I(PU)	I(ST,PU)
12	NRST	I	I	28	A0	I(PU)	I(ST,PU)
13	DVDD	SUPPLY	SUPPLY	29	NRD	I(PU)	I(ST,PU)
14	DGND	SUPPLY	SUPPLY	30	NWR	I(PU)	I(ST,PU)
15	NIRQ	O	O	31	NCS	I(PU)	I(ST,PU)
16	TP	O(analog)		32	KPROG	NC	SUPPLY

上述表中：I ----- 输入引脚

O ----- 输出引脚

SUPPLY -----电源供电输入

I/O -----输入/输出双向引脚

PU -----Internal Pull-up 内部有上拉电阻

C. 其他说明

MCM200 是 Philips 公司的产品，另外还有其他公司与之相兼容的产品。

UniVision Engineering Limited (联视工程有限公司)持有 Mifare 制造专利，其生产的相当与 MCM200 的产品命名为 SB201。

SB201 与 MCM200 模块在硬件上有一处不同，表现在软件上有一个 MODE 方式寄存器的设置不同。SB201 的 MODE 设置为 0xD6H；MCM200 的设置 0xC6H（这在以后的章节中还会论述）。除此以外，两者软/硬件 100%全兼容。

法国的 GEMPLUS 公司向 PHILIPS 公司购买 MCM 中核心 ASIC RC150 或 RC170 芯片制造的 MCM200 模块及读写器与 MCM200 全兼容等。

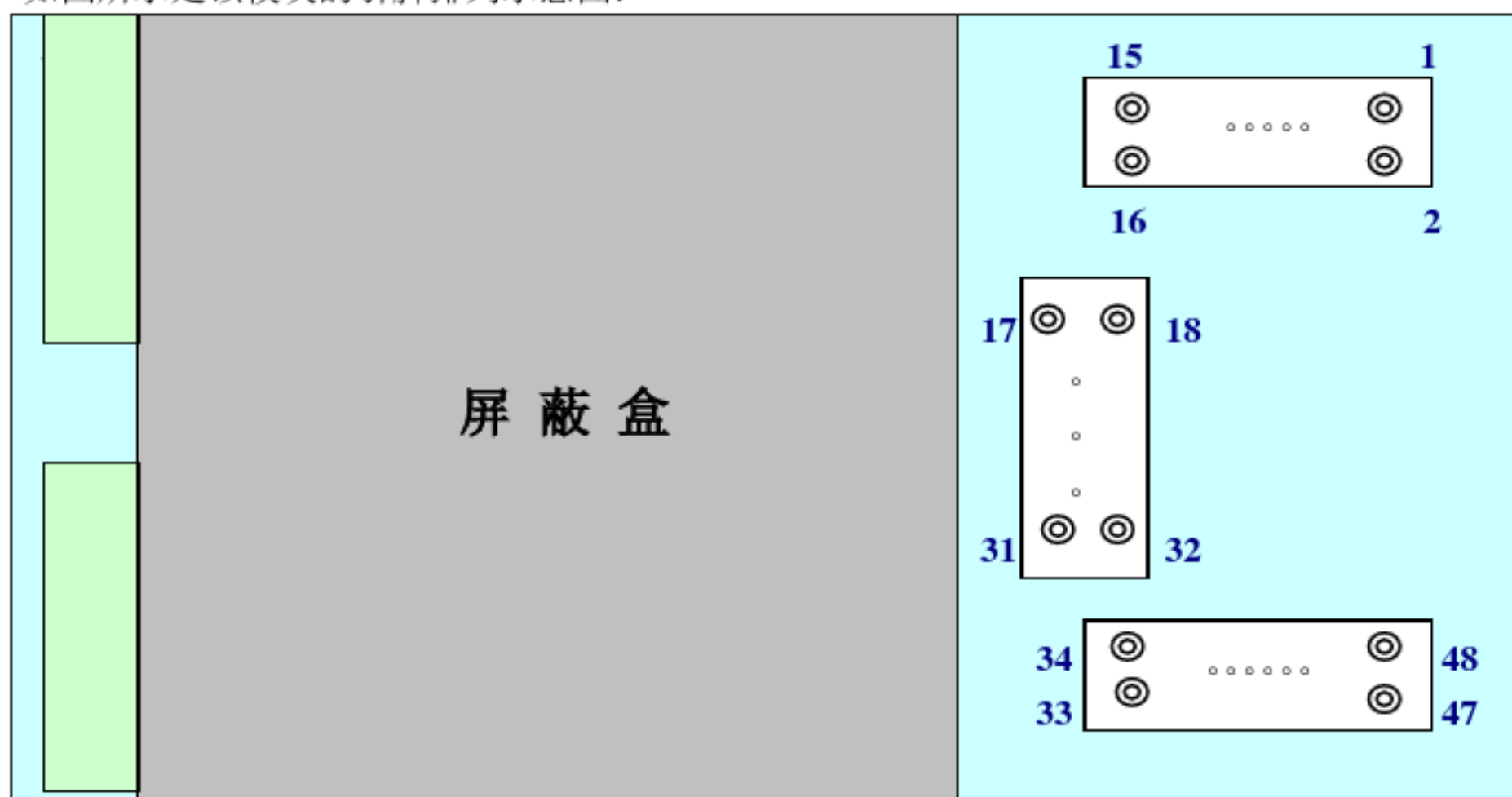
(二) . MCM500 模块说明

A.. 特性:

1. 工作频率：13.56MHZ
2. 标准的+12V 电源供电，
3. 工作频率：13.56MHZ
4. 读写卡片距离可达 100MM 以上
5. 与卡片的通信速率可达 106KBps
6. 模块与卡片通信时，数据加密
7. 每个扇区设有 3 套密码及其认证和密码存储器
8. 有防卡片重叠功能
9. 16 个字节的 FIFO（先进先出）队列接收/发送缓冲寄存器
10. 在模块与卡片通信时自动侦查错误，自动对数据流分析
11. 对 RF（射频）通道自动监控
12. 内建 8 位/16 位的 CRC 协处理器，提供 CRC，PARITY 等数据校验
13. 支持多种方式的有源天线，并且不需“天调系统”（天线调节系统）对天线进行补偿调节
14. 标准的 MIFARE 并行接口与 MCM200 100%全兼容
15. MCM200 的软件与 MCM500 模块 100%全兼容
16. 可遥控，级联 MCM500 模块
17. 工作温度范围在：-20 ~ +70 度

B. MCM500 模块引脚说明

如图所示是该模块的引脚排列示意图：



如图所示是该模块的引脚排列示意图：

引脚	引脚名称	引脚类型	功 能 描 述
1	NPAUSE1	O	串行数据输出，驱动 RF1 单元，该引脚必须连到 RF 单元的 TxNPAUSE 端。该引脚的“LOW”信号指示一个能量暂停。
2	NWR	I	写信号使能端
3	GATE1	O/NC	（未使用）
4	NRD	I	读信号使能端
5	MOD1	O/NC	（未使用）
6	MODE	I	并行协议模式，必须用“HIGH”高电平来驱动。
7	SYNC1	I/NC	（未使用）
8	NCS	I	(Not Chip Select)当 MCU 用“LOW”低电平来驱动 该引脚时，将选中 MCM
9	WKOMP1	I	（未使用）
10	CS	I	(Chip Select)当 MCU 用“HIGH”高电平来驱动 该引脚时，将选中 MCM
11	KOMP1	I	(Compartor input of RF1)RF1 的比较器输入端，必须连到 RF 单元的 RXKOMP 端
12	N.T.C.		（未使用）
13	DGND		数字电路接地端
14	N.T.C.		（未使用）
15	DVDD		电源供应端+5V。内部在 RxKOMP 端上有上拉电阻。
16	CLOCK	O/NC	（未使用）
17	RFVDD		+12V 电源输入，供给 RF 射频单元
18	N.T.C.		（未使用）
19	RFGND		RF 射频单元（模拟电路）接地端
20	N.T.C.		（未使用）
21	RxKOMP	O	RF 射频单元的输出信号端。它提供与数字电平兼容的接收信号。其对地（DGND）为集电极开路输出，并有上拉电阻 2K2 接到 DVDD 端
22	KOMP0	I	(Compartor input of RF0)RF0 的比较器输入端，必须连到 RF 单元的 RXKOMP 端
23	RxWKOMP	O	（未使用）
24	WKOMP0	I	（未使用）
25	SYNC	O/NC	（未使用）

制造的 MCM500 模块及读写器与 MCM500 全兼容。 等等。

二. MCM 的硬件内核寄存器描述

MCU 是通过对 MCM 内核特殊的内存寄存器的读写来控制 MCM 的。这些寄存器位于 MCM 中的 ASIC（特殊应用 IC）的内部。共有 16 个寄存器可存取。在对 MCM 进行读/写操作时，各寄存器担负着不同的功能和作用,并且不是所有的寄存器都是可写或可读的,即有些寄存器只能读而不能写,有些则反之。

读取 MCM, 一般的 指向 MIFARE 1 卡请求。MCM 实际上是 MCU 与非接触式 IC 智能(射频)卡之间信息(数据)交换的”中间人”。任何读取卡片上的数据,或写进卡片上的数据,均必须通过 MCM 来传递。写 MCM 意味着去控制 MCM，例如，送一些类型的指令给它。

以下是 MCM 中 ASIC 内核特殊寄存器一览表。

MCM—ASIC 寄存器				
寄存器名	地址	读 (READ)	写 (WRITE)	
DATA	00H 0	READ-BYTE	WRITE-BYTE	
STACON	01H 1	DV TE PE CE BE AE -- --	SOR RFS -- -- 1 1 NRF AC	
ENABLE	02H 2	N/A	1 PR CE CR -- -- -- --	
BCNTS	03H 3	N/A	BIT-COUNT-SEND	
BCNTR	04H 4	N/A	BIT-COUNT-RECEIVE	
BAUDRATE	05H 5	N/A	-- -- -- -- 1 1 1 BR	
TOC	06H 6	N/A	TIMEOUT-COUNTER	
MODE	07H 7	N/A	1 1 0 0 0 P2 P1 P0	
CRCDATA	08H 8	CRC-BYTE-READ	CRC-BYTE-WRITE	
CRCSTACON	09H 9	CV -- -- -- -- -- CZ	C8 -- -- -- -- -- CR	
KEYDATA	0AH 10	N/A	KEY-BYTE-WRITE	
KEYSTACON	0BH 11	-- -- -- -- --	AL 0 -- -- -- -- KS1 KS0	
KEYADDR	0CH 12	N/A	AL AB A5 A4 A3 A2 A0	
	0DH			

	13			
RCODE	0EH	N/A	-- -- -- -- 0 0 RC1 RC0	
	14			
	0FH			
	15			

表 1：MCM-ASIC 寄存器

各种特殊寄存器材的具体说明如下：

1. DATA 寄存器 地址:00H

寄存器名	地址	读 (READ)	写 (WRITE)
DATA	00H 0	READ-BYTE	WRITE-BYTE

任何传递到 MIFARE 1 卡的数据 或 来自 MIFARE 1 卡的数据 都必须分别地 被写入 DATA 寄存器 或 从 DATA 寄存器中 读出。

例如, MCU 向卡片写数据 Xi, 则 Xi 必须首先被写入到 DATA 寄存器中, 然后在 MCM 与 Mifare 1 卡片进行通信, 由卡片上的 ASIC 将 DATA 寄存器中的数据读取并存放在指定的卡片上的存储器中, 从而完成 MCU 向 MIFARE 1 卡片写数据的过程。 反之亦然。

DATA 寄存器中有一个 16 BYTE 的 FIFO (先进先出) 队列寄存器。 数据写到 DATA 寄存器后,被存放在这个 16 BYTE 的 FIFO (先进先出) 队列寄存器中, 等待向卡片上或 MCU 传送。

如要进行传送 DATA 寄存器中数据, 则首先必须先设定要传输 多少位(BIT) 或多少字节(BYTE), 即必须对 BCNTS 寄存器进行有效的正确设置,以确定有多少字节将要被传送。

当卡片上的数据流接收 结束或没有被接收时, STACON 寄存器中的 DV 标志被置位“1”, 并且 MCM 上的引脚 NIRQ 也将由“HIGH”变为“LOW”。

如果设置了 DV-bit, 则 STACON 寄存器中的 error-bits (出错位) 将被设置。如果没有相关的 error-bit 被设置, 则 FIFO 寄存器中的数据将被假设为正确, 且可以被读出。

2. STACON 寄存器 地址:01H

SATCON 是 STATUS CONTROL (状态控制)的缩写,是一个状态控制寄存器。

寄存器名	地址	读 (READ)	写 (WRITE)
STACON	01H 1	DV TE PE CE BE AE -- --	SOR RFS -- -- 1 1 NRF AC

将数据写到 STACON 寄存器中将对 MCM 进行状态控制,见如下表:

Bit Name 位名	Name 名称	Function 功能
SOR	Soft Reset (软复位)	“0” ----表示无作用 “1”; ----reset MCMA1,自动地清楚一切
RFS	RF-Select (RF 选择)	“0” ----RF 单元 0 被选中激活 “1” ---- RF 单元 1 被选中激活

NRF	NO RF (无 RF)	“0” ----将活动（激活）RF 单元接上 “1” ----将活动（激活）RF 单元关闭
AC	Anti Collision (防重叠)	“0” ----无作用 “1” ----启动防重叠状态机

这里,应该注意,当写数据到 **STACON 寄存器**时（即对 **STACON 寄存器**进行设置），在写数据到 **STACON 寄存器**的这一写周期中，RFS 位的设置必须与 RF 单元相互一致。

