

HCE-300 系列磁卡读写机 程序员手册

二 二年十月

概 述

磁卡的使用已经有很长的历史了。由于磁卡成本低廉，易于使用，便于管理，且具有一定的安全特性，因此它的发展得到了很多世界知名公司，特别各国政府部门几十年的鼎力支持，使得磁卡的应用非常普及，遍布国民生活的方方面面。打电话可以用磁卡，坐飞机检票可以用磁卡，股票市场可以用磁卡，等等，值得一提的是银行系统几十年的普遍推广使用使得磁卡的普及率得到了很大的发展。据资料报道，美国平均每个(成年)人拥有的各类磁卡多达 4 张，新加坡也有类似的普及率。

在美国等一些发达国家，由于磁卡广泛应用于银行、证券等系统，磁卡的应用系统非常完善，如果将已有的这些磁卡应用系统，包括 Visa 卡 / MasterCard 卡应用系统在内，全部换成正在日益成熟的智能卡系统，那么每年的投入至少上千亿美元，并且将严重影响国民的生活使用习惯以及应用系统的正常运转等。这也是智能卡系统在美国的发展远比欧洲国家要慢的原因所在。

在未来很长的一段时间内特别是像美国这样一个银行磁卡应用系统高度发达的国家，银行磁卡应用系统将同智能卡应用系统以互补方式共同存在。

智能卡的总体安全保密性比磁卡的确要好，但是非常完善的磁卡应用系统（例如银行系统）弥补了磁卡本身在其安全保密特性上所存在的不足，因此对使用者来说并不会明显体会两种卡的安全特性有差异及影响使用等。

我公司生产的 HCE-300 系列磁卡读写机可联接任何具有 RS-232 串口的电脑或终端，用于各种介质的磁卡或存折本，包括透明介质的磁条信息。该系列磁卡读写机操作编程简单，读写均一次刷卡完成，具有读、写双重校验功能。读写状态有灯光、声响双重提示功能。该产品性能稳定可靠，并且兼容性好（能同时兼容国内磁卡读写机厂家的命令集），是计算机系统理想的外围设备。可广泛用于金融、邮电、交通、海关等各个领域，特别是银行系统的信用卡、磁卡和存折的读写。

HCE-300 系列磁卡读写机技术指标

1. 拉卡速度：10 ~ 120cm/s
2. 记录格式：兼容 IBM、ISO 格式，可用控制命令切换。
3. 记录密度：第 1 轨 210BPI，最多 79 个字符。
第 2 轨 75BPI/210BPI 可选，最多 37/107 个字符。
第 3 轨 210BPI，最多 107 个字符。
4. 串行通讯参数：波特率：9600bps；数据格式：8 位无校验；1 位起始位；1 位停止位。
5. 磁头寿命：600,000 次。
6. 电源电压：DC 5V ± 5%。
7. 电源电流：200mA。
6. 工作环境：温度：0 ~ 45 湿度：10 ~ 90%RH

磁卡背景知识

磁卡的 ISO 标准

相关的磁卡，特别是应用于银行系统的磁卡的一些 ISO 标准分别为：ISO7810，ISO7811 - 1 至 ISO7811 - 6，ISO7812，ISO7813 以及 ISO15457 等等。

其中：

ISO7810 标准：制定了磁卡的物理特性等；

ISO7812 标准：制定了磁卡的记录技术标准；

ISO781 - 4 标准：制定了磁卡上只读的 Track1 和 Track2 的记录技术标准；

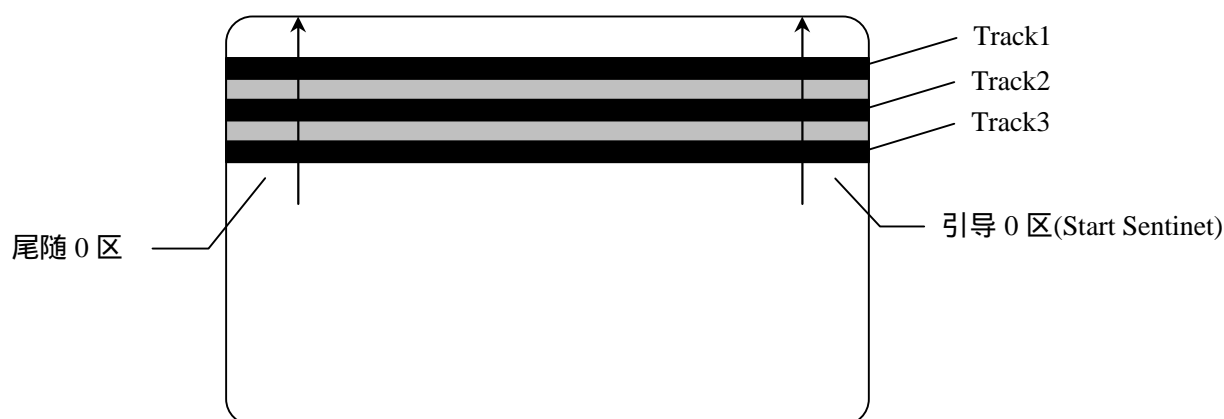
ISO781 - 5 标准：制定了磁卡上可读 / 写的 Track3 的记录技术标准；

ISO15457 标准：制订了磁卡物理标准 / 测试方式 Track 标准 F / 2F 技术标准；

磁卡的物理结构及数据结构

一般而言,应用于银行系统的磁卡上的磁带有3个磁道,分别为Track1,Track2及Track3。每个Track都记录着不同的信息,这些信息有着不同的应用。此外,也有一些应用系统的磁卡只使用了两个磁道(Track),甚至只有一个Track。在我们所设计的应用系统中,根据具体情况,可以使用全部的三个Track或是二个或一个Track。

