Advanced Card Systems Ltd.
Card & Reader Technologies

ACR89U-A1
手持式
智能卡读写器

参考手册 V1.05

info@acs.com.hk
www.acs.com.hk

目录

1.0. 简介 4

1.1. 概览 4

2.0. 硬件设计 5

2.1. 体系结构 5
2.2. USB 接口 5
2.3. 通信参数 5
2.4. 端点 6
2.5. 接触式智能卡接口 6
2.5.1. 智能卡电源 VCC（C1） 6
2.5.2. 卡片类型选择 6
2.5.3. CPU 卡接口 6

3.0. ACR89-A1 的 USB 通信协议 7

3.1. 设备配置 7
3.2. CCID 类特定请求 8
3.2.1. 命令汇总 8
3.3. CCID 命令通道 Bulk-Out 消息 9
3.3.1. 命令汇总 9
3.4. CCID 命令通道 Bulk-In 消息 15
3.4.1. 消息汇总 15
3.5. ACR89 兼容的扩展命令通道消息 18
3.5.1. 扩展命令通道 Bulk-OUT 消息 18
3.5.2. 命令详情 19
3.5.3. 扩展命令通道 Bulk-IN 消息 26
3.5.4. 消息详情 26
3.5.5. 扩展命令响应码和 返还状态 29
3.6. CCID Interrupt-IN 消息 30
3.6.1. 消息汇总 30
3.7. CCID 错误码和状态码 31

4.0. 动态链接库（DLL） 32

4.1. ACR89 DLL 的 API 声明 32
4.1.1. 枚举数 32
4.1.2. 读写器命令数据结构 33
4.1.3. 读写器响应数据 38
4.1.4. 读写器共享命令/响应数据结构 39
4.2. ACR89 DLL 的 API 函数 42
4.2.1. 一般描述 42
4.2.2. 端口函数 42
4.2.3. 设备函数 43
4.2.4. LCD 函数 46
4.2.5. 键盘函数 50
4.2.6. 实时时钟函数 53
4.2.7. 其它函数 54

附录 A.错误码(DLL 异常) 56

图目录

图 1 ： ACR89U-A1 架构 5
图 2 ： CCID PC_to_RDR_Escape 消息 18
图 3 ： PC_to_ACR89_DisplayGraphic – 位图结构 21
图 4 ： CCID RDR_to_PC_Escape 消息 26
图 5 ： ACR89 读写器的位图格式 47

表目录

表 1 ： USB 接口配线 5
表 2 ： CCID 错误码和状态码 31
表 3 ： 键盘输入格式 51
表 4 ： DLL 错误码 56

1.0. 简介

本文描述了如何用 ACR89 软件编程接口控制 ACR89 多功能智能卡读写器的内置附件。内置附件包括嵌入 ACR89 的键盘，LCD 屏幕，LED，蜂鸣器和实时时钟。这类组件不受智能卡读写器库控制。

两种方法可以控制 ACR89 外设：

1. PC/SC 直接（Escape）命令

   PC/SC 接口的 SCardControl() 函数发布的直接命令用于封装可控制 ACR89 外设的 CCID 命令消息，

2. 动态链接库（DLL）

   下文中的接口都用 ACR89 DLL 指代。ACR89 DLL 基于 C 语言，兼容 Windows 7, Vista 和 XP 系统。DLL 名字是 acr89.dll，头文件 acr89.h 提供函数（本文中有描述）给应用程序使用。

1.1. 概览

- 小节 3 描述了用于控制外设的 PC/SC 直接命令。还包括用于 CCID 命令消息定义的 ACR89 的 USB 通信协议。
- 小节 4 描述了 ACR89 DLL 的 API 接口，与 Windows 的 PC/SC 子系统无关。DLL 与 ACR89 的内置外设及应用程序通信时不使用任何 PC/SC。