另外，设置了 AC 位，即“AC=1”，将使**防重叠**状态机启动工作，执行内部存取。这一存取在设置 AC 位后的 12us 开始，至 35us 结束。在这一间隙时间，不允许 MCU 向 MCM 进行写（WRITE）数据操作。必须延迟 35us。这是为了保证**防重叠**状态机处理软件的正确运行，让其能识别叠放在一起的多张 Mifare 1 卡片，为下一步进行选择一张指定 SN(Serial Number)的 Mifare 1 卡片进行数据读/写而作准备。Philips 公司(或 UniVision Engineering Limited 联视工程公司)推荐至少延迟 35us，然后进行写 DATA，TOC 寄存器等操作。

读取 **STACON 寄存器**，将告诉我们一系列的 MCM 及卡片的当前状况信息。以下是读取 **STACON 寄存器**后的一系列标志位表。

Bit Name	位名	Name 名称	Function 功能
DV		Data Valid (数据有效)	卡上的数据传输到 FIFO 寄存器中 或一个定时计数器溢出（以在确定 TE 标志时，是二者中的那一个溢出）。DV 位也可以在正确的 Authentication(认证)操作完成后而被设置。
TE		Timeout Error (溢出出错)	一个定时计数器溢出发生。
PE		Parity Error (奇/偶校验错)	在通讯时有奇/偶校验错
CE		CRC Error (CRC 错)	在通讯时有 CRC 错
BE		Bitcount Error (位计数器出错)	在 BCNTR 寄存器中有大量的指定 bits(位)没有收到。
AE		Authentication (认证出错)	卡的认证应答码不正确或当执行一个取密码 (KEY Error Load) 操作时指定的 ROM KEY(只读存储器中密码)不正确。

在上述表中,仅当 DV(Data Valid 数据有效)位被设置有效时(即 DV="1")，TE, PE, CE, BE 及 AE 标志才有效。但有一例外，即 当**认证（Authentication）**正确完成后，AE 标志也直接有效。

SOR 是 Software Reset(软件复位)的缩写。当执行 SOR 后，所有的这些标志将被清除，且当对 MCM 的任何一个**寄存器**执行写周期时，所有的这些标志也将被清除。详见 SOR 后的复位表。

当对 MCM 内的任何一个寄存器执行写周期时,所有的这些标志也将被清除;仅当 FIFO 是空时,DV 标志将被写周期清除。

有时候 MCM 接收少于 8 bits(例如 在接收 ACK 或 NACK 时)时，MCM 将不影响 PE, CE, 及 AE 标志。

在 MCU 与 MCM, MCM 与 Mifare 1 卡片进行数据通信时,经常会出现各种错误。例如,卡片没能认证(Authentication) 通过,则会使 AE 出错位置位,等等。

以下是在检查 DV 位之后,相关的**出错**情况,列为下表:

Command Name (命令名):	Relevant Error Flags (相关的出错标志):
---------------------	---------------------------------

Answer to Request(Request 应答)	TE,BE
Anti Collision (防重叠)	TE,BE
Select Card (卡片选择)	TE, BE, PE, CE
Authentication (认证)	TE, BE, PE, AE
Read Command (读命令)	TE, BE
Read DATA (读数据)	TE, BE, PE, CE
Write Command (写命令)	TE, BE
Write DATA (写数据)	TE, BE
InCR。/DeCR。/Rest。 Command (增/减/复位 命令)	TE , BE
InCR。/DeCR。/Rest。 Value (增/减/复位 值)	TE , BE
Halt (停机)	TE , BE

出 错 表

注意,送到卡片上的所有的命令和数据都提供奇/偶校验(**parity**)和 **CRC** 检查 (但“**REQUEST**”命令操作除外)。这是由卡片上的 **ASIC** 负责检查。

3. **ENABLE 寄存器, 地址:02H**

ENABLE 寄存器的设置将影响卡片在通信时对 **parity** 和 **CRC** 的校验。在 **ENABLE 寄存器**中有针对 **parity** 和 **CRC** 校验的复位允许位 (reset **ENABLE** bits)。

寄存器名	地址	读 (READ)	写 (WRITE)
ENABLE	02H 2	N/A	1 PRE CEN CRE -- -- -- --

对 **ENABLE 寄存器**进行写操作, 将执行对 **parity** 和 **CRC** 块的控制。

Bit Name 位名	Name 名称	Function 功能
PR	Parity reset (Parity 复位)	“0”-----无效 “1”-----复位 parity 块, 自动清除
CE	CRC ENABLE (CRC 使能)	“0”----- 关闭 CRC 发生器及校验 “1”----- 打开 CRC 发生器及校验
CR	CRC Reset (CRC 复位)	“0”----- 无效 “1”----- 复位 CRC 块, 自动清除

在对 **MCM** 进行编程的开始时,必须对 **ENABLE 寄存器**进行写操作, 即必须将 **CE** 位关闭。仅当执行”Select”命令操作时才打开 **CE** 位:

由于 **MCM** 在电源接通时或在任何数据通信开始时,都会复位 **parity** 和 **CRC** 块, 因此无须额外地去执行这一操作。

4. **BCNTS 寄存器, 地址: 03H**

寄存器名	地址	读 (READ)	写 (WRITE)
BCNTS	03H 3	N/A	BIT-COUNT-SEND

BCNTS 是英文 Bit-Counter-for-Sending 的缩写,意为传送时的位计数器(寄存器)。 这里的

传送指的是 CPU 向 MCM 的 DATA 寄存器写数据。因此 BCNTS 寄存器实际上是一个字节发送控制器,它控制了 MCU 向 DATA 寄存器中写进的数据字节数目。

例如,设置 BCTRS=10H,则可向 MCM 的 DATA 寄存器写进的数据字节数目为 2 个(8 位字长的数据,因为总的 bit 数目=10H=16D),多余的数据,MCM 将不予接收。

由于 BCNTS 寄存器是控制向 DATA 寄存器写数据的字节数目,因此这一操作必须在数据写入 DATA 寄存器之前完成。使用 2 进制代码来操作。

在大量的字节数据被写入 DATA 寄存器之后,MCM 自动地与卡片进行通信。写入 DATA 寄存器的 bits 数量由 BCNTS 寄存器制定,保持。

5. BCNTR 寄存器, 04H

BCNTR 是英文 Bit-Counter-for-Receiving 的缩写,意为接收时的位计数器(寄存器)。

寄存器名	地址	读 (READ)	写 (WRITE)
BCNTR	04H 4	N/A	BIT-COUNT-RECEIVE

与 BCNTS 寄存器相反,BCNTR 寄存器控制了 MCU 读取 DATA 寄存器的数据字节数目

例如,设置 BCNTR=20H,则可向 MCM 的 DATA 寄存器读取的数据字节数目为 4 个(8 位字长的数据,因为总的 bit 数目=20H=32D),多余的数据,MCM 将不予理会。

由于 BCNTR 寄存器是控制向 DATA 寄存器读取数据的字节数目,因此这一操作必须在读 DATA 寄存器之前完成。使用 2 进制代码来操作。

BCNTR 寄存器中的值将与实际接收到的数据字节相比较,如果有差别,则 STACON 寄存器中的 BE 标志被设置。

6. BAUDRATE 寄存器, 地址: 05H

BAUDRATE 意为数据传输到卡片上或卡片上的数据传到 MCM 时通信的位速率。虽然 BAUDRATE 寄存器的后四位相关与指定的位速率,但是实际有助于 MIFARE 1 卡的只是最后一位。

寄存器名	地址	读 (READ)	写 (WRITE)
BAUDRATE	05H 5	N/A	-- -- -- -- 1 1 1 BR

BAUDRATE 寄存器的设置将直接影响着 MCM 与 Mifare 1 卡片之间的数据通信速率。MCM 中有一个时钟发生器。写数据至 BAUDRATE 寄存器,可以控制 时钟发生器(CLOCK GENERATOR)。以下是位速率的计算公式:

$$t_{\text{bit}} = \frac{(BRX+2) \times 8}{13.56} [\mu s]$$

如果 BRX = 0x0Eh 则有: $t_{\text{bit}} = 9.44 (\mu s) = 1 / 105.94 \text{ kHz}$

上述 BRX = 0x0Eh 为对 BAUDRATE 寄存器进行设置的推荐值(对于 MIFARE 1 卡来说 BRX(大约)值为 0X0Eh,即初始化时 BAUDRATE 寄存器应设置为: "0000 1110b"=0Eh。

7. TOC 寄存器, 地址: 06H

TOC 是 Time Out Counter ((定)时间溢出计数器) 的缩写。

寄存器名	地址	读 (READ)	写 (WRITE)
TOC	06H 6	N/A	TIMEOUT-COUNTER

对 **TOC 寄存器** 的设置即对定时时间的控制。写数据至 **TOC 寄存器**，可以控制 **定时溢出计数器**。

定时溢出的计算公式如下：
$$t_{TO} = \frac{10.128}{13.56} \times TOC \text{ [us]} = 100 \times TOC \text{ [us]}$$

如果用 0x00 H 写给 **TOC**，即 **TOC = 0x00 H**，则将关闭定时溢出计数器，即 $t_{TO} = 0$ 。

TOC 寄存器 中的值 必须在 **MCM** 与 **MIFARE 1** 卡片通信时被设置；

TOC 寄存器 常用的设置值为:0AH,即定时 $t_{TO} = 100 \times 10 \text{ us} = 1 \text{ ms}$ 。

TOC 寄存器 中的值 (非零值) 将在 没有通信时，永久地被递减。因此，在没有通信时，或通信刚结束时,**TOC 寄存器** 中的值必须被设置，即设置 **TOC = 0x00H**。例如： 在 **FIFO** 中的数据有效时，就必须这样做。否则将影响 **STACON 寄存器** 的 **TE** 出错标志。

如果有溢出出现，则 **TE** 标志被设置，**DV** 标志被激活。

在完成写数据到 **DATA 寄存器**，定时溢出计数器 应该经常被初始化。但有个例外，即在认证操作(AUTHENCATION)的开始时段，定时溢出计数器 应该在存取“**KEYSTACON**”和“**KEYADDR**” **寄存器** 之后，在存取 **DATA 寄存器** 之前 被初始化。

一般地在读取 (Read) 和保存 (save) 了 **STACON 寄存器** 中的数据之后 定时溢出计数器必须被关闭。

8. MODE 寄存器， 地址: 07H

MODE 意为在与卡片数据相互往来时的**数据编码模式** (MODE of DATA coding)。

MODE 寄存器 的设置控制了 **MCM** 在与卡片数据相互通信时的**数据编码模式**。

MODE 寄存器 的每一位都与数据通信模式有关，但在使用 **MIFARE 1** 卡时，只有**最后 3 位 (BITS)** 最为关联。

寄存器名	地址	读 (READ)	写 (WRITE)
MODE	07H 7	N/A	1 1 0 0 0 P2 P1 P0

写数据至 **MODE 寄存器**，可以控制 接收器和发生器。

以下是通信时的脉冲长度(pause length)(宽度)的计算公式：
$$t_{\text{pause}} = (13+P)/6.78 \text{ [us]}$$

MODE 寄存器 中的 **P2**，**P1**，及 **P0** 位决定了在 **NPAUSE0** 和 **NPAUSE1** 引脚上的各自的脉冲宽度。脉冲宽度的可变范围在 **2 ~ 3 us** 之间。

在使用 **MIFARE** 卡时，**P2**，**P1**，及 **P0** 位应被设置为 **111 b** 或 **110 b**。

这里必须提醒读者注意的是，当我们使用 **CM200**(Philiphs 产品) 时与使用 **SB201**(UniVision Engineering Limited 联视工程有限公司的产品)时,MODE 寄存器的设置值将不一样。

使用 **CM200**(Philiphs 产品)时, **MODE 寄存器** 的设置值应为: “1100 0110b”,即 **C6H**,但在使用 **SB201**(UniVision Engineering Limited 联视工程有限公司的产品)时, **MODE 寄存器** 的设置值应为: “1101 0110b”,即 **D6H**。这是两个产品唯一存在的硬件上的不兼容,导致软件的设置值不同。 否则由于数据通信时的**数据编码模式**不相同,**MCM** 将不能正常工作,不能读/写 **Mifare 1** 卡片。在实际应用中已有此经验教训。

9. CRCDATA 寄存器, 地址: 08H

被计算 CRC 的数据必须被写入 **CRCDATA 寄存器**中。计算后的 **CRC** 必须从 **CRCDATA 寄存器**中读出。

寄存器名	地址	读 (READ)	写 (WRITE)
CRCDATA	08H 8	CRC-BYTE-READ	CRC-BYTE-WRITE

在写入一个 **BYTE**(字节)到 **CRCDATA 寄存器**后,计算将开始。计算完成后,**STACON** 寄存器的 **CV** 标志被设置。当我们要写下一个字节到 **CRCDATA 寄存器**之前,或在读取 **CRCDATA 寄存器**以得到 **CRC** 之前,或在检查 **CZ** 标志之前,都必须先读取 **CV** 标志,检查 **CV** 标志。

10. CRCSTACON 寄存器, 地址: 09H

CRCSTACON 寄存器是指 **CRC** 处理器状态和控制寄存器。

寄存器名	地址	读 (READ)	写 (WRITE)
CRCSTACON	09H 9	CV --- CZ	C8 -- -- -- CR

写数据到 **CRCSTACON 寄存器**中,即执行对 **CRC**-处理器的控制。

Bit Name 位名	Name 名称	Function 功能
C8	8-bit-CRC (8 位 CRC)	“0”---- 选择 16 位 CRC 处理器 “1”---- 选择 8 位 CRC 处理器
CRE	CRE (CRC-复位)	“0”---- 无效 “1”---- 复位 CRC 处理器

