RD800M 射频读写器
使用说明书

第一章　RD 系列非接触式 IC 卡读写器简介

1.1 概述

RD800M 非接触式 IC 卡读写器读写非接触射频卡，由主机、天线、串行接口等组成，通过 RS232 串行接口及 485 接口能实现同 PC 机及相关设备的连接，带有 SAM 卡操作，可实现安全发卡及满足安全领域的需要。随机提供各种平台的驱动开发包，附带的演示程序实现访问射频卡的全部功能，并带有自动测卡操作。RD800M 非接触式 IC 卡读写器是开发非接触式 IC 卡相关产品及系统集成必备的前端处理设备，其丰富、完善的接口函数，可方便地应用于工商、电信、邮政、税务、银行、保险、医疗及各种收费、储值、查询等智能卡管理应用系统中。

支持的卡型

- PHILIPS 公司的 MF1 S50、MF1 L10
- SIEMENS 公司的 SLE44R35、SLE44R31

可支持的函数接口：

- MS-DOS 操作系统下的 TURBO C、BORLAND C、MicroSofte C、FOXBASE、FOXPRO FOR DOS、CLIPPER 5.x 版等。
- Windows3X 下的 16 位开发平台，如：Visual Basic 3.0、Delphi 1.0、Power Builder 4.0、Visual FoxPro 3.0、Visual C++ 1.52 等。
- Windows 9x/2k/NT/Xp 下的 32 位开发平台，如 Visual Basic 、Delphi 、Power Builder、Visual FoxPro、Visual C++、C++ Builder 等及这些开发平台的高版本。
- SCO UNIX 下的 C 语言开发。
- RedHat Linux 驱动开发

特点

- 工作频率 13.56MHZ，支持 Mifare 标准
- 标准的 ESD 协议，RS232 串行接口,485 远距离通讯接口
- 波特率从 9600 至 115200bit/s，自动侦测
- 单电源 5V 供电，提供电路保护。
- 高速访问射频卡，通信速率为 106Kbit/s
- 数据加密和双向验证
- 与射频卡标准操作距离 25mm
- 防冲突，可同时读取多张射频卡
- SAM 卡操作，读写符合 T=0 和 T=1 协议的 CPU 卡
- 提供了丰富的二次开发平台和范例

1.2 读写器装箱清单

标准：软盘 1 张、保修单 1 张、读写器 1 台、串口线 1 条、电源线 1 条、装箱单 1 张
5V 稳压电源一个

选件：485 通讯网卡、485 通讯线

检查完毕后，请详细阅读使用说明书和有关文档。

1.3 读卡器连接方式

串口方式：将串口线一端插入读卡器，一端插入计算机的串行口，接上 5V 稳压电源。

485 方式：将 485 网卡接在计算机的串行口，将 485 连线水晶头一端接在读写器，另一端接在 485 网卡的接口上，接上 5V 稳压电源。485 的连线制作方式见附录一。

1.4 开机情况

读写器联接正常后，打开电源，读写器蜂鸣一声，数码管全部显示 0.5 秒后，显示时间。

1.5 程序安装

步骤：

- 将读写器连接在计算机通讯口上并接上电源
- 连通计算机和读写器，接法请参照**读写器连接方式，根据开机情况判断有没有连接正确。**
- 开机进入 WINDOWS3.X/95/98/2000/XP 中
- 将随机软盘或光盘插入驱动器中，运行 Rd800M.EXE 即可。

注意：安装软件在根目录下建立缺省名为\RD800 的目录，所有驱动软件均在此目录下。安装完毕后**请仔细阅读说明书**。

1.6 软件

RD800M 读写器的软件包括三部分：演示软件、库函数和应用范例

a. 演示软件

IC 卡读写器功能演示软件 WINDOWS 版 RFDEMO.EXE，您可以用该软件来测试您的读写器有没有连接正确或测试您的卡的卡类型或进行一些卡功能测试。

注意：在开发过程中请检查自己的读写卡过程有没有正确，用该演示程序来验证是一个好办法！

b. 库函数

- C 语言接口函数库(Borland C、Microsoft C)
- FOXPRO FOR DOS(2.6) 接口函数库
- WINDOWS 32 位动态库

请根据您的开发环境来选用正确的库函数。相信应用范例可能帮您的忙。

c. 应用范例

包括 Visual Basic、Power Builder、Borland DELPHI、VFP、FORPRO FOR DOS 等

d. 软件目录

\Rd800

\Manual.pdf 本文档
\rfdemo.exe Windows 下的演示程序
\32dll*.* 32 位的 Windows 函数库
\c.lib*.* Dos 下的 C 语言函数库
\clipper*.* Clipper 开发语言的函数库
\dosdemo*.* Dos 下的演示程序
\examples*.* 各种开发语言的例程序
\foxpro.dos*.* Foxpro For Dos 语言的开发函数库

1.7 技术指标

- 通讯接口：RS232 、485
- 串口的波特率：9600 BPS —115200 BPS
- 电源：DC5V±10%
- 最大功耗：100 mW
- 环境温度：0° ~ 50 ° C
- 相对湿度：30% ~ 95%
- 外型尺寸：长 x 宽 x 高: 145mm×110mm×30mm
- 重量： 1 公斤左右

第二章 读写器驱动函数说明

2.1 函数使用规则

(1) 首先要调用通讯口初始化函数 dc_init ，其返回值为设备标识符，它将作为其它函数的调用参数。

(3) 调用 WINDOWS 32 位动态库时，程序退出之前要执行 dc_exit (HANDLE icdev) 函数，关闭串口，释放句柄 icdev；**否则再次初始化串口将出错**。