如图所示是符合ANSI及ISO/IEC标准的磁卡的物理尺寸定义。这些尺寸的定义涉及磁卡读写机具的标准化。因为如果您对磁卡上Track1(或Track2或Track3)进行数据编码时,其数据在磁带上的物理位置偏高或偏低了哪怕几个毫米,则这些已编码的数据信息偏移到了另外的Track上了。



其中：

Track1, 2, 3 的每个磁道宽度相同,大约在 2.80mm (0.11 英寸) 左右,用于存放用户的数据信息;相邻两个 Track 约有 0.05mm (0.02 英寸) 的间隙 (Gap),用于区分相邻的两个磁道;整个磁带宽度在 10.29 毫米 (0.405 英寸) 左右 (如果是应用 3 个 Track 的磁卡),或是在 6.35 毫米 (0.25 英寸) 左右 (如果是应用 2 个 Track 的磁卡)。

实际上我们所接触看到的银行磁卡上的磁带宽度会加宽 1~2mm 左右,磁带总宽度在 12~13mm 之间。

在磁带上,记录 3 个有效磁道数据的起始数据位置和终结数据位置不是在磁带的边缘,而是在磁带边缘向内缩减约 7.44mm (0.293 英寸) 为起始数据位置 (引导 0 区);在磁带边缘向内缩减约 6.93mm (0.273 英寸) 为终止数据位置 (尾随 0 区);这些标准是为了有效保护磁卡上的数据不易被丢失。因为磁卡边缘上的磁记录数据很容易因物理磨损而被破坏。

磁道 Track 上的标准定义

磁道 Track 的应用分配一般是根据特殊的使用要求而定制的,比如银行系统、证券系统、门禁控制系统、身份识别系统、驾驶员驾驶执照管理系统等等,都会对磁卡上的磁卡上的 3 个 Track 提出不同的应用格式要求提出不同的应用格式要求。在此,我们将主要研讨的是符合国际流通的银行/财政应用系统的银行磁卡上的 3 个 Track 的标准定义,这些定义也已经广泛适用于 Visa 信用卡、MasterCard 信用卡等我们常用的一些银行卡。

磁道 Track1: 它的数据标准制定最初是由“国际航空运输协会”IATA (International Air Transportation Association) 完成的。Track1 上的数据和字母记录了航空运输中的自动化信息,例如货物标签信息、交易信息、机票定票/定座情况,等等。这些信息由专门的磁卡读写机具进行数据读写处理,并且在航空公司中有一套应用系统为此服务。应用系统包含了一个数据库,所有这些磁卡的数据信息都可以在此找到记录。

磁道 Track2: 它的数据标准制定最初是由“美国银行家协会”ABA (American Bankers Association) 完成的。该磁道上的信息已经被当今很多的银行系统所采用。它包含了一些最基本的相关信息,例如卡的唯一识别号码、卡的有效期等。

磁道 Track3: 它的数据标准制定最初是由财政行业 (THRIFT) 完成的。其主要应用于一般的储蓄、

贷款和信用单位等那些需要经常对磁卡数据进行更改、重写的场合。典型的应用包括现金售货机、预付费卡（系统）、借贷卡（系统）等等。这一类的应用很多都是处于“脱机”（off line）的模式，即银行（验证）系统很难实时对磁卡上的数据进行跟踪，表现为用户卡上磁道上 Track3 的数据与银行（验证）系统所记录的当前数据不同。

磁道（Track1，Track2，Track3）上允许使用的数字和字符

磁卡上的 3 个 Track 一般都是使用“位”（bit）方式来编码的。根据数据所在的 Track 不同，5 个 bit 或 7 个 bit 组成一个字节。

Track1（IATA）：记录密度为 210BPI；可以记录 0~9 数字及 A~Z 字母等；总共可以记录多达 79 个数字或字符（包含起始结束符和校验符）；每个字符（一个字节）由 7 个 bit 组成。

由于 Track1 上的信息不仅可以用数字 0~9 来表示，还能用字母 A~Z 来表示信息，因此 Track1 上信息一般记录了磁卡的使用类型、范围等一些“标记”性、“说明”性的信息。例如银行用卡中，Track1 记录了用户的姓名，卡的有效使用期限以及其他的一些“标记”信息。

Track2（ABA）：记录密度为 75BPI；可以记录 0~9 数字，不能记录 A~Z 字母；总共可以记录多达 40 个数字（包含起始结束符和校验符）；每个数据（一个字节）由 5 个 bit 组成。

Track3（THRIFT）：记录密度为 210BPI；可以记录 0~9 数字，不能记录 A~Z 字母；总共可以记录多达 107 个数字或字符（包含起始结束符和校验符）；每个字符（一个字节）由 5 个 bit 组成。

由于 Track2 和 3 上的信息只能用数字 0~9 等来表示，不能用字母 A~Z 来表示信息，因此在银行用卡中，Track2，3 一般用以记录用户的帐户信息、款项信息等等，当然还有一些银行所要求的特殊信息等。

在实际的应用开发中，如果我们希望在 Track2 或 3 中表示数字以外的信息，例如“ABC”等，一般应采用按照国际标准的 ASCII 表来映射。例如，要记录字母“A”在 Track2 或 3 上时，则可以用“A”的 ASCII 值“0x41”来表示。“0x41”可以在 Track2 或是 Track3 中用两个数据来表示：“4”和“1”，即“0101”和“0001”。