读取 **CRCSTACON 寄存器**后 用户将知道 **CRC**-处理器数据传输的状态。当标志被设置为“1”时,标志被激活。

Bit Name 位名	Name 名称	Function 功能
CR	CRC-Ready (CRC-准备)	最后一个字节 (BYTE) 被处理完成
CZ	CRC-Zero (CRC-零)	CRC-寄存器内容=00H, 意味着 CRC-校验 OK

8-bit **CRC 寄存器**值的多项式计算公式为： $X^7 + X^5 + X^4 + X^3 + 1$ 一般其初值 = 0xE3H

16-bit **CRC 寄存器**值 多项式计算公式为： $X^{15} + X^{10} + X^3 + 1$ 一般其初值 = 0x31E3 H

读取 16-bit **CRC** 模式时的 **CRC 寄存器** 需要 2 个读周期。第一个读到的字节是 16-bit **CRC** 中的低字节,第二个读到的字节是 16-bit **CRC** 中的低字节。

11. KEYDATA 寄存器, 地址: 0AH (10)

被存储在 **MCM** 中 **RAM** 的密码数据必须先被写入 **KEYDATA 寄存器**。

寄存器名	地址	读 (READ)	写 (WRITE)
KEYDATA	0AH 10	N/A	KEY-BYTE-WRITE

为了能够存取 **MCM** 内部 **RAM** 中的密码,密码的存放地址必须首先在 **KEYSTACON 寄存器**和 **KEYADDR 寄存器**两者中指定。

在做存放密码操作或做密码验证操作之前,首先必须对 **KEYSTACON 寄存器**进行设

置；在密码被准确无误地存进 RAM 之前,相关的传输密码(Transport KEY)Tkey 必须被写入 **KEYDATA-寄存器**中。传输密码 Tkey 和写入 RAM 中的密码都是 6 字节 (BYTE) 长,连续被写入 **KEYDATA 寄存器**中。但是,在“AUTHENCATION”(认证)操作时,这一**寄存器**不必使用。

12. KEYSTACON – 寄存器 地址: 0BH (11)

KEYSTACON 寄存器是指 (密码) **KEY**状态和 (密码) **KEY**控制**寄存器**。

寄存器名	地址	读 (READ)	写 (WRITE)
KEYSTACON	0BH 11		AL 0 X X X X KS1 KS0

写入 数据到 **KEYSTACON – 寄存器**进行设置,将确定存取 **RAM**中的密码 (**KEY**) 或传输密码 (**KEY**) 的密码地址的一部分。

存放在 **MCM** 的 **RAM** 中的密码对程序员来说是透明的,不得而知的。因此也是不可读的。(这里指的是密码本身及存放密码的地址不可读)。

Bit Name 位名	Name 名称	Function 功能
AL	Authenticate (认证) /Load Keys	“0” ---- 准备提取密码 “1” ---- 准备认证
KS1,KS0	Key-Set 密码集	“00” ---- 选择 RAM , KEY-set 0 “01” ---- 选择 RAM , KEY-set 1 “10” ---- 选择 RAM , KEY-set 2 “11” ---- 选择传输密码 (Transport KEY)

提取密码: **AL**=0, 表明将要提取密码;

密码认证(Authentication): **AL**=1, 表明将要认证 “AUTHENTICATION” 操作。

RAM中的密码 和**传输密码 Tkey** 二者中允许被选择。

注意:**KEYSTACON – 寄存器**中的值必须根据所使用的“AUTHENTICATION”命令(60 hex 或 61 hex 指令代码) 来确定。

此外,密码地址通过写数据到 **KEYSTACON** 和 **KEYADDR 寄存器**之后而在 **MCM** 中被确定,之后,通过写“命令”(Command)和写“地址”(address)到 **DATA 寄存器**之后,认证密码“AUTHENTICATION”操作便开始启动执行。

13. KEYADDR—寄存器, 地址: 0CH (12)

KEYADDR 寄存器将存放 **RAM** (密码) **KEY**和 传输 (密码) **KEY**各自的 密码地址的一部分。

寄存器名	地址	读 (READ)	写 (WRITE)
KEYADDR	0CH 12	N/A	AL AB A5 A4 A3 A2 A1 A0

写入 数据到 **KEYADDR– 寄存器** 将确定存取 **MCM** 的 **RAM**中的密码 (**KEY**) 或传输密码 (**KEY**) 的密码地址的一部分。

程序员不可能知道密码实际存放在 **MCM** 的 **RAM** 中的地址,否则密码便没有秘密可言了。

Bit Name	Name 名称	Function 功能
----------	---------	-------------

位名		
AL	Authenticate (认证) /Load KEYs	“0” ---- 准备提取密码 “1” ---- 准备认证
AB	KEY ‘A’ 或 KEY ‘B’	“0” ---- 使用密码 ‘A’ “1” ---- 使用密码 ‘B’
A5 ... A0	KEY address	密码地址 A5 至 A0 指定密码的 sector

提取密码: AL=0, 表明将要提取密码;

认证 **Authentication**: AL=1, 表明将要认证 “AUTHENTICATION” 操作。

14. **RCODE –寄存器, 地址: 0DH (14)**

RCODE –寄存器 用于代码接收。

寄存器名	地址	读 (READ)	写 (WRITE)
RCODE	0EH 14	N/A	X X X X 0 0 RC1 RC2

写入 数据到 **RCODE 寄存器** 对其进行设置, 将使接收器的译码器参数化。

Bit Name 位名	Name 名称	Function 功能
RC1,RC0	Receive (接收)	KOMP 引脚上用于引起中断 (“HIGH” 高电平) 的必不可少的 (脉冲) 边界数目。

RCODE –寄存器 中的值一般应设置为 **0X02 H**。然而, 在有些环境中, 设置为 **0x03 H** 时, 对 **Mifare 1** 卡片的操作距离可能会稍为好 (远) 一点。

三. **MCM 硬件初始化**

为了与 **MIFARE 1** 卡通信, 必须对 **MCM** 各硬件寄存器进行初始化设置。

寄存器	地址	初始化设置(16进制值)
STACON	1	0x0C
		0x4C
ENABLE	2	0xC0
BAUDRATE	5	0x0E
MODE	7	0xC6 或 0xD6
RCODE	13	0x02

一般地, 应该先对 **MCM** 执行 **软复位 (soft-reset)**, 然后对 **MCM** 进行初始化设置, 再进行数据通信。

在任何数据通信之前必须先执行 “REQUEST” 命令, 以和卡片建立第一步的通信联络。

以下是 **MCM** 各硬件寄存器在上电时或在软件复位时的复位表。

MCM 复位表

寄存器	地址	POR 硬件上电复位	SOR 软复位	初始化设置
-----	----	------------	---------	-------

DATA	0	XXXX XXXX	uuuu uuuu	
STACON	1	00nn 0100	0unn uu00	0x0C/0x4C
ENABLE	2	0000 nnnn	0000 nnnn	0xC0
BCNTS	3	0000 0000	uuuu uuuu	
BCNTR	4	0000 0000	uuuu uuuu	
BAUDRATE	5	Nnnn 1111	Nnnn uuuu	0x0E
TOC	6	0000 0000	uuuu uuuu	
MODE	7	0000 0000	uuuu uuuu	0xC6/0xD6
CRCDATA	8	XXXX XXXX	XXXX XXXX	
CRCSTACON	9	XXXX XXXX	XXXX XXXX	
KEYDATA	10	XXXX XXXX	uuuu uuuu	
KEYSTACON	11	00nn nnxx	00nn nnuu	
KEYADDR	12	1000 0000	1uuu uuuu	
	13	nnnn nnnn	nnnn nnnn	
RCODE	14	Nnnn 0011	Nnnn uuuu	0x02
	15	nnnn nnnn	nnnn nnnn	

说明：上表中的 POR 和 SOR 两列中： X ：表示此位不必关注；
N ：表示此位没有使用； U ：表示此位不变。

四. MCM 的硬件内核包括了如下几大部分接口电路：

1. 与 MCU(微处理机 CPU)接口电路
2. RF 模块接口电路
3. 与天线射频接口电路
4. 与电源接口电路

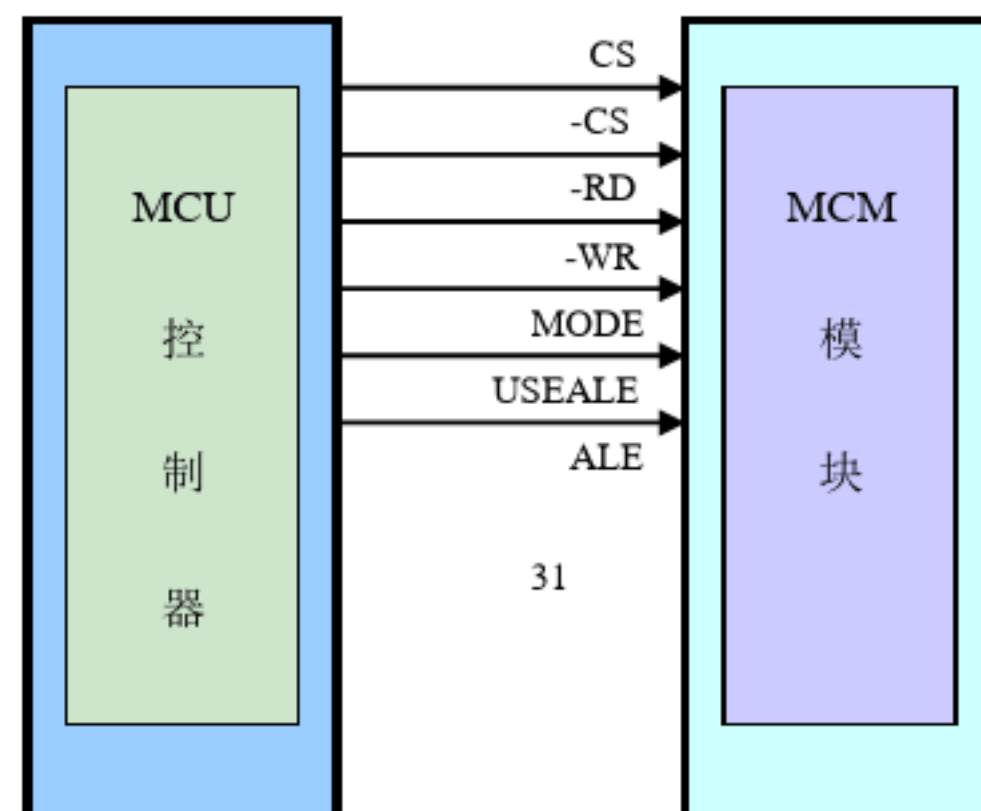
（一） MCM 与 MCU(微处理机 CPU)接口电路

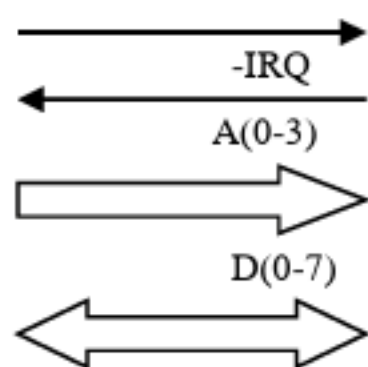
MCM 可以由标准的 MCU 接口信号来控制。这些标准的控制信号可以控制 MCM 的 ASIC 进行工作。其间使用了标准的微控制器和微处理机通信协议。

MCM 可由外部 MCU 发出特殊命令来启动运行。任何情况下，MCM 都可以由对地址的选择来启动工作。例如，MCM 的 RST 引脚，-CS 引脚和 CS 引脚的信号设置等。