(4) 函数调用错误类型，请参照**函数错误类型代码**。所有函数的错误代码均以负数形式返回；Foxpro For Dos 例外。

(5) 动态库的位置应该在声明的相应目录中或缺省的目录当中，否则会有无法寻找到动态库的错误。

(6) 函数的十六进制 HEX 方式调用中，传入和读出的字符数组是以十六进制字符串的方式进行的，其余参数调用方式相同，所以在函数详细说明中不再列出。

注意：函数详细的使用方法，参考 Rd800\EXAMPLES 目录下提供的**范例**。

2.2 函数说明

32 位动态库的函数说明

注意:以下函数名的大小写必须严格区分,否则无法调用

1、HANDLE dc_init(int port,long baud)

说明：初始化串口的函数

参数说明：__int16 port　通讯口号 port=0,1,2,3

　　baud 波特率　 9600 到 115200

返回：调用成功则返回设备描述符 >=0

举例：HANDLE icdev;

　　icdev=dc_init(1,9600);/*初始化 com2 口*/

2、__int16 dc_exit(HANDLE icdev)

说明：关闭串口的函数

参数说明：HANDLE icdev　dc_init 返回的设备描述符

返回：成功返回 0

举例：dc_exit(icdev);

3、__int16 dc_card(HANDLE icdev,unsigned char _Mode,unsigned long *_Snr);

说明：寻卡函数，能返回在工作区域内某张卡的序列号

参数说明：HANDLE icdev　dc_init 返回的设备描述符

　　unsigned char _Mode　设置卡的模式，可以是 IDLE 或 ALL

　　有关_Mode 的解释请参阅 halt()函数说明；IDLE=0, ALL=1

　　unsigned long *_Snr　存放找到的卡的序列号的地址

返回：成功则返回 0;

举例：__int16 st;

　　unsigned long snr;

　　st=dc_card(icdev,IDLE,&snr);

4 、__int16 dc_authentication(HANDLE icdev ,unsigned char _Mode, unsigned char _SecNr)

说明：验证某一扇区密码的函数；
参数说明：HANDLE icdev　dc_init 返回的设备描述符；
unsigned char _Mode　该参数决定验证密码的方法；
_Mode=0: 用 KEYSET0 和 KEYA 验证
_Mode=1: 用 KEYSET1 和 KEYA 验证
_Mode=2: 用 KEYSET2 和 KEYA 验证
_Mode=4: 用 KEYSET0 和 KEYB 验证
_Mode=5: 用 KEYSET1 和 KEYB 验证
_Mode=6: 用 KEYSET2 和 KEYB 验证
unsigned char _SecNr　要验证密码的扇区号(0)
返回：成功则返回 0；
举例：__int16 st;
st=dc_authentication(icdev,0,0);
附加说明：每张卡上有 A 密码和 B 密码，可根据实际需要确定是否使用 B 密码，这由卡的存取控制位来决定(由上面的表可参阅)。此外，读写器中可以存放三套密码，以备读写器有多种用途,用来校验具有不同密码的卡。可用 load_key()来分别装入，只有装入后才能使用验证密码函数验证。

5、__int16 dc_halt(HANDLE icdev)
说明：中止卡操作函数
参数：HANDLE icdev　dc_init 返回的设备描述符;
返回：成功则返回 0；
举例：__int16 st;
st=dc_halt(icdev);
附加说明：使用 card()函数时，有个 _Mode 参数,如果模式选择为 0 则在对卡进行读写操作完毕后，执行 halt();则该卡进入 HALT 模式，只能当该卡离开操作区域并再次进入时读写器才能够再次操作它。

6、__int16 dc_read(HANDLE icdev,usigned char _Adr,unsigned char *_Data)
说明：读函数，一次必须读一个块；
参数说明：HANDLE icdev　dc_init 返回的设备描述符
unsigned char _Adr　所读数据的地址，_Adr=(0-63)
unsigned char *_Data　指向存放被读数据的地址
返回：成功则返回 0；
举例：__int16 st;
static unsigned char data[16]
st=dc_read(icdev,0,data);

7、__int16 dc_read_hex(HANDLE icdev,usigned char _Adr,unsigned char *_Data)
说明：16 进制读函数，一次必须读一个块；
参数说明：HANDLE icdev　dc_init 返回的设备描述符
unsigned char _Adr　所读数据的地址，_Adr=(0-63)
unsigned char *_Data　指向存放被读数据的地址
返回：成功则返回 0；

举例：__int16 st;
static unsigned char data[32]
st=dc_read(icdev,0,data);/*读出数据格式如 write_hex*/

8、__int16 dc_write(HANDLE icdev,unsigned char _Adr,unsigned char *_Data)
说明：写函数，一次必须写一个块；
参数说明：HANDLE icdev　dc_init 返回的设备描述符；
unsigned char _Adr　所写数据的地址，_Adr=(1-63)
unsigned char *_Data　指向存放要写数据的地址，块长度=16
返回：成功则返回 0；
举例：__int16 st;
static unsigned char data[16]
/* 给 data 赋值*/
st=dc_write(icdev,1,data);/*写入块 1*/

9、__int16 dc_write_hex(HANDLE icdev,unsigned char _Adr,unsigned char *_Data)
说明：16 进制写函数，一次必须写一个块；
参数说明：HANDLE icdev　dc_init 返回的设备描述符；
unsigned char _Adr　所写数据的地址，_Adr=(1-63)
unsigned char *_Data　指向存放要写数据的地址，块长度=16
返回：成功则返回 0；
举例：__int16 st;
unsigned char data[32]="aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa";
/* data 赋值为 16 个 0xaa*/
st=dc_write_hex(icdev,1,data);/*写入块 1*/