HCE-300 系列磁卡读写机

Win32 动态链接库函数说明

版本 1.10

函数索引

函数名称	功能简述	页次
HCE300_Open	打开磁卡读写机进行读写磁卡操作	
HCE300_Close	关闭磁卡读写机	
HCE300_Reset	复位磁卡读写机	
HCE300_Read	读取磁卡上指定磁道数据	
HCE300_Write	将指定磁道数据写入磁卡	
HCE300_GetLastStatus	取得上次读写操作结果	
HCE300_SetTrack2Denisty	设置磁道 2 记录密度	
HCE300_GetTrack2Denisty	取得磁道 2 记录密度	
HCE300_SetRecordFormat	设置磁道数据记录格式	
HCE300_GetRecordFormat	取得磁道数据记录格式	
HCE300_SetDeviceType	设置磁卡读写机型号	
HCE300_GetDeviceType	取得磁卡读写机型号	
HCE300_SetShowDialog	设置是否显示操作对话框	

函数说明

关于动态库的重要说明

函数库位于安装目录的 APILIB 子目录下，包含有以下文件：

PComm.DLL	COM 端口操作动态链接库
HCE300_API.DLL	HCE300 磁卡读写机操作动态链接库
HCE300_API.Pas	HCE300 磁卡读写机动态库在 Delphi 中的声明文件
HCE300_API.H	HCE300 磁卡读写机动态库在 C/C++ 中的声明文件
HCE300_API.Bas	HCE300 磁卡读写机动态库在 Visual Basic 中的声明文件
HCE300_API.Prg	HCE300 磁卡读写机动态库在 Visual FoxPro 中的声明文件
HCE300_API.PB	HCE300 磁卡读写机动态库在 PowerBuilder 中的声明文件

其中在最后应用系统发行，必须包含以上两个动态链接库，Pcomm.DLL 和 HCE300_Api.DLL。

关于动态库函数返回值定义的重要说明

返回值	定义	说明
0	HCE300_OK	操作成功
- 1	HCE300_ERROR	操作错误
- 10	HCE300_NOOPEN	设备未打开
- 11	HCE300_ALREADYOPEN	设备已经打开
- 12	HCE300_NOCONNECT	磁卡读写机未联机
- 13	HCE300_ERRPARAMETER	错误的调用参数
- 14	HCE300_TIMEOUT	操作已经超时（对于 HCE300_Open 函数超时时间为 10 秒，对于 HCE300_Read/HCE300_Write 函数超时时间为 120 秒）。
- 15	HCE300_USERBREAK	用户中断
- 16	HCE300_USERESCAPE	用户按下 ESC 键中断操作
- 17	HCE300_EMPTYDATA	空的输入缓冲区
- 100	HCE300_BUSY	当前设备正忙

注意：以上返回值均为长整形（LongInt）。

HCE300_Open

功能描述：

打开 HCE-300 磁卡读写机，以便进行读写等操作，这时 COM 口的通讯参数设置为 9600Bps，8 位数据位，1 位停止位，无校验位。

函数原型：

编程语言	语法
Delphi	function HCE300_Open(ComPort:Integer):Integer;
C/C++	int WINAPI HCE300_Open(int ComPort);
Visual Basic	Function HCE300_Open(ByVal ComPort As Long) As Long
Visual Foxpro	DECLARE INTEGER HCE300_Open In HCE300_Api INTEGER ComPort
PowerBuilder	Function Long HCE300_Open(Long ComPort) Library "HCE300_API.DLL"

参数说明：

ComPort：表示磁卡读写机连接在主机的哪个 COM 口上，取值范围 1~9，即 COM1 ~ COM9。

返回值：

HCE300_OK(0)：磁卡读写机打开成功。

HCE300_ERROR(-1)：与磁卡读写机通讯握手失败。（可能磁卡读写机插头没有插好）

HCE300_ALREADYOPEN(-10)：磁卡读写机已经打开。

HCE300_ERRPARAMETER(-13)：输入参数错误。

示例：

Delphi：

```
var ComPort: Integer;
ComPort := 1; { 从 COM1 口打开磁卡读写机 }
if HCE300_Open(ComPort) <> HCE300_OK then
begin
    { 打开失败 }
end;
```

C/C++：

```
int ComPort;
ComPort = 1;
if(HCE300_Open(ComPort) != HCE300_OK) {
    /* 打开失败 */
}
```

Visual Basic：

```
Dim ComPort As Long
ComPort = 1
If HCE300_Open(ComPort) <> HCE300_OK Then
    ' 打开失败
End If
```

Visual Foxpro：

```
? ComPort = 1
IF HCE300_Open(ComPort) <> 0 THEN
    * 打开失败
ENDIF
```

HCE300_Close

功能描述：

关闭 HCE-300 磁卡读写机。

函数原型：

编程语言	语法
Delphi	function HCE300_Close:Integer;
C/C++	int WINAPI HCE300_Close(void);
Visual Basic	Function HCE300_Close() As Long
Visual Foxpro	DECLARE INTEGER HCE300_Close IN HCE300_Api
PowerBuilder	Function Long HCE300_Close() Library "HCE300_API.DLL"

参数说明：

无

返回值：

HCE300_OK(0)：成功关闭磁卡读写机。

HCE300_NOOPEN(-10)：关闭操作失败。(没有正确打开磁卡读写机)

示例：

Delphi：

```
var ComPort:Integer;
```



```
ComPort := 1;
...
HCE300_Close();
```

C/C++:

```
int ComPort;
ComPort = 1;
...
HCE300_Close();
```

Visual Basic:

```
Dim ComPort As Long
ComPort = 1
...
HCE300_Close()
```

Visual Foxpro:

```
? ComPort = 1
? HCE300_Close()
```

HCE300_Reset

功能描述：

复位磁卡读写机。

函数原型：

编程语言	语法
Delphi	function HCE300_Reset(ResetWay:Integer):Integer;
C/C++	int WINAPI HCE300_Reset(int ResetWay);
Visual Basic	Function HCE300_Reset(ByVal ResetWay As Long) As Long
Visual Foxpro	DECLARE INTEGER HCE300_Reset IN HCE300_Api INTEGER ResetWay
PowerBuilder	Function Long HCE300_Reset(Long ResetWay) Library "HCE300_API.DLL "