对 MCM 内部存储器的存取，意味着将激活-CS 和 CS 信号，以及对地址为 00H~~0FH 的寄存器的合适的设置。

通过读特殊 I/O 地址的信息，可以得到 MCM 的状态信息。采用不同的 MCU 及其连接方法，将会有不同的硬件信号时序及数据信息流信息等。





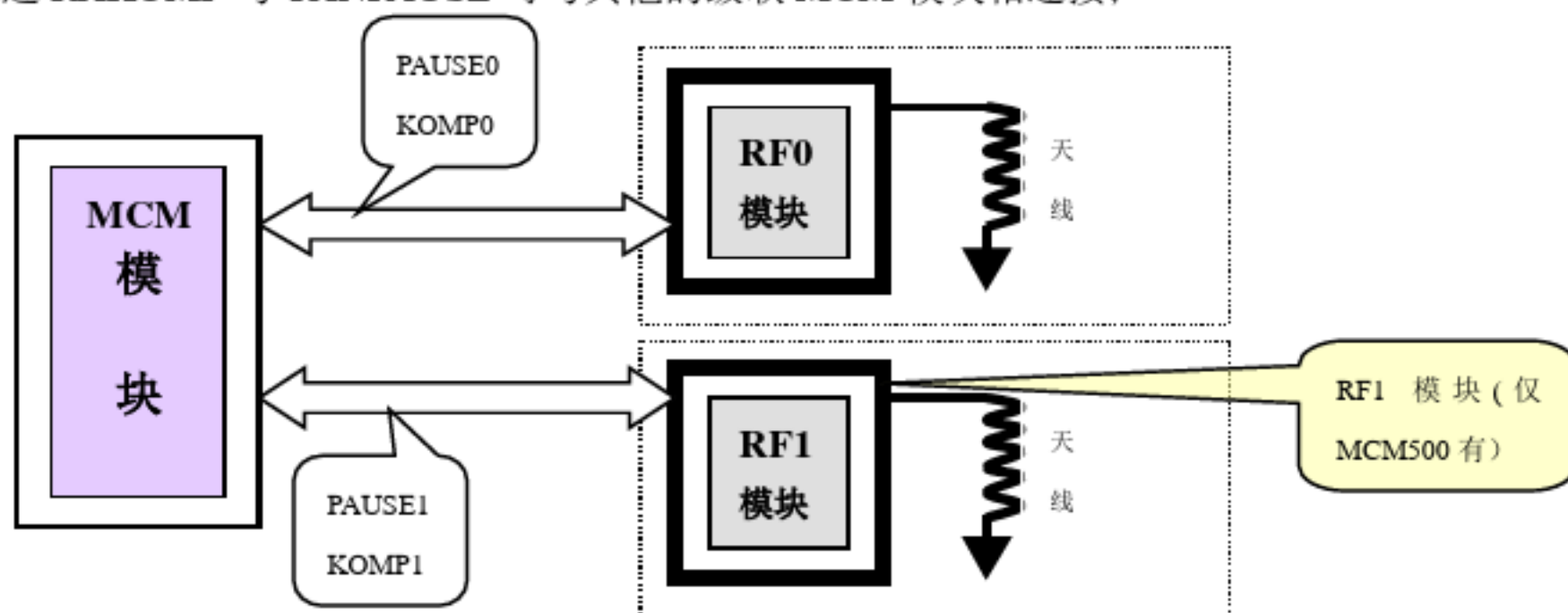
（二） RF 模块接口电路

MCM 与 RF 模块的接口一般有如下几个引脚来实现：

通过 KOMP0 与 NPAUSE0 可与 RF0 模块相连接；

通过 KOMP1 与 NPAUSE1 可与 RF1 模块相连接；（仅有 MCM500 有 RF1）

通过 RXKOMP 与 TXNPAUSE 可与其他级联 MCM 模块相连接；



（三） 与天线射频接口电路

有二个引脚 ANT 和 GND 接口可直接与正极性天线相联接。其间用同轴电缆。阻抗欧姆。连接长度一般应小于 50CM。

MCM 的外部金属屏蔽盒（仅 MCM500 有）必须连接到同轴电缆的外屏蔽包裹线，以使信号不能扩散，且免受外界电磁辐射信号等的影响。

如果天线工作不正常，可用“天线调谐器”来调谐，使之能将 Mifare 1 卡片的有效操作距离保持在 10CM/2.5CM 以上。

（四） 与电源接口电路

与电源的接口包括这样几组信号：

DVDD—数字电路正电源端（+5V）；

DGND--数字电路接地端（0V）；

BP ---WOM 缓冲器供电（通电工作时为+5V；不通电工作时为+3V）；

RFVDD—RF 电路正电源端（MCM500：+12V/MCM200：+5V）；

RFGND-- RF 电路接地端（0V）；

注意，RFGND 和 DGND 有条件的话应该分开接地，否则应接到系统的接地端，并且连接线应该具有高导电特性，且应愈短愈粗愈好。

MCM 供电电源端与 GND 端必须跨接 100NF 的电容，这些电容应尽可能靠近 MCM，以免引起接地环路。

整个系统应有单一电源供电，且应稳压，纹波电压小于 50mv。

第三章 MCM 软件编程指令

(一) . 指令集

MCM能执行有限的一些指令，并将这些指令传输到 MIFARE 1 卡片上。

每一个指令包含 7 个独立的步骤，程序员在编程执行这些指令时必须小心每一个正确的（步骤）时序，这些步骤在不同的 MCM 指令执行中而不一样。以下是 MCM 通信基本指令集。

通信基本指令集

MCU 《==》 MCM 《==》 MIFARE 1 卡片

指 令	指令代码 (hex)	相关的出错标志	接收卡片上数据
Answer to Request (Request 的应答)		TE, BE	Tagtype
Request std	26		
Request all	52		
AntiCollision (防重叠)	93	TE, BE	Serial Number
Select Tag (选卡片)	93	TE, BE, PE, CE	Size
Authentication (认证)		TE, BE, PE, CE	/
Auth_1a	60		
Auth_1b	61		
Load KEY (存取密码)	/	AE	
Read (读)	30	TE, BE, PE, CE	Data
Write (写)	A0	TE, BE	/
Increment (增值)	C1	TE, BE	/
Decrement (减值)	C0	TE, BE	/
Restore (重储)	C2	TE, BE	/
Transfer (传送)	B0	TE, BE	/
Halt (停机)	50	TE, BE	/

指令代码表

在与 MIFARE 卡片进行任何通信之前，程序员必须按照下述方法预先设置 MODE 寄存器：

MODE = “ 1 1 0 0 0 1 1 0 ” bin = C6 hex

注意，使用 CM200(Philiphs 产品)时，MODE 寄存器的设置值应为：“1100 0110b”，即 C6H，但在使用 SB201(UniVision Engineering Limited 联视工程有限公司的产品)时，MODE 寄存器的设置值应为：“1101 0110b”，即 D6H。

出错处理：

DV 标志指明了 MCM 与 MIFARE 卡片之间的传输已经完成，并且主处理机可能已经从

MCM 中收到数据。DV 标志保持“1”的持续时间，出错标志亦将保持有效。

在有效的数据被存储进 FIFO 寄存器时，DV 标志将持续保持。在向 MCM 写数据时的写指令周期内，DV 标志将被清除。DV 标志也可能被软复位（soft-reset）清除。

有时，为了确认有些操作的成功完成，必须使用定时溢出（TE）标志。

1. “Answer to Request” (“Request”应答)指令

指 令	指令代码 (hex)	相关的出错标志	接收卡片上数据
Answer to Request (Request 的应答)		TE,BE	Tagtype
Request std	26		
Request all	52		

Request 指令将通知 MCM 在天线有效的工作范围（距离）内寻找 MIFARE 1 卡片。如果有 MIFARE 1 卡片存在，这一指令将分别与 MIFARE 1 进行通信，读取 MIFARE 1 卡片上的卡片类型号 TAGTYPE（2 个字节），由 MCM 传递给 MCU，进行识别处理。

程序员可以根据 TAGTYPE 来区别卡片的不同类型。

对于 MIFARE 1 卡片来说，返回卡片的 TAGTYPE（2 个字节）可能为 0004h。

从一个指定的卡片开始，后续的卡片可以根据 TAGTYPE 而被选择。当使用“Request std”指令来寻卡时，只有那些卡片上没有设置成“HALT_MODE”（停机模式）的卡片将响应这一指令。

Request all 指令的使用是很重要的，它可以防止 MCM 选择同一卡片好几次。当某一张卡片在 MCM 之天线的有效的工作范围（距离）内，Request all 指令在成功地读取这一张卡片之后，将一直等待卡片的使用者拿走这一张卡片，直到有新一张的卡片进入 MCM 之天线的有效的工作范围（距离）内。当然，这里的“新一张的卡片”亦可以是刚刚拿开的那张卡片。

Request all 指令是非连续性的读卡指令。只读一次。但有个例外，当某一次 Request all 指令读卡片失败时，例如，卡片没能通过密码认证或其他原因而出错时，Request all 指令将连续地读卡，直到读卡成功才进入非连续性的读卡模式。

Request all 指令适用于那些需要有人工干预的场合。

Request std 指令的使用和 Request all 指令刚巧相反，Request std 指令是连续性的读卡指令。当某一张卡片在 MCM 之天线的有效的工作范围（距离）内，Request std 指令在成功地读取这一张卡片之后，进入 MCM 对卡片的其他操作。如果其他操作完成之后，程序员又将 MCM 进入 Request std 指令操作，则 Request std 指令将连续性地再次进行读卡操作，而不管这张卡片是否被拿走。只要有一张卡片进入 MCM 之天线的有效的工作范围（距离）内，Request std 指令将始终连续性地再次进行读卡操作。

Request std 指令是连续性的读卡指令。

Request std 指令适用于那些不需要有人工干预的场合，即全自动的场合。例如，宾馆，酒店，高级写字楼等场所的门禁控制系统（Door Access Control），高速公路，停车场等的不停车收费系统（Non-Stop Road Tolling），等等。

8. “AntiCollision” 防重叠

指 令	指令代码 (hex)	相关的出错标志	接收卡片上数据
AntiCollision (防重叠)	93	TE, BE	Serial Number

如果有多于一张的 Mifare 1 卡片在 MCM 之天线的有效的工作范围（距离）内，必须

使用 **AntiCollision** 指令，使 **MCM** 能够在这一叠 **Mifare 1** 卡片中选择个别的一张卡片。

AntiCollision 指令开始于一个“**AntiCollision loop**”(防重叠循环)。结束时，**AntiCollision** 指令将提供给用户在这些卡片叠中选择的那张个别的卡片的一个有效的 **40 bit 长的序列号 SN** (**serial number**)。

一般地，**AntiCollision** 指令将有序地读所有的处在 **MCM** 之天线的有效的工作范围（距离）内的 **Mifare 1** 卡片。读完第 **Xi** 张卡片后，**MCM** 及 **MCU** 对这 **Xi** 张卡片进行处理，完毕之后，读第 **Xi+1** 张卡片，然后 **MCM** 及 **MCU** 对这 **Xi+1** 张卡片进行处理。循序渐进，直至所有的卡片。

AntiCollision 指令的启动必须是在程序员完成了对 **STACON** 寄存器中的 **AC** 位的成功的设置之后。

注意：**AntiCollision** 指令事实上并不实际地选择一张个别卡片，而仅仅是读取 **Mifare1** 卡片上的序列号 **SN**。**MCM** 与卡片的真正联络，选择某一张卡片，是由程序员向 **MCM** 发送“**SELECT**”命令来完成的。

AntiCollision 指令读取的 **Mifare 1** 卡片上的一个有效的 **40 bit 长的序列号 SN**(**serial number**) 被存储在 **Mifare 1** 卡片上的第 **00H** 扇区中的第 **00h** 块 (**Block**) 中。共 5 个字节。

事实上，对于程序员来说，实际有意义的 **SN** 只有前 4 个字节，最后一个字节只是一个 **SN** 的校验码。

程序员在控制 **MCM** 及 **MCU**，发送 **AntiCollision** 指令，而获得 **SN** 之后，一般地应在程序中对所接收到的 **SN** 进行校验，以确保数据的正确性。具体的方法是对所接收到的 **SN** 的 **bit** 位进行异或校验，这在后面的具体程序编制中将会阐述。

对于 **MIFARE 1** 卡片来说，返回某一张卡片的有效**序列号 SN**（4 个字节）可能为：**007e0a42h**。

3. “**Select Tag**” 选择卡片操作

指 令	指令代码 (hex)	相关的出错标志	接收卡片上数据
Select Tag (选卡片)	93	TE, BE, PE, CE	Size

在一个成功的 **AntiCollision** 指令之后，或在任何时候当程序员想实际地与已知序列号的卡片进行通信时，必须使用 **Select** 指令，以建立与所选卡的通信。

为了允许在 **Select** 指令以后对卡片能进行 **Read/Write** 等指令的操作，**Select** 指令是很重要的，必须首先被使用。被选择的卡片将给出其自己的存储器容量---已编码的一个 **BYTE**（字节）。

Select 指令成功地完成后，**MCU** 将得到 **MCM** 之 **DATA** 寄存器传送来的一个字节长的卡片容量信息---**SIZE** 字节。

SIZE 字节被存储在 **Mifare 1** 卡片上的第 **00H** 扇区中的第 **00h** 块 (**Block**) 中。共 1 个字节（8bits）。

对于 **MIFARE 1** 卡片来说，返回卡片的 **SIZE**（个字节）可能为 **88h**。

4. “**Authentication**” 认证操作

指 令	指令代码 (hex)	相关的出错标志	接收卡片上数据
Authentication (认证)		TE, BE, PE, CE	/
Auth_1a	60		
Auth_1b	61		

在 MCU 将运算获得的数据，准备存储到卡片上的存储器之前，或 MCU 希望能读取 Mifare 1 卡片上的数据之前，程序员必须证明他的读/写请求操作是被允许的。

这可以通过选择秘密地存储在 MCM 之 RAM 中的密码集 (KEYSET) 中的一组密码来进行认证而实现。如果这一组密码匹配与 Mifare 1 卡片上的密码，这一次的操作被允许进行。

卡片上的存储器的每一个 block (块) (128 bits) 都分别地指定了该 BLOCK (块) 的存取条件。这些存取条件是根据密码 A 或 B (它们对整个 sectors (扇区) 始终有效) 而定。这样一来，不同的操作可能会被允许对一个 sector 中的 4 个不同 blocks 的每一个 block 进行操作。

MCM 能够存储 3 个密码集 KEYSET0, KEYSET1, KEYSET2。每一个 KEYSET 又包含了 KEY A 及 KEY B 等，以存取最高达 32Kbit 内存容量的 MIFARE 卡片。

用户必须在 KEYSTACON (0B H 地址) 寄存器中指定一套密码，即设置 KS0, KS1。

KS1KS0 = 00	选择 KEYSET0
KS1KS0 = 01	选择 KEYSET1
KS1KS0 = 10	选择 KEYSET2
KS1KS0 = 11	选择 KEYSET3

KEYADDR 寄存器 (0C H 地址) 中的 AB 位用于选择 KEY A (当 AB=“1” 时) 和 KEY B (当 AB=“0” 时)。

AB = “1”	选择 KEY A
AB = “0”	选择 KEY B

KEYADDR 寄存器(地址 0C H) 中的“AB”设置必须匹配“Authentication”命令 (指令代码 60 和 61 hex)，这是很重要的。否则，“Authentication”命令将失败。