10、__int16 dc_increment(HANDLE icdev,unsigned char _Adr,unsigned long _Value)
说明：增值函数
参数说明：HANDLE icdev　dc_init 返回的设备描述符；
unsigned char _Adr　所增值的地址，_Adr=(1-63)
unsigned long _Value　要增加的值
返回：成功则返回 0；
举例：__int16 st;
unsigned long value;
/* 给 value 赋值*/
value=1;
st=dc_increment(icdev,1,value);/*将块 1 的值增加 value*/

11、__int16 dc_decrment(HANDLE icdev,unsigned char _Adr,unsigned long _Value)
说明：减值函数
参数说明：HANDLE icdev　dc_init 返回的设备描述符；
unsigned char _Adr　要减值的地址，_Adr=(1-63)
unsigned long _Value　要减少的值
返回：成功则返回 0；

举例：__int16 st;
unsigned long value;
/* 给 value 赋值*/
value=1;
st=dc_decrement(icdev,1,value);/*将块 1 的值减少 value*/

12、__int16 dc_initval(HANDLE icdev,unsigned char _Adr,unsigned long _Value)
说明：初始化值函数
参数说明：HANDLE icdev　dc_init 返回的设备描述符；
unsigned char _Adr　要初始化值的地址，_Adr=(1-63)
unsigned long _Value　要初始化的值
返回：成功则返回 0；
举例：__int16 st;
unsigned long value;
/* 给 value 赋值*/
value=1000;
st=dc_initval(icdev,1,value);/*将块 1 的值初始化为 value*/
附加说明：作为数值处理的块，是以特殊格式存贮的，所以必须用初始化值函数初始化，初始化之后方可进行读、减、加的处理。

13、__int16 dc_readval(HANDLE icdev,unsigned char _Adr,unsigned long *_Value)
说明：读值函数
参数说明：HANDLE icdev　dc_init 返回的设备描述符；
unsigned char _Adr　要读值的地址，_Adr=(1-63)
unsigned long _Value　存放读出值的地址
返回：读出值存放在*_Value 中
举例：__int16 st;
unsigned long value;
st=dc_readval(icdev,1,&value);/*读出块 1 的值，放入 value*/

14、__int16 dc_transfer(HANDLE icdev,unsigned char _Adr)
说明：传送函数
参数说明：HANDLE icdev　dc_init 返回的设备描述符；
unsigned char _Adr　要传送的地址
返回：无
举例：__int16 st;
st=dc_transfer(icdev,1);
附加说明：见 dc_restore()的附加说明；

15、__int16 dc_restore(HANDLE icdev,unsigned char _Adr)
说明：回传函数，将_Adr 地址的值传入卡的内部 RAM
参数说明：HANDLE icdev　dc_init 返回的设备描述符；
unsigned char *_Adr　要进行回传的地址
返回：无

举例：__int16 st;
st=dc_restore(icdev,1);
附加说明：用户可以用此函数将某一块内的数值传入内部 RAM，然后用 transfer()函数传送到另一块中去，就达到了块与块之间数值传送的目的。

16、__int16 dc_load_key(HANDLE icdev,unsigned char _Mode,unsigned char SecNr, unsigned char * _Nkey)
说明：装入密码函数
参数说明：HANDLE icdev　dc_init 返回的设备描述符；
unsigned char _Mode 同函数 authentication()
unsigned char _SecNr 同函数 authentication()
unsigned char *_Nkey　包含了要写入硬件系统 RAM 中的卡密码
返回：成功则返回 0；
举例：

```
//key A and key B
unsigned char tk[2][7]= { {0xa0,0xa1,0xa2,0xa3,0xa4,0xa5},
                          {0xb0,0xb1,0xb2,0xb3,0xb4,0xb5}
                        };
/*装入 1 扇区的 A 密码 0 套，*/
 if((dc_load_key(icdev,0, 1,tk[1]))!=0){
      printf("Load key error!");
      return;
     }
```

附加说明： 而*_Nkey 是与卡中的密码相应的密码，可以是密码 A 或密码 B，是根据存取控制，用户可以装入相应的密码并验证。

17、__int16 dc_load_key_hex (HANDLE icdev,unsigned char _Mode, unsigned char SecNr,char * _Nkey)
说明：装入密码(用一串 16 进制表示)函数
参数说明：HANDLE icdev　dc_init 返回的设备描述符；
unsigned char _Mode 同函数 authentication()
unsigned char _SecNr 同函数 authentication()
unsigned char *_Nkey　包含了要写入硬件系统 RAM 中的卡密码
返回：成功则返回 0；
举例：

```
//key A
unsigned char tk[]= "a0a1a2a3a4a5"/*等同于{0xa0_.0xa5}*/
/*装入 1 扇区的 A 密码|0 套，*/
 if((dc_load_key_Hex(icdev,0, 1,tk))!=0){
      printf("Load key error!");
      return;
      }
```

附加说明： 而*_Nkey 是与卡中的密码相应的密码，可以是密码 A 或密码 B，是根据存取控制，用户可以装入相应的密码并验证。

18、int dc_beep(HANDLE icdev,unsigned int _Msec)
说明：蜂鸣函数，使读写器发出蜂鸣声；
参数说明：HANDLE icdev　dc_init 返回的设备描述符；
　　unsigned int _Msec:蜂鸣时间的长短，单位是 10 毫秒；
返回：成功则返回 0；
举例：int st;
　　st=dc_beep(icdev,10);/*鸣叫 100 毫秒*/