参数说明：

ResetWay：复位方式，HCE300_HARD_RESET(0)表示进行硬复位（模拟重新上电过程）；
HCE300_SOFT_RESET(1)表示进行软复位；
HCE300_INIT_RESET(2)表示进行软复位并且进行初始化（设置磁道2密度、设置记录格式）。

返回值：

HCE300_OK(0)：磁卡读写机复位成功。
HCE300_NOOPEN(-10)：关闭操作失败。（没有正确打开磁卡读写机）
HCE300_ERRPARAMETER(-13)：输入参数错误。

示 例：

Delphi:

```
{ Open Success }
HCE300_Reset(HCE300_INIT_RESET);
```

C/C++:

```
/* Open Success
HCE300_Reset(HCE300_INIT_RESET);
```

Visual Basic:

```
' Open Success
HCE300_Reset(HCE300_INIT_RESET)
```

Visual Foxpro:

* Open Success

? HCE300_Reset(HCE300_INIT_RESET)

HCE300_Read

功能描述：

从磁卡读写机读取指定磁道的数据，读取的数据在 ReadData 中，ReadData 缓冲区必须大于 300 字节。若分配缓冲过小，可能引起内存存取错误。

函数原型：

编程语言	语法
Delphi	function HCE300_Read(TrackNo:Integer; ReadData:PChar):Integer;
C/C++	int WINAPI HCE300_Read(int TrackNo, char *ReadData);
Visual Basic	Function HCE300_Read(ByVal TrackNo As Long, ByRef ReadData As Byte) As Long
Visual Foxpro	DECLARE INTEGER HCE300_Read IN HCE300_Api INTEGER InTrackNo, STRING InReadBuf
PowerBuilder	Function Long HCE300_Read(Long TrackNo, REF String ReadData) Library "HCE300_API.DLL"

参数说明：

TrackNo：指定磁道数。

- 1 表示第一磁道；
- 2 表示第二磁道；
- 3 表示第三磁道；
- 4 表示第一、二磁道；
- 5 表示第二、三磁道。

ReadData：被读取磁卡的数据，使用‘|’（十六进制 7CH）来分隔。

返回值：

- HCE300_OK(0)：读取成功；
- HCE300_ERROR(-1)：读取失败；
- HCE300_NOOPEN(-10)：磁卡读写机没有打开；
- HCE300_ERRPARAMETER(-13)：输入参数错误；
- HCE300_TIMEOUT(-14)：等待读卡超时(最长等待读取时间为 120 秒,120 秒后函数返回操作超时错误)；
- HCE300_USERBREAK(-15)：用户中断读取操作；
- HCE300_USERESCAPE(-16)：用户使用 ESC 键中断读取操作。

示 例：

Delphi:

```
var ReadBuf: array[0..250] of Char;
{ Open Success }
if HCE300_Read(2, ReadBuf) = HCE300_OK then
  { 读取磁道 2 成功，数据在 ReadBuf 中 }
else
  { 读取磁道 2 失败 } ;
```

C/C++:

```
char ReadBuf[250];
/* Open Success */
if(HCE300_Read(2, ReadBuf) == HCE300_OK) {
  /* 读取磁道 2 成功，数据在 ReadBuf 中 */
```

```

    }
    else {
        /* 读取磁道 2 失败 */
    }
}

```

Visual Basic:

```

Dim ReadBuf(0 To 250) As Byte;
' Open Success
If HCE300_Read(2, ReadBuf(0)) = HCE300_OK Then
    ' 读取磁道 2 成功，数据在 ReadBuf 中
Else
    ' 读取磁道 2 失败
End If

```

Visual Foxpro:

```

* Open Success
? ReadBuf = "
IF HCE300_Read(2, @ReadBuf) = HCE300_OK THEN
    * 读取磁道 2 成功，数据在 ReadBuf 中
ELSE
    * 读取磁道 2 失败
ENDIF

```

HCE300_Write

功能描述：

将 WriteData 缓冲区中数据写入到磁卡的指定磁道。

函数原型：

编程语言	语法
Delphi	function HCE300_Write(TrackNo: Integer; WriteData: PChar): Integer;
C/C++	int WINAPI HCE300_Write(int TrackNo, Char *WriteData);
Visual Basic	Function HCE300_Write(ByVal TrackNo As Long, ByRef WriteData As Byte) As Long
Visual Foxpro	DECLARE INTEGER HCE300_Write IN HCE300_Api INTEGER InTrackNo, STRING InWriteData
PowerBuilder	Function Long HCE300_Write(Long TrackNo, String WriteData) Library "HCE300_API.DLL"

参数说明：

TrackNo：指定磁道数。

- 1 表示第一磁道；
- 2 表示第二磁道；
- 3 表示第三磁道；
- 4 表示第一、二磁道；
- 5 表示第二、三磁道。

WriteData：将被写入磁卡的数据，不同磁道的数据使用‘|’（十六进制 7CH）来分隔。

返回值：

HCE300_OK(0)：写卡成功；
HCE300_ERROR(-1)：写卡失败；
HCE300_NOOPEN(-10)：磁卡读写机没有打开；

HCE300_ERRPARAMETER(-13) : 输入参数错误 ;
 HCE300_TIMEOUT(-14) : 等待写卡超时(最长等待读取时间为 120 秒 ,120 秒后函数返回操作超时错误);
 HCE300_USERBREAK(-15) : 用户中断读取操作 ;
 HCE300_USERESCAPE(-16) : 用户使用 ESC 键中断读取操作 ;
 HCE300_EMPTYDATA(-17) : WriteData 为空。