即，在“Authentication”命令中，60h 代码用于认证 KEY A；61h 代码用于认证 KEY B。

当用 60h 代码时，在 KEYADDR 寄存器中只能设置 AB=“1”，用于选择 KEY A；

当用 61h 代码时，在 KEYADDR 寄存器中只能设置 AB=“0”，用于选择 KEY B；

当用 60h 代码时，在 KEYADDR 寄存器中不能设置 AB=“0”，否则认证命令将出错；

当用 61h 代码时，在 KEYADDR 寄存器中不能设置 AB=“1”，否则认证命令将出错；

KS1 和 KS0 的设置选择与上述的认证命令的指令代码 60 和 61 hex 的选择无关；

KS1 和 KS0 的设置选择与上述的 KEYADDR 寄存器中 AB 的设置无关。

在 MCM 中，KEY-RAM (密码存储器) 的实际存储器范围是：

3	x	sector	x	# KEYS/sector	x	# bits/KEY	
3	x	64	x	2	x	48	= 18,432 bits
							= 2,304 bytes

注意：KEY-RAM (密码存储器) 中的数据不能读出芯片之外。

MCM 中另外还包含了 KEY-ROM (密码只读存储器)，用以存储一套传输密码(one set of transport keys) 和一个维护密码 (one service KEY)。传输密码对于用户想在 KEY-RAM (密码存储器) 中自己定义密码而编程序时，很有用。

KEY-ROM (密码只读存储器) 的存储器容量为：

# sectors	x	# bits / KEY	+	# bits of service KEY	
64	x	48	+	48	= 3,120 bits
					= 390 BYTES

注意：KEY-ROM (密码只读存储器) 是掩膜方式编程的，不能读出芯片之外。

5. “Load KEY” 存取密码

指 令	指令代码 (hex)	相关的出错标志	接收卡片上数据
Load KEY (存取密码)	/	AE	

在 MCM 中，每一个认证扇区 (Authentication sector) 中包含了一套传输密码 Tkey (one transport KEY)。这些密码在制造产品时，可以编程。它们可以被传输到系统的集合器 (例如，用户开发的硬件系统) 中，集合器负责存取新的用户定义的密码到 MCM-WOM (密码只读存储器) 中。此时，可以通过 “Load KEY” 指令来完成任务。

对于一个单独扇区 (sector) 中的密码提取，则相应的传输密码必须被预先指定。

在用 “Load KEY” 指令来完成存取密码过程中，KEYSTCON 寄存器和 KEYADDR 寄存器的设置与 “Authentication” 的操作相似。但也有些区别。这将在后面的编程中具体介绍。

6. “Read” 读指令操作

指 令	指令代码 (hex)	相关的出错标志	接收卡片上数据
Read (读)	30	TE, BE, PE, CE	Data

Read (读) 指令允许 MCU 通过 MCM 来读取 MIFARE 1 卡片上完整的 16 个 Bytes 的数据块 (Data blocks)。

只有在预先 “Authentication” 认证指令完成后，才允许进行对 Mifare 1 卡片上的某一数据扇区进行 Read (读) 指令操作。

Read (读) 指令操作只能一个块 (Block) 一个块地读，即只能 16 个字节一次性地读取。如果只要求某 Block 中的几个字节的数据，也只能一个整块 16 个字节一起读取，由程序员选取指定的字节。

从卡片上读到的数据必须由 MCU 进行校验，以确保数据的有效性。

密码数据不能被读取。

7. “Write” 写指令操作

指 令	指令代码 (hex)	相关的出错标志	接收卡片上数据
Write (写)	A0	TE, BE	/

“Write” 写指令允许用户写数据到 MIFARE 卡片上 (完整的 16 个 BYTES 的数据块 (Data blocks))。只有在先 “Authentication” 认证指令完成后，才允许进行对要求的数据扇区或数据块 (Block) 进行 “Write” 写指令操作。

为了提供最大的数据集成度，以及包含大量密码值的密码数据块的保密性，我们必须保持一个大数据结构。这样可以允许 MIFARE 卡片执行增值/减值 (Increment/Decrement) 指令。

“value block” (数据块) 的数据结构：

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
0															
<u>address</u>	address	<u>address</u>	address	V	A	L	UE	<u>V</u>	<u>A</u>	<u>L</u>	<u>UE</u>	V	A	L	UE

表：数据块

数据块通过一个写操作，将存储的数据在每一个 block 块中写 3 次，1 次反写，从而完成数据块的初始化。此外，一个地址引导位代码域必须写 4 次，其中 2 次为反向写入。正/负数据值将以标准的 2 的补码格式来表示。

注意：在将数据写到卡片上的某一扇区时，一定要小心。因为有些 block 中存储了密码数据

以及存储允许使能数据。特别是每一个扇区的 Block3 中存放了该扇区的存取条件，包含有 KEYA，KEYB 及该扇区的控制字。Mifare 1 卡片出厂时的 Block3 有缺省值，为：“a0a1a2a3a4a5ff078069b0b1b2b3b4b5”，共 16 个 Bytes。

涉及 Mifare 1 卡片的存储结构等信息，请参考本文的关于 Mifare 1 卡片章节。

程序员在使用 Mifare 1 卡片做应用时，一定要清清楚楚记住每一个扇区的 Block3 的数据，这样也就记住了扇区的密码和存取控制字。否则，扇区的存储空间将不能被 READ/WRITE 等操作而失效。

任何人试图用任何方式来读写不知密码的卡片或某一扇区都是徒劳无益的。

卡片应放在安全的地方，即不要放在离 MCM 天线较近的地方。因为当 MCM 在执行某些指令时，有可能无意对这一卡片进行了读/写等操作。

8. “Increment “ & “ Decrement “ & “ Restore” 增值/减值/重储

指 令	指令代码 (hex)	相关的出错标志	接收卡片上数据
Increment (增值)	C1	TE, BE	/
Decrement (减值)	C0	TE, BE	/
Restore (重储)	C2	TE, BE	/

通过 Mifare 1 内部电路，MIFARE 1 卡片能够执行“Increment “ & “ Decrement “增值/减值 操作，如果：

- 用合适的写指令对数据块 “value block”进行了初始化；
- 根据 KEY A 和 KEY B ，允许进行“Increment “ & “ Decrement “增值/减值 操作；
- “Increment “ 增值 0 操作和“ Decrement “减值 0 操作是不允许的；

“Increment “ 增值：加指定的值到卡片的存储器中

“ Decrement “减值：从卡片的存储器中减去指定的值

“ Restore” 重储： 执行一个 “Decrement 0”（减 0）指令

上述指令的计算结果将被存储在 MCM 的一个内部的数据缓冲寄存器中，直达以后的指令来存取。为了将结果写到卡片上，紧接着必须执行 “Transfer”指令。Transfer”指令并不改变数据缓冲寄存器中的数据值。这一操作保持数据块中的数据结构，直到数据块中的值被自动地反写及校验为止。存储的地址也被正确地传送存储到数据块中。

数据块的长度为 4 字节（BYTES）。

万一数据值溢出，即数据值远大于最大的正数值 或 远小于最小的负数值时，MIFARE 卡片将停止操作，并返回一个 NACK 代码。

9. “Transfer” 传送指令

指 令	指令代码 (hex)	相关的出错标志	接收卡片上数据
Transfer (传送)	B0	TE, BE	/

每一个“Increment “ & “ Decrement “增值/减值 操作都必须跟随一条 “Transfer” 传送指令，这样真正地将数据结果传送到卡片上去。否则，没有传送指令，数据结果仍保持在数据缓冲寄存器（value buffer register）。“Transfer” (传送) 指令的目标地址可能会相同与 “Increment “ & “ Decrement “增值/减值指令时的源地址。然而，如果有了完善的后备管理软件，这样的事可能不会发生。

除了 “Transfer”指令之外，其他所有的指令将改变内部数据缓冲寄存器（value buffer register）的内容。这将要求使用 “Restore”指令。“Restore”指令将重建 “Read “指令之后的数据缓冲寄存器（value buffer register）的内容。

如果 “Transfer” (传送) 指令存取了一个新的存储器位置时，将被要求一个相同于源

地址的存储条件的 目标地址。

即，对于两个内存位置，要求执行“**Increment** “ & “ **Decrement** “增值/减值指令。
“**Transfer**”(传送) 指令自动地保持正确的数据结构，包括存储数据的地址。

10. “Halt” 停机指令

指 令	指令代码 (hex)	相关的出错标志	接收卡片上数据
Halt (停机)	50	TE, BE	/

“Halt” 停机指令将 MIFARE 1 卡片设置为 “**HALT MODE**”。例如，卡片已经退出使用等。卡片将保持 “**HALT MODE**” 状态，直到被复位（例如：重新用于通信 等）。

第四章 MCM 应用开发

在这一章节里，将向广大 读者介绍 MCM 的具体的应用开发。我们选用的卡片为 Mifare 1 S50 系列的标准的符合 ISO/IEC 14443 TYPE A 标准的非接触式 IC 射频卡。选用的读写器的核心模块为 MCM200（或是 SB201）。

根据以上几个章节的介绍，相信读者可以理解如下的通用读写器 DEMO 电路的设计。

一. DEMO 硬件电路的应用开发

通用读写器 DEMO 的硬件电路请见所附的图纸。具体说明如下：

1. DEMO 电路中，对于 MCU 的说明：

笔者选用了市场上容易购置的美国 ATMEL 公司的 89C52 芯片。其内建 8K EEPROM，256 bytes 的 RAM ，内设 P0，P1，P2，P3 等四个端口，其中 P0，P2 为数据/地址双向的多用端口。内有 3 个定时器，T0，T1 和 T2。其振荡晶体选用 11.0592MHz，以利于以后通信时的波特率的精确设计和设置。89C52 中还内设 8 级中断控制系统，3 级单向一次性可编程的密码内存，可以防止芯片内的程序被非法读写，拷贝等，保护知识产权。等等。

AT89C52 的软件指令完全兼容与 MCS-51 系列的 MCU 的标准，包括指令的寻址方式，各种数据的操作等等。

有关详细的资料，可以查阅 ATMEL 公司的网页：[HTTP: //WWW.ATMEL.COM](http://www.atmel.com)

2. Demo 电路中，对 MCM 的说明：

在 DEMO 电路中，笔者选用了 Philips 公司的 MCM200 模块（也可用联视工程公司的 SB201 模块）。

将 MCM 的 D0 ~ D7 连接到 MCU 的 P0 端口；MCM 的读写线连接到 MCU 的 -RD 和 -WR 端上； A0 ~ A7 空缺，使用 ALE 线，连接到 MCU 的第 30 脚上；MCM 的天线端点上（ANT，-ANT 等）必须对地接高频滤波电容，并串接高频电感；MCM 的第 9 脚必须接 3.6V 的后备锂电池；MCM 的第 31 脚及 12 脚分别接 MCU 的 25 脚和 26 脚，由程序员来控制 MCM 的启动和关闭。MCM 上的模拟电路供电和数字电路的供电端必须跨接高低频滤波电容等。

3. Demo 电路中，对显示电路的说明：

在 DEMO 电路中，笔者选用了 Motorola 公司的 MC14499 显示模块。

每个 MC14499 模块可动态扫描 4 位段式 LED（共阴）。DEMO 电路中共使用了 2 个 MC14499 模块，以显示 8 位数据。

MC14499 显示 0 ~9 数字时很直观，但是它是数字型的。由于我们经常要显示 16 进制的数据，因此对于 A, B, C, D, E, F 的显示，MC14499 的显示可能不很直观，希望读者注意。

在新版本的 DEMO 电路中，笔者已经将显示电路改为 LCD 点阵式，多字行的液晶显示电路。这对于将这种低功耗的显示电路直接应用于所要求开发的应用设备中将提供很大方便。

两个 MC14499 模块的 CLK 和 DATA 信号将分时复用，以节省 MCU 有限的资源，分别接到 MCU 的 21 和 22 引脚上，两个 MC14499 模块的--ENB 信号分别接到 MCU 的 23 和 24 引脚上。如果选通-ENB1，则第一个 MC14499 显示模块将接收数据；如果选通-ENB2，则第二个 MC14499 显示模块将接收数据。

4. Demo 电路中，对键盘电路的说明：

在 DEMO 电路中，笔者选用了 4 X 4 方阵式样的键盘电路与 MCU 相连在 P1 端口上。

键盘电路中的 4X4 共 16 个键分别设置为 0~9 十个键，一个“.”键，以及其他一些功能键及字母键。

键盘电路的扫描值通过一个 LS244，送到 MCU 的 P1 的高四位，供 MCU 采集；MCU 的 P1 的低四位送出键盘的扫描信号。LS244 的选通（MCU 采集 P1 的高四位时）信号和显示电路的 DATA 线分时复用，以节省 MCU 有限的资源。

本键盘电路的软件设计不同于传统，而采用了比较直接，明了，快速的“位扫描法”，这对于读者如果要求对键盘数目进行扩充，则读者在仔细阅读本文之后定能迎刃而解。

在软件设计说明章节中将详细介绍键盘扫描电路软件设计方法。

5. Demo 电路中，对指示信号的说明：

LED1：指示 MCM 的-CS 信号是否被选中，LED1 亮，则指示 MCM 已被 MCU 选中，可对 MCM 进行操作。（此时，应注意 MCM 的 RST 端指示信号 LED2 应亮，否则 MCM 不能被操作）。

LED2：指示 MCM 的 RST 信号是否被选中，LED2 亮，则指示 MCM 已被 MCU 选中，可对 MCM 进行操作。（此时，应注意 MCM 的-CS 端指示信号 LED1 应亮，否则 MCM 不能被操作）。