19. int　dc_high_disp(HANDLE icdev,unchar offset,unchar displen,unchar *dispstr);
说明：数码显示.
参数说明：HANDLE icdev　dc_init 返回的设备描述符；
　　unchar offset:要显示字符串的偏移地址　范围 1 到 8
　　unchar displen:要显示的长度 ,长度为 0 时清屏
　　unchar * dispstr:要显示的字符串 高半字节为 8 时小数点亮
返回：成功则返回 0；
举例：int st;
　　uchar dispstr[5]={0x89,0x0a,0x8c,0x01,0x02}
　　st=dc_high_disp(icdev,1,5,dispstr); //从 1 开始显示 ”9.ac.12”

20.int dc_request(int icdev,unsigned char _Mode,unsigned int *TagType)
说明：寻卡请求。
参数说明：HANDLE icdev　dc_init 返回的设备描述符；
　　unsigned char _Mode:寻卡模式；
附加说明：_Mode:=0　选择 IDLE 模式，一次只读一张卡。
　　=1　选择 ALL 模式，一次可读多张卡。
　　=2　只对卡序列号等于_Snr 的卡操作。
　　unsigned int *Tagtype:指向返回的卡类型的数值；
返回：成功则返回 0；
举例：
　　int st;
　　unsigned int *tagtype;
　　st=dc_request(IDLE,tagtype);

21.int dc_anticoll(int icdev,unsigned char _Bcnt,unsigned long *_Snr)
说明：防止卡冲突。
参数说明：　HANDLE icdev　dc_init 返回的设备描述符；
　　unsigned char _Bcnt:
　　unsigned long *_Snr:指向返回的卡序列号；
返回：成功则返回 0；
举例：
　　int st;
　　unsigned long *snr;
　　st=anticoll(0,snr);

22．int dc_select(int icdev,unsigned long _Snr,unsigned char *_Size)
说明：选卡函数，从多个卡中选取一特定序列号的卡。
参数说明：HANDLE icdev　dc_init 返回的设备描述符；
　　unsigned long _Snr:卡序列号;
　　unsigned char *_Size:指向返回的卡容量的数据.
返回：成功则返回 0；
举例:

```
int st;
unsigned long snr=239474;
unsigned char  size;
st=dc_select(snr,&size);
```

23．int　dc_gettime(HANDLE icdev,unsigned char *time)
说明：取时间函数，用于读取设备时间。
参数说明：HANDLE icdev　dc_init 返回的设备描述符；
　　unsigned char * time:返回的时间串
　　time 格式：（各占两字节）“year,week,month,day,hour,minitue,second”
　　年，星期几，月，日，小时，分，秒
返回：成功则返回 0；
相关函数：dc_gettimehex(HANDLE icdev,char *time)返回 HEX 串。
举例:

```
int st;
unsigned char timestr[20];
st=dc_gettime(icdev,timestr);
```

24．int　dc_settime(HANDLE icdev,unsigned char *time)
说明：设置时间函数。
参数说明：HANDLE icdev　dc_init 返回的设备描述符；
　　unsigned char * time:要设置的时间串
返回：成功则返回 0；
相关函数：dc_settimehex(HANDLE icdev,char *time)
　　以十六进制串方式设置时间
举例:

```
int st;
unsigned char timestr[20];
memcpy(timestr,”01020122170740”); //01 年 1 月 22 日星期二，17 点 7 分 40 秒
st=dc_settimehex(icdev,timestr);
```

25．int　dc_setbright(HANDLE icdev,unsigned char bright)
说明：设置亮度函数。
参数说明：HANDLE icdev　dc_init 返回的设备描述符；
　　unsigned char bright:要设置的显示亮度值 0-15;
返回：成功则返回 0；
举例:

```
int st;
st= dc_setbright( icdev,9);
```

26．int　dc_ctl_mode(HANDLE icdev,unsigned char mode)
说明：控制显示函数。
参数说明：HANDLE icdev　dc_init 返回的设备描述符；
　　unsigned char mode: 0 计算机控制显示，1 读写器控制显示,显示设备时间;
返回：成功则返回 0；
举例:

```
int st;
st= dc_ctl_mode ( icdev,0);
```

27． int dc_disp_mode(HANDLE icdev,unsigned char mode)
说明：显示格式函数。
参数说明：HANDLE icdev　dc_init 返回的设备描述符；
　　unsigned char mode:0 显示年-月-日 1 显示时-分-秒;
返回：成功则返回 0；
举例:

```
int st;
st= dc_ disp _mode ( icdev,0);
```

28． int dc_cpureset(HANDLE icdev,unsigned char *rlen,unsigned char *databuffer)
说明：SAM 卡上电复位函数。
参数说明：HANDLE icdev　dc_init 返回的设备描述符；
　　rlen：返回复位信息的长度
　　databuffer：保存返回的复位信息
返回：成功则返回 0；
相关函数：dc_cpureset_hex　以 HEX 的方式保存返回信息
举例:

```
int st;
unsigned char　rlen, recdata[100];
st= dc_cpureset ( icdev,&rlen,recdata);
```

29． int dc_cpuapdu(HANDLE icdev,unsigned char slen,unsigned char *senddata,
　　unsigned char *rlen,unsigned char *recdata)
说明：SAM 卡 APDU（应用协议数据单元）信息交换函数。该函数封装了 T=0 和 T=1 操作.
参数说明：HANDLE icdev　dc_init 返回的设备描述符；
　　slen: 发送到 SAM 卡的信息
　　senddata:保存要发给 SAM 卡的信息
　　rlen：返回复位信息的长度
　　recdata：保存返回的复位信息
返回：成功则返回 0；
相关函数：dc_cpuapdu_hex　以 HEX 的方式执行
举例:

```
int st;
unsigned char    slen,rlen,sneddata[100], recdata[100];
slen=5;
senddata[0]=0x00;senddata[1]=0x84;senddata[2]=0x00,senddata[3]=0x00;senddata[4]=0x04;
st= dc_cpuapdu ( icdev,slen,senddata,&rlen,recdata);
```