2.0. 硬件设计

2.1. 体系结构

ACR89U 的库架构如下图所示：

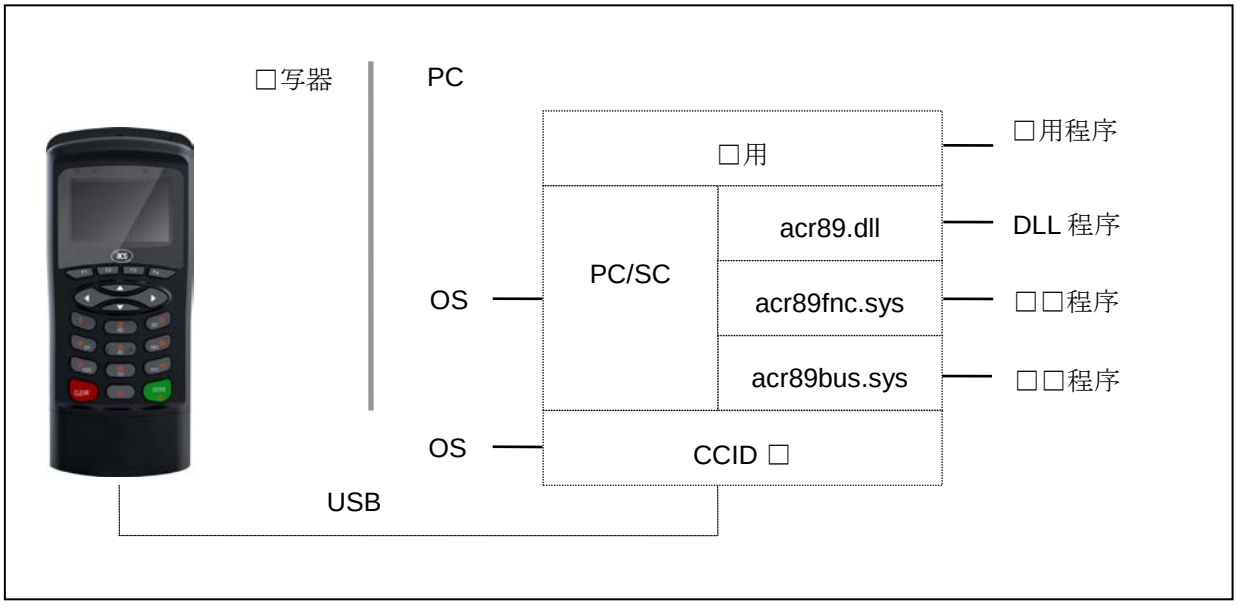

图1：ACR89U-A1 架构

2.2. USB 接口

ACR89U-A1 通过标准 USB 接口与计算机建立连接。

2.3. 通信参数

如《USB 规格书 2.0》所述，ACR89U-A1 通过 USB 连接计算机，支持 USB 全速模式（12 Mbps）。

引脚	信号	功能
1	V_{BUS}	为读写器提供+5 V 的电源
2	D-	ACR89U-A1 和 PC 间以微分信号传输数据
3	D+	ACR89U-A1 和 PC 间以微分信号传输数据
4	GND	参考电压等级

表1：USB 接口配线

注：*为了使 ACR89U-A1 通过 USB 接口正常运行，应该先安装设备驱动程序。*

2.4. 端点

ACR89U-A1 通过以下端点与主机进行通信：

Control Endpoint	设置和控制
Bulk OUT	主机发送至 ACR89U-A1 的命令（数据包为 64 字节）
Bulk IN	ACR89U-A1 发送至主机的命令（数据包为 64 字节）
Interrupt IN	ACR89U-A1 发送至主机的卡片状态报文（数据包为 8 字节）

2.5. 接触式智能卡接口

ACR89U-A1 与智能卡之间的界面符合 ISO 7816-3 标准协议，并进行了某些限制或提升来增强 ACR89U-A1 的实用功能。

2.5.1. 智能卡电源 VCC（C1）

智能卡电流消耗不得大于 50 mA。

2.5.2. 卡片类型选择

激活插入的卡片之前，主控计算机需要向 ACR89U-A1 发送适当的命令来选择卡片类型。

如果 MCU 卡同时支持 T=0 和 T=1，读写器可通过协议与参数选择（PPS）为 MCU 卡选择 T=0 或 T=1 中作为首选协议。如果 MCU 卡仅支持 T=0 或 T=1，读卡器会自动采用该协议类型，而不管应用程序选择哪一种。

2.5.3. CPU 卡接口

CPU 卡只使用触点 C1（VCC）、C2（RST)、C3（CLK）、C5（GND）和 C7（I/O）。时钟信号（C3）的频率为 4.8 MHz。

3.0. ACR89-A1 的 USB 通信协议

ACR89-A1 用 USB 连接主机（连机模式下）。现在的行业内规范 — CCID 标准，已经为 USB 芯片-智能卡接口设备定义了与此相关的协议。CCID 涵盖了操作智能卡和 PIN 所需的全部协议。然而，它并没有定义操作 ACR89-A1 其他外设特性的协议。ACR89U-A1 的通信协议须遵循 CCID 标准并可扩展支持读写器其他特性。

3.1. 设备配置

ACR89U-A1 的 USB 端点配置和使用符合 CCID 标准（1.1 版本）第 4 部分的规定。概述如下：

1. 控制命令通过控制通道（缺省通道）发送，其中包括类特定请求和 USB 标准请求。由缺省通道发送的命令会通过缺省通道向主机反馈信息。
2. CCID 事件通过中断通道发送。
3. **CCID 命令**经由 BULK-OUT 端点发出。发送至 ACR89 的每个命令都有一个相关的最终响应，一些命令也有过程响应。
4. **CCID 响应**经由 BULK-IN 端点发出。所有发送至 ACR89 的命令都必须同步发送（即：对于 ACR89 来说，bMaxCCIDBusySlots 等于 1）。