LED1 和 LED2 的指示应该同时有效才能使 MCM 工作。

LED3：整个 DEMO 电路的“OK”指示信号。即在每次对 MCM 操作成功之后，LED3 将指示为亮。例如在对 MCM 成功地进行了 SELECT 命令或 REQUEST 命令等操作之后，LED3 将指示为亮。特别地，在本文的“门禁控制系统”应用程序中，LED3 将直接作为门锁的驱动信号。在实际的装置中，如果“门禁控制系统”的读写器读到某一张写有该“门禁控制系统”密码的非接触式射频卡，则 LED3 将指示发亮，并驱动门锁，打开大门。（读者如果已完全理解，读通“门禁控制系统”应用程序的话，完全可在程序中适当的位置加入一段小程序，连同 LED3 的驱动信号，而作为门锁的复合驱动信号。这样的“门禁控制系统”将更具有安全性，可靠性。）

喇叭（蜂鸣器）：DEMO 电路的喇叭（蜂鸣器）指示音信号，在每次对 MCM 操作成功之后，喇叭（蜂鸣器）将发声。例如在对 MCM 成功地进行了 SELECT 命令或 REQUEST 命令等操作之后，喇叭（蜂鸣器）将发声。特别地，在本文的“门禁控制系统”应用程序中，喇叭（蜂鸣器）将直接作为门锁的驱动指示音信号。配合 LED3 的指示信号，仅当门锁被合法地打开时，喇叭（蜂鸣器）将给出指示音信号。否则，喇叭（蜂鸣器）将不会给出指示音

信号。

读者如果已完全理解，读通“门禁控制系统”应用程序的话，完全可在程序中适当的位置对 LED3 或喇叭（蜂鸣器）进行任意的程序设置。

电源输入保护电路：在 DEMO 电路中，加入了简单的电源保护电路。即在电源的输入端加入一个反向二极管，当电源极性接反时，将输入的电源短路，而不会使 DEMO 电路损坏。

6. 其他电路

通信电路：在新版本的 DEMO 电路中，笔者已经将 RS232 的通信电路连接上。以利于用上位 PC 来具体地控制读写器，并能编制用户界面良好的应用软件，等等。

为了方便读者对于实际应用中不同的通信方式的要求，例如要求长线（长距离）方式（例如 RS422/485 等），笔者业已设计完成相应的“全双工 RS232 《==》 RS422/485 通信（双向）转接设备”。该转接设备非常轻巧，低功耗，成本很低，非常适合于各种不同应用类型的计算机与应用设备之间的转接。例如大楼中的门禁系统中各读写设备与上位计算机之间的长线方式的通信就应采用这种转接设备。其长线驱动可达 1200 米以上，且稳定可靠。传统的 RS232 通信至多不超过 20 米。（注：关于“全双工 RS232 《==》 RS422/485 通信（双向）转接设备”的介绍，笔者将会在另外文章中再做详细介绍。）

LCD 显示电路：轻巧，低功耗的非接触式卡片读写器已越来越受到欢迎。因此大电流显示的 LED 将被 LCD 显示所取代。笔者在新版本的 DEMO 电路中，已经将 LCD 应用于读写设备中，且为图形方式点阵大屏幕显示。等等。

二. DEMO 电路的软件应用开发

DEMO 电路的软件应用开发的目标是：主要根据所设计的 DEMO 硬件电路而进行第一步，最低层的面向微处理机 MCU 的应用程序的开发。

由于在 DEMO 电路中笔者选用的 MCU 为完全兼容与 MCS-51 系列微处理机的 ATMEL 公司的 AT89C52，因此，笔者将重点介绍 MCS-51 系列的面向硬件操作的最低层的汇编语言的 DEMO 电路的应用程序的开发。

应用程序的开发将主要分为两大部分，即对 MCM 的应用程序的开发及对读写器的其他电路的应用程序的开发。

程序开发的方式将主要建立一系列的汇编语言函数子程序以供主控程序的随时调用。即对 MCM 模块或读写器的其他电路分别编制汇编语言函数子程序。

（一）. 对 MCM200（SB201）操作指令的程序设计和开发

对 MCM 指令的说明：

MCU 对 MCM 的控制是以 MCU 发出 MCM 的指令来达到的。MCU 对 MCM 的控制事实上也就是对 Mifare 1 非接触式 IC 射频卡的控制。

MCM 的指令主要有这样一些：Request std, Request all, Anticollision, Select, Load_key, Authentication(Auth_1a,Auth_1b), Read, Write, Increment, Decrement, Restore, Transfer, Halt 等等。其中比较重要的是前 8 条指令，以及 Halt 停机指令等。因为它们可以完成 MCU 对 Mifare 1 非接触式 IC 射频卡的很多应用场合的控制。例如 门禁控制系统（Door Access System），高速公路不停车收费系统，停车场收费管理系统，公交/地铁非接触式 IC 智能射频卡的“刷卡”读写器（收费管理系统），机关/企业内部考勤管理系统，医疗保险管理系统，出租车收费管理系统，银行服务“一卡通”等等。

MCU 对 MCM 的某一指令操作不是简单的一条指令所能完成的，必须有一个程序的序列来完成，其中有对 MCM 硬件内核寄存器的设置以及汇编语言软件上的判断和设置。例如对卡片进行读（READ）操作，则程序员必须对 MCM 内部的 BCNTR 寄存器，BCNTS 寄存器，TOC 寄存器，STACON 寄存器等设置，同时还必须对地址进行设置，对每一个状态进行判别等等，对最终读得的数据还必须进行校验等等。就连一条最简单的停机（Halt）指令也必须首先对 MCM 内部的诸多寄存器进行设置，等等。不同的指令将设置不同的 MCM 内部寄存器以及应有不同的汇编语言程序序列。

1. 对 MCM200（SB201）“Request std”和“Request all”指令的汇编语言应用程序的设计

功能：调用该汇编语言子程序，将执行 MCM “Request std”或“Request all”指令操作，读取 Mifare 1 非接触式 IC 射频卡的 TAGTYPE 字节，建立与卡片的第一步必不可少的通信联系。

程序说明：

1. 在以下的汇编语言程序中,如果要调用该 MCM 指令操作，则应首先设置 MCU 的 R2 的值：

R2="00H" 执行 Request std 操作；

R2="01H" 执行 Request all 操作；

在该指令程序中用到了一些其他的子程序分别为：

D500US: 为 500us 的延时子程序；

H_SEND_TO_BUF01:

DEMO 电路中,将 A 寄存器中的数据送显示电路中的高四位中的第 0, 1 位,以准备显示,地址为 34H,35H;

H_SEND_TO_BUF23:

DEMO 电路中,将 A 寄存器中的数据送显示电路中的高四位中的第 2, 3 位,以准备显示,地址为 32H,33H;

SEND_TO_BUF01:

DEMO 电路中,将 A 寄存器中的数据送显示电路中的低四位中的第 0, 1 位,以准备显示,地址为 39H,3AH;

SEND_TO_BUF23:

DEMO 电路中，将 A 寄存器中的数据送显示电路中的低四位中的第 2, 3 位,以准备显示,地址为 37H,38H;

DISPLAY:

DEMO 电路中，显示 8 位数据中的低 4 位数据值。即将存储在 37H 至 3AH 显示内存中的低四位数据显示在 LED 上。

H_DISPLAY:

DEMO 电路中，显示 8 位数据中的高 4 位数据值。即将存储在 31H 至 34H 显示内存中的高四位数据显示在 LED 上。

等等。

2. 在以下的汇编语言程序中，读者可能会碰到一些寄存器变量，或其他应用子程序，这些可能是用于主控程序或是调试而用。特加以说明。

程序清单如下：

```
;===== MCM REQUEST OPERATION =====  
request:
```



```

REPEAT_RQT:
_1_STACON:                                ;设置 MCM 中的 STACON 寄存器为 : 0CH
    MOV A, #0CH                           ;设置 A 寄存器 = 0CH
    MOV R0, #01H                          ;STACON 寄存器地址为 01H
    MOVX @R0, A                           ;将 A 寄存器内容送入 STACON 寄存器中
_2_BAUDRATE:                              ;设置 MCM 中的 BAUDRATE 寄存器为:0EH
    MOV A, #0EH
    MOV R0, #05H
    MOVX @R0, A
_3_ENABLE:                               ;设置 MCM 中的 ENABLE 寄存器为:0C0H
    MOV A, #0C0H
    MOV R0, #02H
    MOVX @R0, A
_4_MODE:                                 ;设置 MCM 中的 MODE 寄存器为:0C6H
    MOV A, #0D6H                           ;(如果采用 SB201, 则 MODE 寄存器应为:0D6H)
    MOV R0, #07H
    MOVX @R0, A
_5_STACON_AGAIN:
    MOV A, #0CH
    MOV R0, #01H
    MOVX @R0, A
_6_RCODE:
    MOV A, #02H
    MOV R0, #0EH
    MOVX @R0, A
_7_BCNTS:
    MOV A, #07H
    MOV R0, #03H
    MOVX @R0, A
_8_BCNTR:
    MOV A, #10H
    MOV R0, #04H
    MOVX @R0, A
;-----
JUDG_RQT_STD_OR_ALL:                     ; 根据 R2 值, 判断是执行 Request std 操作
    MOVA,R2                               ; 还是 Request all 操作
    XRL A, #01H
    JNZ RQT_STD
;-----
RQT_ALL:
    MOV A, #52H
    AJMP _11_RQT_MCM
RQT_STD:
    MOV A, #26H

```

```

_11_RQT_MCM:
    MOV R0, #00H
    MOVX @R0, A
_12_TOC:
    MOV A, #0AH
    MOV R0, #06H
    MOVX @R0, A          ;TOC = 0AH
;-----
RD_STACON:
    MOV R0, #01H
    MOVX A, @R0          ;READ STACON()
;=====
JUDG_DV_BIT:
    JNB ACC. 7, RD_STACON ; RD_STACON
;=====
    mov r7, a            ;protect A = stacon()
;=====
    MOV A, #00H
    MOV R0, #06H
    MOVX @R0, A          ;TOC=00H
;=====
    mov a, r7            ;return stacon() value to A
;=====                ;for "BE" and "TE" error-flag
    ACALL H_SEND_TO_BUF23 ;DISPLAY R7XX ---> R7 = STACON()
;=====
_13_JUDG_ERR:
    JB ACC. 6, TE_ERR    ; TE_ERR
    JB ACC. 3, BE_ERR    ; BE_ERR
;=====
    MOV R0, #00H        ;READ_TAGTYPE_0
    MOVX A, @R0
;
    MOV 45H, A
    ACALL SEND_TO_BUF01 ;SEND TAGTYPE 0 TO DISP_BUF_LOW
;-----
    mov r0, #00h
    MOVX A, @R0          ;READ_TAGTYPE_1
;
    MOV 46H, A
    ACALL SEND_TO_BUF23 ;SEND TAGETYPE 1 TO DISP_BUF_HIGH
;-----
    MOV B, #00H          ;"00H" is "OK" flag
;=====
    mov a, b

```

```

        ACALL H_SEND_TO_BUF01      ;XXBB
;=====
        SETB P2.6                  ;"OK" LAMP NOW WORKING
        mov a, #00h
        ACALL SPK_ON               ;if "ok" then spk on
        LJMP request_exit          ;exit and RET
;=====
TE_ERR: MOV B, #01                 ;"TE" error flag is "01H"
        MOV R6, B

        LJMP RQT_EXIT
;=====
BE_ERR: MOV R7, #0AH              ;READY TO DELAY 500us
        ACALL D500US               ;延时 500us
        MOV B, #0BH                ;"BE" error flag is "0BH"

        ; MOV A, #88H
        ; ACALL SEND_TO_BUF01
        ; ACALL SEND_TO_BUF23
;=====
RQT_EXIT:
        LJMP REPEAT_RQT
;=====
request_exit:
        RET
;=====End of MCM Request Operation =====

```

2. 对 MCM200 (SB201) “AntiCollision” (防卡片重叠) 指令的汇编语言应用程序的设计

功能: 调用该汇编语言子程序, 将执行 MCM “AntiCollision” (防卡片重叠) 指令操作。如果卡片读写器的天线之有效工作距离内有多张重叠的 Mifare 1 非接触式 IC 射频卡, 则该指令将帮助程序员选择其中指定的一张卡片进行操作。“AntiCollision” (防卡片重叠) 指令操作将读取卡片的系列号 (SERIAL NUMBER) 字节。

程序说明:

1.) 在以下的汇编语言程序中, 用到了一些其他的子程序 (和上述的 “Request std 和 Request all” 指令操作中的相同), 分别为:

D500US: 为 500us 的延时子程序;

H_SEND_TO_BUF01: (同上)

H_SEND_TO_BUF23: (同上)

SEND_TO_BUF01: (同上)

SEND_TO_BUF23: (同上)

DISPLAY: (同上)

H_DISPLAY: (同上)

2.) 在以下的汇编语言程序中, 读者可能会碰到一些寄存器变量, 或其他应用子程序, 这些可能是用于主控程序或是调试而用。特加以说明。等等。

3.) 在本汇编语言子程序中, 校验, 计算卡片序列号的方法采用了相邻两个字节相互异或, 得出的结果与下一个字节再异或这样一种方法。前四个字节异或的结果与第 5 个字节应该相同, 即与第 5 个字节异或的结果应为“00H”, 否则 MCU 读到的卡片的序列号有错。因此对于我们来说, 卡片的序列号前四个字节是有意义的, 第 5 个字节仅仅用于校验之用。并无实际意义。我们通常所说的序列号指的是前 4 个字节。并不包括第 5 个字节。这五个字节被存储在卡片的第 0 扇区的第 0 块中, 由卡片的生产商制定, 固化, 不得更改, 且在市面上流通的 Mifare 1 S50 系列的非接触式 IC 射频卡中, 每一张卡片的序列号都不相同, 都是唯一的。