30． int dc_cpuapdusource(HANDLE icdev,unsigned char slen,unsigned char *senddata,
unsigned char *rlen,unsigned char *recdata)

说明：SAM 卡 APDU（应用协议数据单元）信息交换函数。该函数不封装，用户需自行判断协议类型并组织数据发送。

参数说明：HANDLE icdev　dc_init 返回的设备描述符；
slen: 发送到 SAM 卡的信息
senddata:保存要发给 SAM 卡的信息
rlen：返回复位信息的长度
recdata：保存返回的复位信息

返回：成功则返回 0；

相关函数：dc_cpuapdusource_hex　以 HEX 的方式执行

举例: T=1 协议卡的操作

```
int st;
unsigned char    slen,rlen,sneddata[100], recdata[100];
slen=5;
senddata[0]=nad;senddata[1]=pcb;senddata[2]=5,senddata[3]=0x00;senddata[4]=0x84;
senddata[5]=0x00;senddata[6]=0x00;senddata[7]=0x08;
for(st=0;st<8;st++)senddata[8]^=senddata[st];   //计算异或和
st= dc_cpuapdusource ( icdev,slen,senddata,&rlen,recdata);
```

31． int dc_cpudown(HANDLE icdev)

说明：SAM 卡下电函数。

参数说明：HANDLE icdev　dc_init 返回的设备描述符；

返回：成功则返回 0；

举例:

```
int st;
st= dc_cpudown ( icdev);
```

32．int　dc_swr_eeprom(HANDLE icdev,int offset,int lenth,unsigned char* send_buffer)

说明：写存储 EEPROM 的函数。
范围 0-1597

参数说明：HANDLE icdev　dc_init 返回的设备描述符；
offset：偏移地址
length：长度
send_buffer：存储数据内容

返回：成功则返回 0；

举例:

```
int st;
```

data[0]=0x12;data[1]=0x34;data[3]=0x56;
st= dc_swr_eeprom (icdev，32，3，data);

33．int　dc_srd_eeprom(HANDLE icdev,int offset,int lenth,unsigned char* read_buffer)
说明：读存储 EEPROM 的函数。
范围 0-1597
参数说明：HANDLE icdev　dc_init 返回的设备描述符；
offset：偏移地址
length：长度
read_buffer：读出数据内容
返回：成功则返回 0；
举例:
int st;
unsigned char data[10];
st= dc_srd_eeprom (icdev，32，10，data);

34. int　dc_disp_str(HANDLE icdev,unchar *dispstr);
说明：数码显示.
参数说明：HANDLE icdev　dc_init 返回的设备描述符；
unchar * dispstr:要显示的字符串，由数值字符’0’-‘F’和‘.’组成；
数值字符数不超过 8 个，显示自动居后；
dispstr 长度为 0 时，显示全灭；
返回：成功则返回 0；
举例：int st;
st=dc_high_disp(icdev,” 123.45”); //显示 ”123.45”

FoxPro for Dos 函数库（.PLB）

1、init(_Comstr,_Baudrate);

说明：初始化串口函数

参数说明：_Comstr　是一个字符串，为“COM1”或　“COM2”。

Baudrate 是一个整数，用来设置通信波特率，如 9600、19200 直到 115200 皆可。一般为 115200。

返回：调用成功则返回 0

举例：

st=init("COM1",115200);

2、exit();

说明：恢复串口函数

参数说明：无参数

返回：无

举例：exit();

3、card(_Mode);

说明：寻卡函数，能返回在工作区域内某张卡的序列号

参数说明：　_Mode 是一个整数，设置卡的模式，可以是 0(IDLE)或 1(ALL),有关_Mode 的说明请参阅 halt()函数说明；

返回：返回 9 字节字符串，第一个字节表示执行状态，后 8 字节为序列号；

举例：

snr=card(0);

4、pass(_Mode, _SecNr)

说明：验证某一扇区密码的函数；

参数说明：_Mode 是一个整数，决定验证密码的方法；

_Mode=0: 用 KEYSET0 和 KEYA 验证

_Mode=1: 用 KEYSET1 和 KEYA 验证

_Mode=2: 用 KEYSET2 和 KEYA 验证

_Mode=4: 用 KEYSET0 和 KEYB 验证

_Mode=5: 用 KEYSET1 和 KEYB 验证

_Mode=6: 用 KEYSET2 和 KEYB 验证

_SecNr 是一个整数，为要验证密码的扇区号(0)

返回：成功则返回 0；

举例：

用 A 密码和 0 套密码去验证扇区 0

st=pass(0,0);

附加说明：每张卡上有 A 密码和 B 密码，可根据实际需要确定是否使用 B 密码，这由卡的存取控制位来决定(由上面的表可参阅)。此外，读写器中可以存放三套密码，以备读写器有多种用途,用来校验具有不同密码的卡。可用 load_key()来分别装入，只有装入后才能使用验证密码函数验证。

5、halt(void);

说明：中止卡操作函数
参数：无
返回：成功则返回 0；
举例：
st=halt();
附加说明：使用 card()函数时，有个 _Mode 参数,如果模式选择为 0(IDLE),则在对卡进行读写操作完毕后，执行 halt(),则该卡进行 halt 模式，只能当该卡离开操作区域并再次进入时读写器才能够再次操作它;如果为 1（ALL），则可以对卡继续操作。