示 例 :

Delphi:

```
var WriteBuf: array[0..250] of Char;
{ Open Success }
StrPCopy(WriteBuf, ' 0123456789 ');
if HCE300_Write(2, WriteBuf) = HCE300_OK then
  { 写卡成功 }
else
  { 写卡失败 } ;
```

C/C++:

```
char WriteBuf[250];
/* Open Success */
StrCopy(WriteBuf, " 0123456789 ");
if(HCE300_Write(2, WriteBuf) == HCE300_OK) {
  /* 写卡成功 */
}
else {
  /* 写卡失败 */
}
```

Visual Basic:

```
Dim WriteBuf(0 to 250) As Byte
' Open Success
if HCE300_Write(2, WriteBuf(0)) = HCE300_OK Then
  ' 写卡成功
Else
  ' 写卡失败
End If
```

Visual Foxpro:

```
? WriteBuf = " 0123456789 "
* Open Success
IF HCE300_Write(2, WriteBuf) = HCE300_OK THEN
  ' 写卡成功
ELSE
  ' 写卡失败
ENDIF
```

HCE300_GetLastStatus

功能描述 :

读取上次读写操作后的状态。

函数原型 :

编程语言 语法

Delphi	function HCE300_GetLastStatus: Integer;
C/C++	int WINAPI HCE300_GetLastStatus(void);
Visual Basic	Function HCE300_GetLastStatus() As Long
Visual Foxpro	DECLARE INTEGER HCE300_GetLastStatus IN HCE300_Api AS HCE300_GetLStatus
PowerBuilder	Function Long HCE300_GetLastStatus() Library "HCE300_API.DLL "

参数说明：**返回值：**

HCE300_OK(0)：上次读写卡成功；
HCE300_ERROR(-1)：上次读写卡失败；
HCE300_NOOPEN(-10)：磁卡读写机没有打开；

示 例：**Delphi：**

```
{ Open Success }
if HCE300_GetLastStatus = HCE300_OK then
  { 上次读写操作成功 }
else
  { 上次读写操作失败 } ;
```

C/C++：

```
/* Open Success
if (HCE300_GetLastStatus == HCE300_OK) {
  /* 上次读写操作成功 */
}
else {
  /* 上次读写操作失败 */
}
```

Visual Basic：

```
' Open Success
If HCE300_GetLastStatus() = HCE300_OK Then
  ' 上次读写操作成功
Else
  ' 上次读写操作失败
End If
```

Visual Foxpro：

```
* Open Success
IF HCE300_GetLStatus() = HCE300_OK THEN
  * 上次读写操作成功
ELSE
  * 上次读写操作失败
ENDIF
```

HCE300_SetTrack2Density**功能描述：**

设置第二磁道的写入密度，有 75BPI 和 210BPI 两种密度。

函数原型：

编程语言	语法
Delphi	function HCE300_SetTrack2Density(InBPI: Integer): Integer;

C/C++	int WINAPI HCE300_SetTrack2Density(int InBPI);
Visual Basic	Function HCE300_SetTrack2Density(ByVal InBPI As Long) As Long
Visual Foxpro	DECLARE INTEGER HCE300_SetTrack2Density IN HCE300_Api AS HCE300_SetTK2BPI INTEGER InBPI
PowerBuilder	Function Long HCE300_SetTrack2Density(Long InBPI) Library "HCE300_API.DLL"

参数说明：

InBPI：第 2 磁道的记录密度有 75BPI 和 210BPI 之分。当设置成 75BPI 时，第 2 磁道可记录 37 个字符；当设置成 210BPI 时，第 2 磁道可记录 104 个字符。请注意，当设置成高密 210BPI 时，可能你现有的磁卡阅读设备不支持而不能正确读取，请向你的设备供应商询问第 2 磁道是否支持高密方式。

TRACK2_75BPI(0)：设置第 2 磁道为 75BPI；

TRACK2_210BPI(1)：设置第 2 磁道为 210BPI。

返回值：

HCE300_OK(0)：设置成功；

HCE300_NOOPEN(-10)：磁卡读写机没有打开；

HCE300_ERRPARAMETER(-13)：输入参数错误。

示 例：

Delphi:

```
{ Open Success }
HCE300_SetTrack2Density(TRACK2_75BPI);
```

C/C++:

```
/* Open Success
HCE300_SetTrack2Density(TRACK2_75BPI);
```

Visual Basic:

```
' Open Success
HCE300_SetTrack2Density(TRACK2_75BPI)
```

Visual Foxpro:

```
* Open Success
? HCE300_SetTK2BPI(TRACK2_75BPI)
```

HCE300_GetTrack2Density

功能描述：

取得第二磁道的写入密度设置值。

函数原型：

编程语言	语法
Delphi	function HCE300_GetTrack2Density: Integer;
C/C++	int WINAPI HCE300_GetTrack2Density(void);
Visual Basic	Function HCE300_GetTrack2Density() As Long
Visual Foxpro	DECLARE INTEGER HCE300_GetTrack2Density IN HCE300_Api AS HCE300_GetTK2BPI
PowerBuilder	Function Long HCE300_GetTrack2Density() Library "HCE300_API.DLL"

参数说明：**返回值：**

TRACK2_75BPI(0)：设置第 2 磁道为 75BPI；

TRACK2_210BPI(1)：设置第 2 磁道为 210BPI；

HCE300_NOOPEN(-10)：磁卡读写机没有打开；

示 例：

Delphi:

```
{ Open Success }
if HCE300_GetTrack2Density = TRACK2_75BPI then
  { 当前磁道 2 设置为 75BPI ( 低密 ) }
else
  { 当前磁道 2 设置为 210BPI ( 高密 ) } ;
```