例如, 某一张卡片的序列号为: 007e0a42H, 则计算异或方法如下:

00h 异或 7eh: (第一和第二字节异或)

0000 0000

0111 1110

第一次异或结果为: 0111 1110 为 7eh;

将 7eh 与 0ah 相异或: 0111 1110

0000 1010

第二次异或结果为: 0111 0100 为 74h;

将 74h 与 42h 相异或: 0111 0100

0100 0010

第三次异或结果为: 0011 0110 为 36h;

这样前四个字节得到的异或结果为 36H, 如果读到的卡片上的序列号的第 5 个字节为 36H, 则证明此次得到的卡片的序列号是有效的。如果读到的卡片上的序列号的第 5 个字节不为 36H, 则证明此次得到的卡片的序列号是无效的。

第四次异或为: 将第三次异或结果 36h 与第 5 个字节相异或, 如果第 5 个字节读到的值为 36H, 则:

将 36h 与 36h 相异或: 0011 0110

0011 0110

第四次异或结果为: 0000 0000 为 00h

这样四次异或结果为 00H, 表明 MCU 读取的卡片序列号是有效的。具体的程序如下, 请读者仔细分析, 定能理解上述关于卡片序列号的校验方法。

程序清单如下:

;===== MCM ANTICOLLISION OPERTION =====

anticollision:

ANTI_REPEAT:

MOV A, #10H

MOV R0, #03H

MOVX @R0, A ;BCNTS = 10H

MOV A, #0CH

```

    SETB ACC. 0      ;AC = "1"
    MOV R0, #01H
    MOVX @R0, A
    MOV R7, #02H     ;DELAY 1000US
    ACALL D500US
    MOV A, #93H
    MOV R0, #00H
    MOVX @R0, A      ;DATA = 93H (ANTICOLLISION OPERATION CODE)
    MOV A, #20H
    MOV R0, #00H
    MOVX @R0, A      ;DATA = 20H
    MOV A, #28H
    MOV R0, #04H
    MOVX @R0, A      ;BCNTR = 28H
    MOV A, #0AH
    MOV R0, #06H
    MOVX @R0, A      ;TOC = 0AH
ANTI_RD_STACON:
    MOV R0, #01H
    MOVX A, @R0      ;READ: STACON ==> A
    JNB ACC. 7, ANTI_RD_STACON ;IF DV = "1"? NO, IT WILL READ AGAIN
;----- NOW DV="1", JUDG OTHER FLAGS-----
    MOV A, #00H
    MOV R0, #06H
    MOVX @R0, A      ;TOC = 00H
    JB ACC. 6, ANTI_TE_ERR ;TE ERR
    JB ACC. 3, ANTI_BE_ERR ;BE ERR
;-----NOW NO ERRORS OCCUR, CAN READ SNR -----
    MOV R7, #04H     ;R7 IS A COUNTER, FOR READING 4-BYTE SERIAL NUMBER
    MOV B, #00H
    MOV R1, #40H
    MOV R0, #00H
ANTI_LOOP:
    MOVX A, @R0
    MOV @R1, A      ;STORE SNR(1)---(4) IN 40H, 41H, 42H, 43H
    XRL B, A        ;CACULATE
    INC R1
    DJNZ R7, ANTI_LOOP
    MOVX A, @R0      ;READ 5th CHK_SUM_VALUE
    XRL A, B        ;CACULATE
    JNZ ANTI_CHK_ERR_EXIT ;If "Z"="0"? If Not, CHK_SUM_VALUE ERROR
;----- NOW THE SNR IS VALID -----
_OK:
    ; MOV B, #00H ;IF SNR "OK", THEN (B) = 00H <----RETURN_VALUE

```

```

MOV R1, #40H                ; (40H) ==> SNR(1)
MOV A, @R1
ACALL SEND_TO_BUF01         ; SEND SNR(1) TO DISP_BUF_LOW
INC R1                      ; (41H) ==> SNR(2)
MOV A, @R1
ACALL SEND_TO_BUF23         ; SEND SNR(2) TO DISP_BUF_HIGH
INC R1                      ; (42H) ==> SNR(3)
MOV A, @R1
ACALL H_SEND_TO_BUF01       ; SEND SNR(3) TO H_DISP_BUF_LOW
INC R1                      ; (43H) ==> SNR(4)
MOV A, @R1
ACALL H_SEND_TO_BUF23       ; SEND SNR(4) TO H_DISP_BUF_HIGH

SETB P2.6                  ; "OK" LAMP NOW WORKING P2.3
ACALL SPK_ON                ; if "ok" then spk on
MOV B, #00H                ; IF SNR "OK", THEN (B) = 00H <----RETURN_VALUE
AJMP ANTI_OK_EXIT

;-----ERROR PROCESS-----
ANTI_TE_ERR:
MOV B, #01H                ; IF "TE_ERR", THEN (B) = 01H
AJMP ANTI_ERR_EXIT
ANTI_BE_ERR:
MOV B, #0AH                ; IF "BE_ERR", THEN (B) = 01H
AJMP ANTI_ERR_EXIT

;----- exit process -----
ANTI_CHK_ERR_EXIT:
MOV B, #08H
MOV A, B
;ACALL H_SEND_TO_BUF23      ; SEND RETURN_ERR_VALUE TO DISP_BUF_HIGH
ANTI_ERR_EXIT:
ANTI_OK_EXIT:
RET

;----- End of MCM AntiCollision Operation -----

```

3. 对 MCM200 (SB201) “Select” (选卡片) 指令的汇编语言应用程序的设计

功能：调用该汇编语言子程序，将执行 MCM “Select” (选卡片) 指令操作。如果卡片读写器的天线之有效工作距离内有多张重叠的 Mifare 1 非接触式 IC 射频卡，则该指令在程序员成功完成 “AntiCollision” 操作之后，将帮助程序员正式选择其中指定的一张卡片进行操作（包括密码存取，密码认证，卡片上指定扇区的读写，等等。）“Select” (选卡片) 指令操作将读取卡片的 SIZE 字节。

程序说明：

1.) 在以下的汇编语言程序中，用到了一些其他的子程序（和上述的 “Request std 和

Request all”指令操作中的相同), 分别为:

D500US: (与上述功能相同);
H_SEND_TO_BUF01: (与上述功能相同);
H_SEND_TO_BUF23: (与上述功能相同);
SEND_TO_BUF01: (与上述功能相同);
SEND_TO_BUF23: (与上述功能相同);
DISPLAY: (与上述功能相同);
H_DISPLAY: (与上述功能相同);
等等。

2.) 在以下的汇编语言程序中, 读者可能会碰到一些寄存器变量, 或其他应用子程序, 这些可能是用于主控程序或是调试而用。特加以说明。

程序清单如下:

```
;=====MCM SELECT OPERATION=====
select:
    ;SETB PSW. 3
    ;SETB PSW. 4
    MOV A, #0CH
    MOV R0, #01H
    MOVX @R0, A                ;STACON() = 0CH
    MOV A, #0F0H
    MOV R0, #02H
    MOVX @R0, A                ;ENABLE() = F0H
    MOV A, #38H
    MOV R0, #03H
    MOVX @R0, A                ;BCNTS() = 38H =56 D
    MOV A, #08H
    MOV R0, #04H
    MOVX @R0, A                ;BCNTR() = 08H <--RECEIVE ONE BYTE:"SIZE"BYTE
    MOV A, #93H
    MOV R0, #00H
    MOVX @R0, A                ;DATA() = 93H -->"SELECT" INSTRUCTION CODE
    MOV A, #70H
    MOV R0, #00H
    MOVX @R0, A                ;IMPORTANT: DATA()=70H--BITCOUNT=70Hbits
;-----
    MOV R7, #04H                ;R7 IS A COUNTER, FOR WRITING 04H BYTES TO DATA
    MOV B, #00H
    MOV R0, #00H
    MOV R1, #40H                ;THE FIRST ADDRESS OF SNR. 40H---43H
SELECT_WRT_LOOP:
    MOV A, @R1
```

```

MOVX @R0, A
XRL B, A
INC R1
DJNZ R7, SELECT_WRT_LOOP
MOV A, B
MOVX @R0, A          ;SEND THE 5th CHK_SUM_VALUE
MOV A, #0AH
MOV R0, #06H
MOVX @R0, A          ;TOC() = 0AH
SEL_RD_STACON_AGAIN:
MOV R0, #01H
MOVX A, @R0          ;READ STACON() ==> A
MOV B, A
JNB ACC. 7, SEL_RD_STACON_AGAIN ;IF "DV"="1"? IF NOT, REPEAT READ
MOV B, A
;----- NOW DV="1", DATA VALID -----
MOV A, #00H
MOV R0, #06H
MOVX @R0, A          ;TOC() = 00H
;----- NOW JUDGE ANY DATA ERRORS -----
MOV A, B
JB ACC. 6, SEL_TE_ERR ;TE ERR
JB ACC. 5, SEL_PE_ERR ;PE ERR
JB ACC. 3, SEL_BE_ERR ;BE ERR
JB ACC. 4, SEL_CE_ERR ;CE ERR
;----- NOW NO ERRORS, READING CARD SIZE -----
MOV R0, #00H
MOVX A, @R0          ;READ SIZE: DATA() ==> A
MOV 44H, A           ;STORE "SIZE" IN 44H
ACALL SEND_TO_BUF01  ;SEND "SIZE" TO DISP_BUF_LOW
SETB P2.6            ;"OK" LAMP NOW WORKING P2.3
ACALL SPK_ON         ;if "ok" then spk on

MOV B, #00H          ;SELECT OPERATION'S "OK" RETURN_VALUE=00H
MOV A, B              ;READY FOR DISSPLAY IN DISP_BUF_HIGH(23)
LJMP SEL_OK_EXIT      ;IF "OK", EXIT SELECT OPERATION
;-----ERRORS PROCESS-----
SEL_TE_ERR:
MOV B, #01H
LJMP SEL_ERR_EXIT
SEL_PE_ERR:
MOV B, #05H
LJMP SEL_ERR_EXIT
SEL_CE_ERR:

```

```

        MOV B, #02H
        LJMP SEL_ERR_EXIT
SEL_BE_ERR:
        MOV R7, #0AH
        ACALL D500US
        MOV B, #0BH
        LJMP SEL_ERR_EXIT
;-----EXIT-----
SEL_ERR_EXIT:
        MOV A, B
SEL_OK_EXIT:
        ACALL SEND_TO_BUF23      ;SEND ERR_RETURN_VALE TO DISP_BUF_HIGH
        RET
;----- End of MCM AntiCollision Operation -----

```

通过以上对 MCM 的几个指令的软件设计，读者应该能有较大的信心，将 MCM 其他的一些指令也能设计成功。以供主控程序随时调用。

(二) . 对 MCM200 (SB201) 组成的 DEMO 电路的主控应用程序设计和开发

1. 硬件电路请参照所附的 DEMO 电路图纸

2. 软件清单如下

程序清单如下:

```
=====
;
;                WRITTEN BY ROCKY, OCT' 1999
;
=====

        ORG 0000H
start:
        LJMP  MAIN
        NOP
        NOP

;-----
;
;      ;ORG 001BH
;      ;LJMP 0500H
;-----
;
        ORG 0080H
MAIN:
        MOV SP, #0E1H
        NOP
        CLR EA                ;CLOSE CPU INT
        CLR  P2.5             ;RESET MCM (-RST="0")
        SETB P2.4             ;RESET MCM (-CS ="1")
        NOP
        ACALL INIT_DISP_BUF
        ACALL DISPLAY
        ACALL H_DISPLAY
        NOP
        CLR P2.6              ;CLOSE "OK" LAMP
        setb p2.0             ;disenable KB function
        mov p1, #11110000b    ;init P1 port:P1.0~~P1.3-->out (line)
                                ;                P1.4~~P1.7-->in  (colum)
        SETB P2.5             ;RESET MCM (-RST="1")
        CLR  P2.4             ;SELECT MCM (-CS="0")
        ACALL D500MS
        MOV A, #80H           ;SOFTWARE_RESET MCM
        MOV R0, #01H
        MOVX @R0, A
        ACALL D500MS
        ACALL SPK_ON          ;SPEAK ON
        ;MOV 21H, #01H        ;THE kb_TEST_CHAR
```

```

KB_LOOKING:
    LCALL KB_SCAN
    ACALL DISPLAY
    ACALL H_DISPLAY
    CLR P2.6                ;CLOSE "OK" LAMP  P2.3
    AJMP KB_LOOKING
KB_ALL_EXIT:
    MOV R2, #00H            ;SET RQT_STD
    SETB P2.5               ;RESET: MCM ON (-RST="1") P1.4
    CLR P2.4                ;SELECT MCM ON (-CS="0") P1.5
;-----
REPEAT:
    SETB P2.5               ;RESET: MCM ON (-RST="1") P1.4
    CLR P2.4                ;SELECT MCM ON (-CS="0") P1.5
;=====
    ACALL REQUEST           ;MCM REQUEST OPERATION
;=====
    ACALL DISPLAY
    ACALL H_DISPLAY
    ACALL D500MS
    CLR P2.6                ;CLEAR "OK" LAMP  P2.3
    ACALL INIT_DISP_BUF
    ACALL DISPLAY
    ACALL H_DISPLAY
;=====
    ACALL ANTICOLLISION     ;MCM ANTICOLLISION OPERATION
;=====
    ACALL DISPLAY
    ACALL H_DISPLAY
    ACALL D500MS
    CLR P2.6                ;CLEAR "OK" LAMP  P2.3
    ACALL INIT_DISP_BUF
    ACALL DISPLAY
    ACALL H_DISPLAY
;=====
    ACALL SELECT            ;MCM SELECT OPERATION
;=====
    ACALL DISPLAY
    ACALL H_DISPLAY
    ACALL D500MS
    CLR P2.6                ;CLEAR "OK" LAMP  P2.3
    ACALL INIT_DISP_BUF
    ACALL DISPLAY
    ACALL H_DISPLAY