6、readasc(_Adr);
说明：读函数，一次必须读一个块,长度为 16 个字节；
参数说明： _Adr 是一个整数，为所读数据的地址，_Adr=(0-63)
返回：成功则返回所读数据的第一个字符为'0',否则请参考错误值；
从第二个字符开始 16 个字节为该块的数据
举例：
读块 0 中的数据并将数据存放在变量 value 中
value=read(0);

7、readhex(_Adr);
说明：读函数，一次必须读一个块,长度为 16 个字节；
参数说明： _Adr 是一个整数，为所读数据的地址，_Adr=(0-63)
返回：成功则返回所读数据的第一个字符为'0',否则请参考错误值；
从第二个字符开始 32 个字节为该块用 16 进制表示的数据
举例：
读块 0 中的数据并将数据存放在变量 value 中
value=readhex(0);

8、writeasc(_Adr,_Data);
说明：写函数, 一次必须写一个块；
参数说明：_Adr 是一个整数，为所写数据的地址，_Adr=(1-63)
_Data 是一个字符串，存放要写数据
返回：成功则返回 0；
举例：
将存放在字符串 data 中的数据写到块 1 中。
st=write(1,data)

9、writehex(_Adr,_Data)
说明：写函数，一次必须写一个块；
参数说明：
icdev: init 返回的设备描述符；
_Adr: 所写数据的地址（1-63）；
*_Data: 指向存放要写数据的地址,存放 16 个用 16 进制表示的字符,长度为 32。
返回：成功则返回 0；
举例:

st=writehex(1, "ffffffffffffffffffffffffffffffff");

10、incval(_Adr, _Value);
说明：增值函数
参数说明：_Adr 是一个整数，为所增值的地址，_Adr=(1-63)
_Value 是一个整数，为要增加的值（长度为 4 字节）
返回：成功则返回 0；
举例：
块 1 增加数值 value;
st=incval(1,value);

11、decval(_Adr, _Value);
说明：减值函数
参数说明：_Adr 是一个整数，为要减值的地址，_Adr=(1-63)
_Value 是一个整数，为要减少的值（长度为 4 字节）
返回：成功则返回 0；
举例：
块 1 减去数值 value
st=decval(1,value);

12、initval(_Adr, _Value);
说明：初始化值函数
参数说明：_Adr 是一个整数，为要初始化值的地址，_Adr=(1-63)
_Value 是一个整数，为要初始化的值（长度为 4 字节）
返回：成功则返回 0；
举例：
将块 1 的值初始化为 value
st=initval(1,value);
附加说明：作为数值处理的块，是以特殊格式存贮的，所以必须用初始化值函数初始化，初始化之后方可进行读、减、加的处理。

13、readval(_Adr);
说明：读值函数
参数说明： _Adr 是一个整数，为要读值的地址，_Adr=(1-63)
返回：读出值（整数）
举例：
读出块 1 的数值并存放在变量 value 中
value=readval(1);

14、transf(_Adr);
说明：传送函数
参数说明：_Adr 是一个整数，为要传送的地址
返回：无
举例：

将寄存器的内容传送到块 1 中
st=transf(1);
附加说明：见 restore()的附加说明；

15、restore(_Adr);
说明：回传函数，将_Adr 地址的值传入卡的内部 RAM
参数说明：_Adr 是一个整数，为要进行回传的地址
返回：无
举例：
将块 1 的内容回传到寄存器中
st=restore(1);
附加说明：用户可以用此函数将某一块内的数值传入内部 RAM，然后用 transfer() 函数传送到另一块中去，就达到了块与块之间数值传送的目的。

16、loadkey(_Mode,_SecNr,Key);
说明：装入密码函数
参数说明：_Mode 同函数 pass()
_SecNr 同函数 pass()
Key 包含了要写入硬件系统 RAM 中的卡密码
返回：成功则返回 0；
举例：Key=Chr(160)+Chr(161)+Chr(162)+Chr(163)
+Chr(164)+Chr(165)
st=loadkey(0,1,Key);

17、loadkeyh(_Mode,_ SecNr,Key);
说明：装入密码函数
参数说明：_Mode 同函数 pass()
_SecNr 同函数 pass()
Key 包含了要写入硬件系统 RAM 中的卡密码
返回：成功则返回 0；
举例：Key="a0a1a2a3a4a5"
st=loadkeyh(0,1,Key);

18.beep(_Msec);
说明：蜂鸣函数，使读写器发出蜂鸣声；
参数说明：
_Msec:蜂鸣时间的长短，单位是 10 毫秒；
返回：成则返回 0；
举例： st=beep(10);/*鸣叫 100 毫秒*/

19.light(_onoff);
说明：控制信号灯.
参数说明：
_onoff:1 亮灯,0 灭灯；

返回：成功则返回 0；
举例：
st=light(1);/*信号灯亮*/

20.dispstr(str);
说明：数码显示.
参数说明：
str:要显示的字符串
返回：成功则返回 0；
举例：
st=dispstr("12.345");/*显示 1234.5*/

CLIPPER 函数库（.LIB）

1、initcom(com,baud);
说明：初始化串口函数
参数说明： com 是一个整数, 0 表示串口一,1 表示串口二
baud 是一个整数，用来设置通信波特率，0 表示 9600,1 表示 115200
返回：调用成功则返回 0
举例：
st=initcom(0,0);

2、closecom();
说明：恢复串口函数,退出程序时一定要使用该函数.
参数说明：无参数
返回：无
举例：exit();

3、card(_Mode,snr);
说明：寻卡函数，能返回在工作区域内某张卡的序列号
参数说明： _Mode 是一个整数，设置卡的模式，可以是 0(IDLE)或 1(ALL),有关_Mode 的
说明请参阅 halt()函数说明；
snr 是一个字符串
返回：成功则返回 0
举例：
st=card(0,snr);