C/C++:

```
/* Open Success
if(HCE300_GetTrack2Density() == TRACK2_75BPI) {
  /* 当前磁道 2 设置为 75BPI ( 低密 ) */
}
else {
  /* 当前磁道 2 设置为 210BPI ( 高密 ) */
}
```

Visual Basic:

```
' Open Success
If HCE300_GetTrack2Density() = TRACK2_75BPI Then
  ' 当前磁道 2 设置为 75BPI ( 低密 )
Else
  ' 当前磁道 2 设置为 210BPI ( 高密 )
End If
```

Visual Foxpro:

```
* Open Success
IF HCE300_GetTK2BPI() = TRACK2_75BPI THEN
  * 当前磁道 2 设置为 75BPI ( 低密 )
ELSE
  * 当前磁道 2 设置为 210BPI ( 高密 )
ENDIF
```

HCE300_SetRecordFormat

功能描述：

设置写入磁卡的数据格式，有 ISO 格式和 IBM 格式两种。

函数原型：

编程语言	语法
Delphi	function HCE300_SetRecordFormat(InFormat: Integer): Integer;
C/C++	int WINAPI HCE300_SetRecordFormat(int InFormat);
Visual Basic	Function HCE300_SetRecordFormat(ByVal InFormat As Long) As Long
Visual Foxpro	DECLARE INTEGER HCE300_SetRecordFormat IN HCE300_Api AS HCE300_SetFormat INTEGER InFormat
PowerBuilder	Function Long HCE300_SetRecordFormat(Long InFormat) Library "HCE300_API.DLL "

参数说明：

InFormat：第 2 磁道的记录格式，有 ISO 格式和 IBM 格式之分。请注意，当设置成 IBM 格式时，可能你现有的磁卡阅读设备不支持而不能正确读取，请向你的设备供应商询问是否支持 IBM 格式。

RECORDFORMAT_ISO(0)：设置磁道数据记录格式为 ISO 格式；

RECORDFORMAT_IBM(1)：设置磁道数据记录格式为 IBM 格式。

返回值：

HCE300_OK(0)：设置成功；
HCE300_NOOPEN(-10)：磁卡读写机没有打开；
HCE300_ERRPARAMETER(-13)：输入参数错误。

示 例：

Delphi：

```
{ Open Success }
HCE300_SetRecordFormat(RECORDFORMAT_ISO);
```

C/C++：

```
/* Open Success
HCE300_SetRecordFormat(RECORDFORMAT_ISO);
```

Visual Basic：

```
' Open Success
HCE300_SetRecordFormat(RECORDFORMAT_ISO)
```

Visual Foxpro：

```
* Open Success
? HCE300_SetFormat(RECORDFORMAT_ISO)
```

HCE300_GetRecordFormat

功能描述：

取得写入磁卡的数据格式设置值。

函数原型：

编程语言	语法
Delphi	function HCE300_GetRecordFormat:Integer;
C/C++	int WINAPI HCE300_GetRecordFormat(void);
Visual Basic	Function HCE300_GetRecordFormat() As Long
Visual Foxpro	DECLARE INTEGER HCE300_GetRecordFormat IN HCE300_Api AS HCE300_GetFormat
PowerBuilder	Function Long HCE300_GetRecordFormat() Library "HCE300_API.DLL "

参数说明：

返回值：

RECORDFORMAT_ISO(0)：设置磁道数据记录格式为 ISO 格式；
RECORDFORMAT_IBM(1)：设置磁道数据记录格式为 IBM 格式；
HCE300_NOOPEN(-10)：磁卡读写机没有打开；

示 例：

Delphi：

```
{ Open Success }
if HCE300_GetRecordFormat = RECORDFORMAT_ISO then
  { 当前磁道数据记录格式为 ISO 格式 }
else
  { 当前磁道数据记录格式为 IBM 格式 } ;
```

C/C++：

```
/* Open Success
if(HCE300_GetRecordFormat() == RECORDFORMAT_ISO) {
  /* 当前磁道数据记录格式为 ISO 格式 */
}
```



```

else {
    /* 当前磁道数据记录格式为 IBM 格式 */
}

```

Visual Basic:

```

' Open Success
If HCE300_GetRecordFormat() = RECORDFORMAT_ISO Then
    ' 当前磁道数据记录格式为 ISO 格式
Else
    ' 当前磁道数据记录格式为 IBM 格式
End If

```

Visual Foxpro:

```

* Open Success
IF HCE300_GetFormat() = RECORDFORMAT_ISO THEN
    * 当前磁道数据记录格式为 ISO 格式
ELSE
    * 当前磁道数据记录格式为 IBM 格式
ENDIF

```

HCE300_SetDeviceType

功能描述：

设置当前磁卡读写机的型号，此函数针对老版本的 HCE-300 磁卡读写机而设置，新版的 HCE-300 磁卡读写机在打开时会自动识别型号，而旧版的 HCE-300 磁卡读写机没有这个功能，而必须使用此函数设置磁卡读写机的型号，使磁卡读写机正常工作。

函数原型：

编程语言	语法
Delphi	function HCE300_SetDeviceType(InType: Integer): Integer;
C/C++	int WINAPI HCE300_SetDeviceType(int InType);
Visual Basic	Function HCE300_SetDeviceType(ByVal InType As Long) As Long
Visual Foxpro	DECLARE INTEGER HCE300_SetDeviceType IN HCE300_Api AS HCE300_SetDeviceType INTEGER InType
PowerBuilder	Function Long HCE300_SetDeviceType(Long InType) Library "HCE300_API.DLL "