ACR89 支持的 CCID 功能由其类别描述符定义：

偏移	字段	大小	值	说明
0	*bLength*	1	36h	描述符的字节数
1	*bDescriptorType*	1	21h	CCID 功能描述符的类别
2	*bcdCCID*	2	0100h	CCID 以二进制编码的十进制指定版本号码
4	*bMaxSlotIndex*	1	04h	5 个插槽
5	*bVoltageSupport*	1	07h	支持 1.8V、3.0V 和 5.0 V 的槽位电压
6	*dwProtocols*	4	00000003h	支持 T=0 和 T=1 协议
10	*dwDefaultClock*	4	000012C0h	默认 ICC 时钟频率为 4.8 MHz
14	*dwMaximumClock*	4	000012C0h	ICC 支持的最大时钟频率为 4.8 MHz
18	*bNumClockSupported*	1	00h	不支持手动设置时钟频率
19	*dwDataRate*	4	003267h	默认 ICC I/O 数据传输速率为 12,903 bps
23	*dwMaxDataRate*	4	00032673h	ICC I/O 支持的最大数据传输速率为 206,451 bps
27	*bNumDataRatesSupported*	1	00h	不支持手动设置数据传输速率
28	*dwMaxIFSD*	4	00000FEh	T=1 协议下，最大 IFSD 是 254
32	*dwSynchProtocols*	4	00000000h	不支持同步卡
36	*dwMechanical*	4	00000000h	不支持特殊机械特性

偏移	字段	大小	值	说明
40	*dwFeatures*	4	000204B2h	ACR89 有以下特性： • 根据 ATR 数据自动配置参数 • 根据参数自动改变 ICC 时钟频率 • 根据频率和 FI、DI 参数自动改变波特率 • 根据当前参数自动进行 PPS • 自动 IFSD • 与 ACR1281U-K1 进行短 APDU 级交换
44	*dwMaxCCIDMessageLength*	4	00000110h	最大信息长度为 272 字节
48	*bClassGetResponse*	1	FFh	Get Response 命令中 APDU 回应级别
49	*bClassEnvelope*	1	FFh	无意义（短 APDU 级交换）
50	*wLCDLayout*	2	0815h	LCD，8 行 x 21 个字符
52	*bPINSupport*	1	03h	支持 PIN 码验证和修改
53	*bMaxCCIDBusySlots*	1	01h	同一时间内只能有 1 个槽位处于工作状态

注： *CCID 标准采用小端格式。*

3.2. CCID 类特定请求

基于 ACR89 的命令消息结构标准,ACR89 通过 USB 与计算机进行通讯。ACR89 支持通过控制通道发送的 CCID 类特定请求。

3.2.1. 命令汇总

停止当前运行的任意命令，使 ACR89 可执行新命令：

bmRequestType	bRequest	wValue	wIndex	wLength	数据
00100001Bh	ABORT (01h)	bSeq, bSlot	接口	0000h	无

3.3. CCID 命令通道 Bulk-Out 消息

ACR89 遵循 CCID 协议(1.1 版本)第 6.1 部分有关 Bulk-OUT 消息的规定。CCID 还定义了操作附加功能的扩展命令。此节列举了 ACR89 支持的 CCID 类 Bulk-OUT 消息。**小节 3.5** 将介绍扩展命令。

3.3.1. 命令汇总

3.3.1.1. PC_to_RDR_IccPowerOn

激活插槽并返回卡片的 ATR。

偏移	字段	大小	值	说明
0	*bMessageType*	1	62h	-
1	*dwLength*	4	00000000h	此消息的额外字节的大小
2	*bSlot*	1	-	标识命令的插槽号
5	*bSeq*	1	-	命令的序号
6	*bPowerSelect*	1	-	ICC 上的电压值 00h – 自动电压选择 01h - 5 V 02h - 3 V 03h - 1.8 V
7	*abRFU*	2	-	保留为将来使用

此消息的响应是 RDR_to_PC_DataBlock 消息，返回的数据是复位应答（ATR）。

3.3.1.2. PC_to_RDR_IccPowerOff

取消激活插槽。

偏移	字段	大小	值	说明
0	*bMessageType*	1	63h	-
1	*dwLength*	4	00000000h	此消息的额外字节的大小
5	*bSlot*	1	-	标识命令的插槽号
6	*bSeq*	1	-	命令的序号
7	*abRFU*	3	-	保留为将来使用