```

```

;+++++++ set up register/ram for load_key() ++++++
    mov 7FH, #0ABH
    MOV 7EH, #00H      ; (7EH) = 00H-->SELECT KEYSET0 (KS1=0;KS0=0)
    MOV 7DH, #02H      ; (7DH): 1)SELECT KEYA or KEYB
                        ;      2)SELECT sector_ADDRESS :0-15
    MOV 7CH, #60H      ; (7CH) = 47H-->TKey(1) FIRST ADDRESS
;=====
    ACALL LOAD_KEY ;MCM LOAD_KEY OPERATION
;=====
    ACALL DISPLAY
    ACALL H_DISPLAY
    ACALL d500ms
    CLR P2.6 ;CLEAR "OK"LAMP P2.3
    ACALL INIT_DISP_BUF
    ACALL DISPLAY
    ACALL H_DISPLAY
;+++++++ set up register/ram for Authenticate() ++++++
    ;mov 7fh, #04h
    ;7EH:for future use!
    MOV 7EH, #80H      ; (7EH) = 80H-->SELECT KEYSET0 (KS1=0;KS0=0)

    MOV 7DH, #82H      ; (7DH): 1)SELECT KEYA or KEYB (for auth_)
                        ;      2)SELECT sector ADDRESS :0-15 (for auth_)
    MOV 7CH, #60H      ;Select: Auth_ instruction code
;=====
    ACALL AUTHENTICATION ;MCM AUTHENTICATE OPERATION
;=====
    ACALL DISPLAY
    ACALL H_DISPLAY
    ACALL D500MS
    CLR P2.6 ;CLEAR "OK"LAMP P2.3
    ACALL INIT_DISP_BUF
    ACALL DISPLAY
    ACALL H_DISPLAY
;+++++++ set up register/ram for Write() ++++++
    ; MOV 7EH, #00H      ; (7EH):The first_address(MSB) storing the data which
    ;7EH:for future use!; you want to write the data to the MCM/CARD
    ; MOV 7DH, #50h      ; (7DH): The first_address(LSB) storing the data which
                        ; you want to write the data to the MCM/CARD
    ; MOV 7CH, #0BH      ;Select: written address(block no.):0~~63
    ; LCALL WRITE_KEY_A
;=====
    ; ACALL write ;MCM write OPERATION

```



```

;=====
;   ACALL DISPLAY
;   ACALL H_DISPLAY
;   ACALL D500MS
;   CLR P2.6      ;CLEAR "OK"LAMP    P2.3
;   ACALL INIT_DISP_BUF
;   ACALL DISPLAY
;   ACALL H_DISPLAY
;+++++++ set up register/ram for Read() ++++++
;MOV 7EH, #00H    ;(7EH):The first address(HIGH_byte) which you want to
;7EH:for future use! ;read the data from the MCM/CARD into cpu_ram
MOV 7DH, #50h     ;(7DH):The first address(LOW_byte) which you want to
;               ; read the data from the MCM/CARD into cpu_ram
MOV 7CH, #08h     ;Select:The address(block no.) you want to read:0-63
;=====
;   LCALL READ      ;MCM READ OPERATION
;=====
;   ACALL DISPLAY
;   ACALL H_DISPLAY
;   ACALL D500ms
;   CLR P2.6      ;CLEAR "OK"LAMP    P2.3
;   ACALL INIT_DISP_BUF
;   ACALL DISPLAY
;   ACALL H_DISPLAY
;=====
;   LCALL HALT      ;MCM HALT OPERATION
;=====
;   ACALL DISPLAY
;   ACALL H_DISPLAY
;   ACALL D500MS
;   CLR P2.6      ;CLEAR "OK"LAMP    P2.3
;   ACALL INIT_DISP_BUF
;   ACALL DISPLAY
;   ACALL H_DISPLAY
;=====
;   LCALL DISP_RD_BLK      ;DISPLAY JUST READ BLOCK VALUES
;=====
;   LCALL CLEAR_DISP_RD_BLK_IN_CPU ;CLEAR :FILL WITH #11H
;=====
;   CLR P2.6      ;CLEAR "OK"LAMP    P2.3
;   ACALL INIT_DISP_BUF
;   ACALL DISPLAY
;   ACALL H_DISPLAY
;   CLR P2.5      ;RESET MCM (-RST="0") P1.4

```

```
;SETB P2.4      ;RESET MCM (-CS="1")    P1.5
MOV A, #80H      ;SOFTWARE_RESET MCM
MOV R0, #01H
MOVX @R0, A
;LJMP REPEAT
RET
```

;

```

D500US:
D500US_LOOP1:
    MOV R6, #05H
D500US_LOOP2:
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    NOP
    DJNZ R6, D500US_LOOP2
    DJNZ R7, D500US_LOOP1
    RET

```

;

```

D500MS:
    setb psw.3
    setb psw.4
    MOV R3, #19H
LOOP_1: MOV R4, #32H
LOOP_2: MOV R5, #64H
LOOP_3: DJNZ R5, LOOP_3
        DJNZ R4, LOOP_2
        DJNZ R3, LOOP_1
        clr psw.3
        clr psw.4
        RET

```

;

```

D2S:
    setb psw.3
    setb psw.4
    MOV R6, #06H
LOOP:  ACALL D500MS
        DJNZ R6, LOOP
        clr psw.3
        clr psw.4
        RET

```

;

```

SPK_ON:
    setb psw.3
    setb psw.4
    CLR P2.7
    ACALL D500MS

```



```

        SETB P2.7
        clr psw.3
        clr psw.4
        RET
;-----
DISPLAY:
        SETB PSW.3
        SETB PSW.4
        CLR P2.2      ;SELECT MC14499'S "-ENB"
        MOV R0,#36H   ;DISPLAY BUF FIRST ADDRESS
        MOV R1,#05H   ;R1 AS A COUNTER
NEXT_CHAR: MOV A , @R0
        SWAP A
        ACALL DISP_ONE
        INC R0
        DJNZ R1,NEXT_CHAR
        SETB P2.2     ;RELEASE MC14499'S "-ENB"
        CLR PSW.3
        CLR PSW.4
        RET
;-----
DISP_ONE:
        MOV R2,#04H
NEXT_BIT:SETB P2.1     ;SELECT MC14499'S "CLK "=1
        RLC A         ;ONE BIT MOVE TO PSW.C
        MOV P2.0, C
        CLR P2.1      ;"CLK "=0
        DJNZ R2, NEXT_BIT
        RET
;-----
H_DISPLAY:
        SETB PSW.3
        SETB PSW.4
        CLR P2.3      ; -ENB
        MOV R0,#31H
        MOV R1,#05H
        NOP
        NOP
H_NEXT_CHAR: MOV A, @R0
        SWAP A
        ACALL H_DISP_ONE
        INC R0
        DJNZ R1,H_NEXT_CHAR
        SETB P2.3     ; -ENB

```

```

        CLR PSW. 3
        CLR PSW. 4
        RET
;-----
H_DISP_ONE:
        MOV R2, #04H
H_NEXT_BIT: SETB P2. 1      ; CLK
        RLC  A
        MOV P2. 0, C      ; DATA
        CLR P2. 1      ; CLK
        DJNZ R2, H_NEXT_BIT
        RET
;=====
INIT_DISP_BUF:
        SETB PSW. 3
        SETB PSW. 4
        MOV R0, #0AH
        MOV R1, #31H
INIT_AGAIN: MOV @R1, #00H    ;SET 31H--40H UNITS AS 00H
        INC R1
        DJNZ R0, INIT_AGAIN
        CLR PSW. 3
        CLR PSW. 4
        RET
;-----
SEND_TO_BUF01:
        SETB PSW. 3
        SETB PSW. 4
        MOV B, A
        MOV 3AH, A          ; (3AH)
        SWAP A
        MOV 39H, A          ; (39H)
        MOV A, B
        CLR PSW. 3
        CLR PSW. 4
        RET
;-----
SEND_TO_BUF23:
        SETB PSW. 3
        SETB PSW. 4
        MOV B, A
        MOV 38H, A          ; (38H)
        SWAP A
        MOV 37H, A          ; (37H)

```

```

        MOV A, B
        CLR PSW. 3
        CLR PSW. 4
        RET
;-----
H_SEND_TO_BUF01:
        SETB PSW. 3
        SETB PSW. 4
        MOV B, A
        MOV 35H, A                ; (35H)
        SWAP A
        MOV 34H, A                ; (34H)
        MOV A, B
        CLR PSW. 3
        CLR PSW. 4
        RET
;-----
H_SEND_TO_BUF23:
        SETB PSW. 3
        SETB PSW. 4
        MOV B, A
        MOV 33H, A                ; (33H)
        SWAP A
        MOV 32H, A                ; (32H)
        MOV A, B
        CLR PSW. 3
        CLR PSW. 4
        RET
;-----
SET_POINT0:
        SETB PSW. 4
        SETB PSW. 3
        MOV R1, #36H              ; (36H)
        MOV @R1, #01H
        MOV R1, #31H              ; (31H)
        MOV @R1, #02H
        CLR PSW. 3
        CLR PSW. 4
        RET
;-----
INDICATE:
        CLR P2. 6                ;INDICATE P2. 6 AS "0-1-0"
        SETB P2. 6
        ACALL D500MS

```


CLR P2. 6

RET

;

```

;=====MCM REQUEST OPERATION=====
request:
    RET
;-----
;=====MCM ANTICOLLISION OPERATION=====
anticollision:
    RET
;-----
;=====MCM SELECT OPERATION=====
select:
    RET
;-----
;===== AUTHENTICATION =====
authentication:
    RET
;-----
;===== LOAD KEY =====
LOAD_KEY:
    RET
;-----
;=====MCM WRITE OPERATION=====
write:
    RET
;-----
;=====MCM read OPERATION=====
read:
    RET
;-----

```

```

;===== HALT OPERATION=====
halt:
    MOV A, #10H
    MOV R0, #03H
    MOVX @R0, A
    MOV A, #04H
    MOV R0, #04H
    MOVX @R0, A
    MOV A, #50H
    MOV R0, #00H
    MOVX @R0, A
    MOV A, #00H
    MOV R0, #00H
    MOVX @R0, A
;----- SET TOC() = 0AH -----
    MOV A, #0AH
    MOV R0, #06H
    MOVX @R0, A          ;TOC() = 0AH
;----- READ STACON() -----
HALT_RD_STACON_AGAIN:
    MOV R0, #01H
    MOVX A, @R0          ;READ STACON()
;----- JUDGE DV FLAG -----
    JNB ACC. 7, HALT_RD_STACON_AGAIN
;----- CLOSE TOC() -----
    MOV B, A
    MOV A, #00AH
    MOV R0, #06H
    MOVX @R0, A          ;TOC() = 00H
    MOV A, B
;----- JUDG ANY ERRORS -----
    JB ACC. 6, HALT_OK_EXIT ;TE="1" IS HALT "OK" FLAG
;----- HAVE ANY ERRORS -----
    MOV B, #06H
    LJMP HALT_ERR_EXIT
;----- EXIT -----
HALT_OK_EXIT:
    MOV B, #66H          ; "ok" disp_flag
    SETB P2. 6           ; "OK" LAMP NOW WORKING P2. 3
    ACALL SPK_ON         ;if "ok" then spk on
HALT_ERR_EXIT:
    MOV A, B
    ACALL SEND_TO_BUF01  ;SEND RETURN_VALE TO DISP_BUF_LOW
    RET

```


;===== End of MCM HALT OPERATION =====

```

;-----DISPLY READ BLOCK VALUE(16 BYTES) -----
DISP_RD_BLK:
    MOV R0, #50H
    MOV R7, #04H
DISP_RD_BLK_LOOP:
    MOV A, @R0
    LCALL H_SEND_TO_BUF23    ;SEND RETURN_VALE TO DISP_BUF_LOW
    INC R0
    MOV A, @R0
    LCALL H_SEND_TO_BUF01    ;SEND RETURN_VALE TO DISP_BUF_HIGH
    INC R0
    MOV A, @R0
    LCALL SEND_TO_BUF23     ;SEND RETURN_VALE TO H_DISP_BUF_LOW
    INC R0
    MOV A, @R0
    LCALL SEND_TO_BUF01     ;SEND RETURN_VALE TO H_DISP_BUF_HIGH
    INC R0
    LCALL DISPLAY
    LCALL H_DISPLAY
    LCALL D500MS
    LCALL D500MS
    DJNZ R7, DISP_RD_BLK_LOOP
    RET
;-----CLEAR DISPLY_READ_BLOCK VALUE(16 BYTES) IN CPU RAM -----
CLEAR_DISP_RD_BLK_IN_CPU:
    MOV R0, #50H
    MOV R7, #10H
CLEAR_DISP_RD_BLK_LOOP:
    MOV @R0, #11H           ;SEND #11H TO DISP_RD_BLK_IN_CPU
    INC R0
    DJNZ R7, CLEAR_DISP_RD_BLK_LOOP
    RET

```

(待续)