参数说明：

InType：设置 HCE300 磁卡读写机的型号。HCE300 磁卡读写机的型号在底盖标签上。具体可设置的型号参数如下：

```

HCE301(1)：HCE-301 单第 1 磁道；
HCE302(2)：HCE-302 单第 2 磁道；
HCE303(3)：HCE-303 单第 3 磁道；
HCE312(4)：HCE-312 双第 1、2 磁道；
HCE323(5)：HCE-323 双第 2、3 磁道。

```

返回值：

```

HCE300_OK(0)：设置成功；
HCE300_NOOPEN(-10)：磁卡读写机没有打开；
HCE300_ERRPARAMETER(-13)：输入参数错误。

```

示 例：

Delphi:

```

{ Open Success }

```

```

HCE300_SetDeviceType(HCE323);    { 设置当前磁卡读写机型号为 HCE-323(双第 2、3 磁道) }
C/C++:
/* Open Success
HCE300_SetDeviceType(HCE323);    /* 设置当前磁卡读写机型号为 HCE-323(双第 2、3 磁道) */
Visual Basic:
' Open Success
HCE300_SetDeviceType(HCE323)      ' 设置当前磁卡读写机型号为 HCE-323(双第 2、3 磁道)
Visual Foxpro:
* Open Success
? HCE300_SetDeviceType(HCE323)    * 设置当前磁卡读写机型号为 HCE-323(双第 2、3 磁道)

```

HCE300_GetDeviceType

功能描述：

取得当前磁卡读写机的型号。

函数原型：

编程语言	语法
Delphi	function HCE300_GetDeviceType: Integer;
C/C++	int WINAPI HCE300_GetDeviceType(void);
Visual Basic	Function HCE300_GetDeviceType() As Long
Visual Foxpro	DECLARE INTEGER HCE300_GetDeviceType IN HCE300_Api AS HCE300_GetDeviceType
PowerBuilder	Function Long HCE300_GetDeviceType() Library "HCE300_API.DLL "

参数说明：

返回值：

HCE300_UNKNOW(0)：未知型号；
 HCE301(1)：HCE-301 单第 1 磁道；
 HCE302(2)：HCE-302 单第 2 磁道；
 HCE303(3)：HCE-303 单第 3 磁道；
 HCE312(4)：HCE-312 双第 1、2 磁道；
 HCE323(5)：HCE-323 双第 2、3 磁道。
 HCE300_NOOPEN(-10)：磁卡读写机没有打开；

示 例：

Delphi:

```

var DeviceType: Integer;
{ Open Success }
...
DeviceType := HCE300_GetDeviceType;

```

C/C++:

```

int DeviceType;
/* Open Success
...
DeviceType = HCE300_GetDeviceType();

```

Visual Basic:

```

Dim DeviceType As Long
' Open Success
...
DeviceType = HCE300_GetDeviceType()

```

Visual Foxpro:

```
? DeviceType = HCE300_UNKNOW
* Open Success
? DeviceType = HCE300_GetDeviceType()
```

HCE300_SetShowDialog

功能描述：

设置是否在磁卡读写操作中显示内建对话框，在此对话框上有一取消按钮可中断当前的操作。

函数原型：

编程语言	语法
Delphi	function HCE300_SetShowDialog(InFlag: Integer): Integer;
C/C++	int WINAPI HCE300_SetShowDialog(int InFlag);
Visual Basic	Function HCE300_SetShowDialog(ByVal InFlag As Long) As Long
Visual Foxpro	DECLARE INTEGER HCE300_SetShowDialog IN HCE300_Api AS HCE300_SetShowDialog INTEGER InFlag
PowerBuilder	Function Long HCE300_SetShowDialog(Long InFlag) Library "HCE300_API.DLL"

参数说明：

InFlag：设置是否显示操作对话框标志。

0：不显示对话框（用户可按 ESC 键中断当前操作）；

1：显示操作对话框（用户可按 ESC 键或点击对话框上中止按钮中断当前操作）。

返回值：

HCE300_OK(0)：设置操作成功；

示 例：

Delphi:

```
HCE300_SetShowDialog(1);    { 显示操作对话框 }
HCE300_SetShowDialog(0);    { 不显示操作对话框 }
```

C/C++:

```
HCE300_SetShowDialog(1);    /* 显示操作对话框 */
HCE300_SetShowDialog(0);    /* 不显示操作对话框 */
```

Visual Basic:

```
HCE300_SetShowDialog(1)      ' 显示操作对话框
HCE300_SetShowDialog(0)      ' 不显示操作对话框
```

Visual Foxpro:

```
? HCE300_SetShowDialog(1)    * 显示操作对话框
? HCE300_SetShowDialog(0)    * 不显示操作对话框
```