4、auth(_Mode, _SecNr)
说明：验证某一扇区密码的函数；
参数说明：_Mode 是一个整数，决定验证密码的方法；
_Mode=0: 用 KEYSET0 和 KEYA 验证
_Mode=1: 用 KEYSET1 和 KEYA 验证
_Mode=2: 用 KEYSET2 和 KEYA 验证
_Mode=4: 用 KEYSET0 和 KEYB 验证
_Mode=5: 用 KEYSET1 和 KEYB 验证
_Mode=6: 用 KEYSET2 和 KEYB 验证
_SecNr 是一个整数，为要验证密码的扇区号(0)
返回：成功则返回 0；
举例：
用 A 密码和 0 套密码去验证扇区 0
st=auth(0,0);
附加说明：每张卡上有 A 密码和 B 密码，可根据实际需要确定是否使用 B 密码，这由卡的
存取控制位来决定(由上面的表可参阅)。此外，读写器中可以存放三套密码，以备
读写器有多种用途,用来校验具有不同密码的卡。可用 load_key()来分别装入，
只有装入后才能使用验证密码函数验证。

5、halt(void);
说明：中止卡操作函数
参数：无
返回：成功则返回 0；
举例：
st=halt();
附加说明：使用 card()函数时，有个 _Mode 参数,如果模式选择为 0(IDLE),则在对卡进行读写操作完毕后，执行 halt(),则该卡进行 halt 模式，只能当该卡离开操作区域并再次进入时读写器才能够再次操作它;如果为 1（ALL），则可以对卡继续操作。

6、read(_Adr,data);
说明：读函数，一次必须读一个块,长度为 16 个字节；
参数说明： _Adr 是一个整数，为所读数据的地址，_Adr=(0-63)
data 是一个字符串
返回：成功则返回 0
举例：
st=read(0,data);

7、readhex(_Adr,data);
说明：读函数，一次必须读一个块,长度为 16 个字节；
参数说明： _Adr 是一个整数，为所读数据的地址，_Adr=(0-63)
返回：成功则返回 0
举例：
st=readhex(0,data);

8、write(_Adr,_Data);
说明：写函数, 一次必须写一个块；
参数说明：_Adr 是一个整数，为所写数据的地址，_Adr=(1-63)
_Data 是一个字符串，存放要写数据
返回：成功则返回 0；
举例：
将存放在字符串 data 中的数据写到块 1 中。
st=write(1,data)

9、writehex(_Adr,_Data)
说明：写函数，一次必须写一个块；
参数说明：
icdev: init 返回的设备描述符；
_Adr: 所写数据的地址（1-63）；
*_Data: 指向存放要写数据的地址,存放 16 个用 16 进制表示的字符,长度为 32。
返回：成功则返回 0；
举例:
st=writehex(1, "ffffffffffffffffffffffffffffffff");

10、incval(_Adr, _Value);
说明：增值函数
参数说明：_Adr 是一个整数，为所增值的地址，_Adr=(1-63)
_Value 是一个整数，为要增加的值（长度为 4 字节）
返回：成功则返回 0；
举例：
块 1 增加数值 value;
st=incval(1,value);

11、decval(_Adr, _Value);
说明：减值函数
参数说明：_Adr 是一个整数，为要减值的地址，_Adr=(1-63)
_Value 是一个整数，为要减少的值（长度为 4 字节）
返回：成功则返回 0；
举例：
块 1 减去数值 value
st=decval(1,value);

12、initval(_Adr, _Value);
说明：初始化值函数
参数说明：_Adr 是一个整数，为要初始化值的地址，_Adr=(1-63)
_Value 是一个整数，为要初始化的值（长度为 4 字节）
返回：成功则返回 0；
举例：
将块 1 的值初始化为 value
st=initval(1,value);
附加说明：作为数值处理的块，是以特殊格式存贮的，所以必须用初始化值函数初始化，初始化之后方可进行读、减、加的处理。

13、readval(_Adr,data);
说明：读值函数
参数说明： _Adr 是一个整数，为要读值的地址，_Adr=(1-63)
data 是一个字符串
返回：成功则返回 0；
举例：
value=readval(1,data);

14、loadkey(_Mode,_SecNr,Key);
说明：装入密码函数
参数说明：_Mode 同函数 auth()
_SecNr 同函数 auth()
Key 包含了要写入硬件系统 RAM 中的卡密码
返回：成功则返回 0；
举例：Key=Chr(160)+Chr(161)+Chr(162)+Chr(163)

```
            +Chr(164)+Chr(165)
        st=loadkey(0,1,Key);
```

15、loadkeyh(_Mode,_ SecNr,Key);
说明：装入密码函数
参数说明：_Mode 同函数 auth()
_SecNr 同函数 auth()
Key 包含了要写入硬件系统 RAM 中的卡密码
返回：成功则返回 0；
举例：Key="a0a1a2a3a4a5"

```
        st=loadkeyh(0,1,Key);
```

16.beep(_Msec);
说明：蜂鸣函数，使读写器发出蜂鸣声；
参数说明：
_Msec:蜂鸣时间的长短，单位是 10 毫秒；
返回：成则返回 0；
举例： st=beep(10);/*鸣叫 100 毫秒*/

17.light(_onoff);
说明：控制信号灯.
参数说明：
_onoff:1 亮灯,0 灭灯；
返回：成功则返回 0；
举例：

```
        st=light(1);/*信号灯亮*/
```