磁卡读写机常规命令集

润兴命令集：

功能		主机发给读写机	读写机返回主机	备注
复位	软复位	Esc 0		
	硬复位	Esc a		
联络		Esc e	Esc y	
取状态字	读写磁道 1			状态字最后一位为 p 表示读写正确;为 q 表示读写错误
	读写磁道 2	Esc j	Esc r p/q	
	读写磁道 1,2			
	读写磁道 3	Esc j	Esc T r p/q	
	读写磁道 2,3	Esc j	Esc B r p/q	
置记录密度		Esc L		置 2 轨为 75BPI
		Esc H		置 2 轨为 210BPI
置记录格式		Esc T 1		置 2,3 轨为 ISO 格式
		Esc T 2		置 2,3 轨为 IBM 格式
读磁道 1	Esc r		正确:Esc s STX 磁道 1 数据?Fs	
			错误:Esc D r q	
读磁道 2	Esc j		正确:Esc s 磁道 2 数据?Fs	
			错误:Esc r q	
读磁道 1,2	Esc D j		正确:Esc s 磁道 2 数据 STX 磁道 1 数据?Fs	
			错误:Esc D r q	
读磁道 3	Esc T j		正确:Esc s A 磁道 3 数据?Fs	
			错误:Esc T r q	
读磁道 2,3	Esc B j		正确:Esc s 磁道 2 数据 A 磁道 3 数据?Fs	
			错误:Esc B r q	
写磁道 1	Esc w STX 磁道 1 数据 Gs Esc \		正确:Esc D r p	
			错误:Esc D r q	
写磁道 2	Esc t 磁道 2 数据 Gs Esc \		正确:Esc r p	
			错误:Esc r q	
写磁道 1,2	Esc w 磁道 2 数据 STX 磁道 1 数据 Gs Esc \		正确:Esc D r p	
			错误:Esc D r q	
写磁道 3	Esc t A 磁道 3 数据 Gs Esc \		正确:Esc T r p	
			错误:Esc T r q	
写磁道 2,3	Esc t 磁道 2 数据 A 磁道 3 数据 Gs Esc \		正确:Esc B r p	
			错误:Esc B r q	

南天命令集：

功能		主机发给读写机	读写机返回主机	备注
复位	软复位	Esc 0		
	硬复位	Esc a		
联络		Esc e	Esc y	
取 状 态 字	读写磁道 1			状态字最后一位为 p 表示读写正确;为 q 表示读写错误
	读写磁道 2	Esc j	Esc r p/q	
	读写磁道 1, 2			
	读写磁道 3	Esc j	Esc T r p/r	
	读写磁道 2, 3	Esc j	Esc B r p/s	
置记录密度		Esc L		置 2 轨为 75BPI
		Esc H		置 2 轨为 210BPI
置记录格式		Esc T 1		置 2, 3 轨为 ISO 格式
		Esc T 2		置 2, 3 轨为 IBM 格式
读磁道 1				磁道数据: 读正确为该磁道数据; 读错误为 (7FH)
读磁道 2		Esc]	Esc s 磁道 2 数据?Fs	
读磁道 1, 2				
读磁道 3		Esc T]	Esc A 磁道 3 数据?Fs	
读磁道 2, 3		Esc B]	Esc s 磁道 2 数据 A 磁道 3 数据?Fs	
写磁道 1				
写磁道 2		Esc t 磁道 2 数据 Gs Esc \		
写磁道 1, 2				
写磁道 3		Esc t A 磁道 3 数据 Gs Esc \		
写磁道 2, 3		Esc t 磁道 2 数据 A 磁道 3 数据 Gs Esc \		

ECM 命令集：

功能		主机发给读写机	读写机返回主机	备注
复位	软复位	Esc 0		
	硬复位	Esc a		
联络		Esc e	Esc y	
取 状 态 字	读写磁道 1			
	读写磁道 2		Esc r p/q	
	读写磁道 1, 2			
	读写磁道 3		Esc T r p/q	
	读写磁道 2, 3		Esc B r p/q	
置记录密度		Esc L		置磁道 2 为 75BPI
		Esc H		置磁道 2 为 210BPI
置记录格式		Esc L		置磁道 2, 3 为 ISO 格式
		Esc H		置磁道 2, 3 为 IBM 格式
读磁道 1				
读磁道 2		Esc r	Esc s 磁道 2 数据?Fs Esc 状态字	读磁道成功返回磁道数据，读失败返回 (7FH)
读磁道 1, 2				
读磁道 3		Esc p	Esc t 磁道 3 数据?Fs Esc 状态字	
读磁道 2, 3		Esc q	Esc s 磁道 2 数据? Esc t 磁道 3 数据?Fs Esc 状态字	
写磁道 1				
写磁道 2		Esc w ESC s 磁道 2 数据?Fs	Esc 状态字	
写磁道 1, 2				
写磁道 3		Esc u ESC t 磁道 3 数据?Fs	Esc 状态字	
写磁道 2, 3		Esc v ESC s 磁道 2 数据?Esc t 磁道 3 数据?Fs	Esc 状态字	

国光命令集：

功能		主机发给读写机	读写机返回主机	备注
复位	软复位	DEL(7FH)	——	
	硬复位			
联络		Esc (1BH)	AK(06H)	
取 状 态 字	读写磁道 1			
	读写磁道 2			
	读写磁道 1, 2			
	读写磁道 3			
	读写磁道 2, 3			
置记录密度				
置记录格式				
读磁道 1				读卡正确，返回数据包；读卡错误，仅返回 NK(15H)
读磁道 2			STX;磁道 2 数据?ETX BCC	
读磁道 2, 3			正确:STX;磁道 2 数据?+ 磁道 3 数据? ETX BCC 错误:NK	
读磁道 3			STX+磁道 3 数据?ETX BCC	
读磁道 1, 2			正确:STX%磁道 1 数据?;磁道 2 数据? ETX BCC 错误:NK	
写磁道 1		STX%磁道 1 数据?ETX BCC	STX%磁道 1 数据?ETX BCC	
写磁道 2		STX;磁道 2 数据?ETX BCC	STX;磁道 2 数据?ETX BCC	
写磁道 2, 3		STX;磁道 2 数据?+ 磁道 3 数据? ETX BCC	STX;磁道 2 数据?+ 磁道 3 数据? ETX BCC	
写磁道 3		STX+磁道 3 数据?ETX BCC	STX+磁道 3 数据?ETX BCC	
写磁道 1, 2		STX%磁道 1 数据?; 磁道 2 数据? ETX BCC	STX%磁道 1 数据?; 磁道 2 数据? ETX BCC	