- 32 位 Windows 库（.DLL）**函数错误代码**

返回值（负数）	错误类型
0x10（16）	通讯错误
0x11（17）	超时错误
0x20（32）	打开端口错误
0x21（33）	获得端口参数错误
0x22（34）	设置端口参数错误
0x23（35）	关闭端口出错
0x24（36）	端口被占用
0x30（48）	格式错误
0x31（49）	数据格式错误
0x32（50）	数据长度错误
0x40（64）	读错误
0x41（65）	写错误
0x42（66）	无接收错误
0x50（80）	不够减错

- FoxPro for Dos 函数库（.PLB）**特殊函数错误代码**

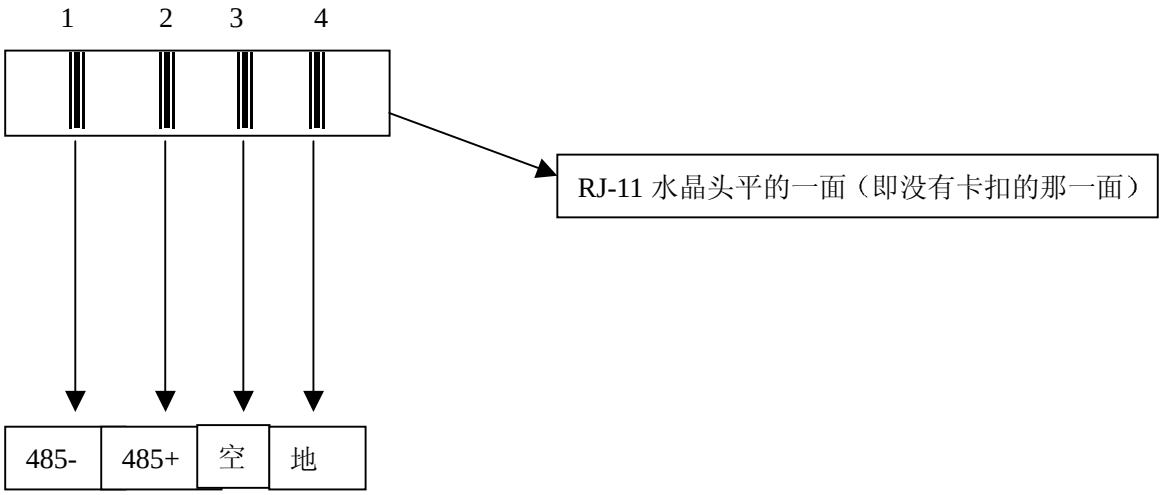
readasc,readhex,card 三个函数返回值是字符串，其中第一个字符表示执行的状态。

返回值	字符	类型
0x2F（47）	‘/’	通讯错误
0x30（48）	‘0’	执行成功
0x31（49）	‘1’	没有卡在范围内
0x34（52）	‘4’	校验错误
0x3A（58）	‘:’	卡没有校验
0x3F（63）	‘?’	写错误
0x40（64）	‘@’	加错误
0x41（65）	‘A’	减错误
0x42（66）	‘B’	读错误
0x4E（78）	‘E’	数据格式错误

- C 语言及 FoxPro for Dos 函数库（.PLB）**普通函数错误代码**

返回值	类型
-1	初始化串口错
0xFF（255）	通讯错误
0x0 （0）	执行成功
0x1 （49）	没有卡在范围内
0x4 （52）	校验错误
0x10（58）	卡没有校验
0x15（63）	写错误
0x16（64）	加错误
0x17（65）	减错误
0x18（66）	读错误
0x1E（30）	数据格式错误

附录一　485 通讯网卡和读写器水晶头 RJ-11 的连接方法



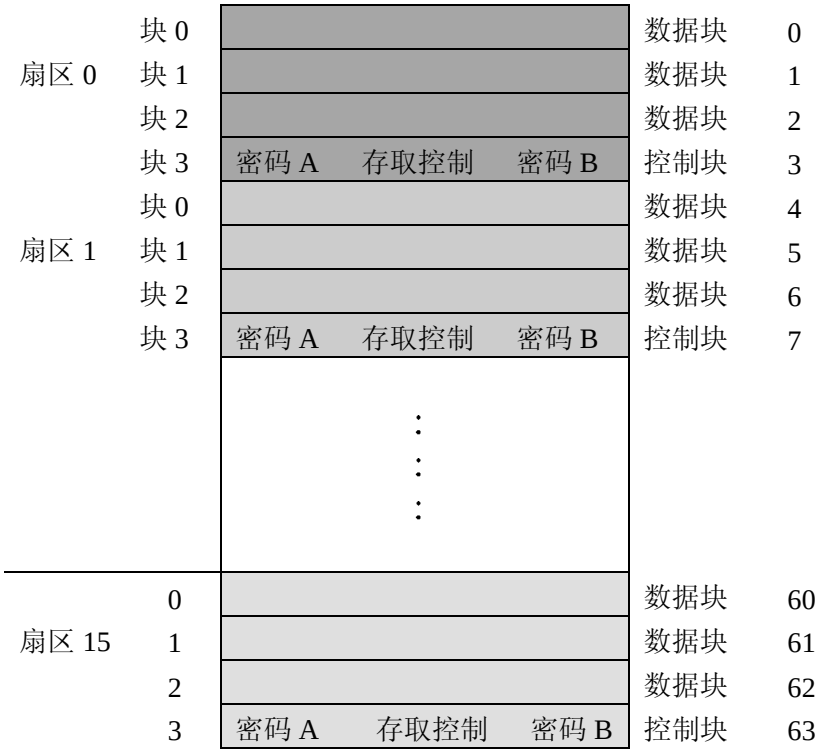
附录二　　非接触式 IC 卡性能简介（M1）

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℃(温度为 90%)
- 工作频率：13.56MHZ
- 通信速率：106KBPS
- 读写距离：10mm 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：



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。