此消息的响应是 RDR_to_PC_SlotStatus 消息。

3.3.1.3. PC_to_RDR_GetSlotStatus

获取当前的插槽状态。

偏移	字段	大小	值	说明
0	*bMessageType*	1	65h	-
1	*dwLength*	4	00000000h	此消息的额外字节的大小

偏移	字段	大小	值	说明
5	*bSlot*	1	-	标识命令的插槽号
6	*bSeq*	1	-	命令的序号
7	*abRFU*	3	-	保留为将来使用

此消息的响应是 RDR_to_PC_SlotStatus 消息。

3.3.1.4. PC_to_RDR_XfrBlock

向 ICC 传输数据块。

偏移	字段	大小	值	说明
0	*bMessageType*	1	6Fh	-
1	*dwLength*	4	-	此消息的 abData 字段的大小
5	*bSlot*	1	-	标识命令的插槽号
6	*bSeq*	1	-	命令的序号
7	*bBWI*	1	-	用于延长当前传输的 CCID 块超时等待时间。“该数值乘以块等待时间”的时间段过去后，CCID 将设置该块超时。
8	*wLevelParameter*	2	0000h	短 APDU 级，保留为将来所用
10	*abData*	字节型数组	-	发送给 CCID 的数据块。信息是“按原样”发送至 ICC（短 APDU 级）

此消息的响应是 RDR_to_PC_DataBlock 消息。

3.3.1.5. PC_to_RDR_GetParameters

获取插槽参数。

偏移	字段	大小	值	说明
0	*bMessageType*	1	6Ch	-
1	*dwLength*	4	00000000h	此消息的额外字节的大小
5	*bSlot*	1	-	标识命令的插槽号
6	*bSeq*	1	-	命令的序号
7	*abRFU*	3	-	保留为将来使用

此消息的响应是 RDR_to_PC_Parameters 消息。

3.3.1.6. PC_to_RDR_ResetParameters

重置插槽参数为默认值。

偏移	字段	大小	值	说明
0	bMessageType	1	6Dh	-
1	dwLength	4	00000000h	此消息的额外字节的大小
5	bSlot	1	-	标识命令的插槽号
6	bSeq	1	-	命令的序号
7	abRFU	3	-	保留为将来使用

此消息的响应是 RDR_to_PC_Parameters 消息。

3.3.1.7. PC_to_RDR_SetParameters

设置插槽参数。

偏移	字段	大小	值	说明
0	*bMessageType*	1	61h	-
1	*dwLength*	4	-	此消息的额外字节的大小
5	*bSlot*	1	-	标识命令的插槽号
6	*bSeq*	1	-	命令的序号
7	*bProtocolNum*	1	-	指定采用的协议数据结构。 00h = T=0 协议结构 01h = T=1 协议结构 以下值保留为将来使用： 80h：2 线协议结构 81h：3 线协议结构 82h：I2C 协议结构
8	*abRFU*	2	-	保留为将来使用
10	*abProtocolDataStructure*	字节型数组	-	协议数据结构

T=0 的协议数据结构（*dwLength*=00000005h）

偏移	字段	大小	值	说明
10	*bmFindexDindex*	1	-	B7-4 – FI – ISO/IEC 7816-3:1997 中表 7 的索引，选择一个时钟速率转换因子 B3-0 – DI - ISO/IEC 7816-3:1997 中表 8 的索引，选择一个波特率转换因子

偏移	字段	大小	值	说明
11	*bmTCCKST0*	1	-	B0 – 0b, B7-2 – 000000b B1 – 使用的约定（b1=0：正向约定；b1=1：反向约定） 注：CCID 忽略该位。
12	*bGuardTimeT0*	1	-	两个字符间的额外保护时间。在通常的保护时间（12etu）基础上增加 0-254 个 etu。FFh 等同于 00h。
13	*bWaitingIntegerT0*	1	-	T=0 时 WI 用于定义 WWT
14	*bClockStop*	1	-	支持 ICC 时钟停止 00h = 不允许停止时钟 01h = 时钟信号为低时停止 02h = 时钟信号为高时停止 03h = 时钟信号为高或为低时停止

T=1 的协议数据结构(*dwLength*=00000007h)

偏移	字段	大小	值	说明
10	*bmFindexDindex*	1	-	B7-4 – FI – ISO/IEC 7816-3:1997 中表 7 的索引，选择一个时钟速率转换因子 B3-0 – DI - ISO/IEC 7816-3:1997 中表 8 的索引，选择一个波特率转换因子
11	*BmTCCKST1*	1	-	B7-2 – 000100b B0 – 校验和的类型（b0=0：LRC；b0=1：CRC） B1 – 使用的约定（b1=0：正向约定；b1=1：反向约定） 注：CCID 忽略该位。
12	*BGuardTimeT1*	1	-	额外保护时间（两个字符间为 0-254 个 etu）若值为 FFh，则保护时间减少 1 个 etu。
13	*BWaitingIntegerT1*	1	-	B7-4 = BWI 值 0-9 有效 B3-0 = CWI 值 0-Fh 有效
14	*bClockStop*	1	-	支持 ICC 时钟停止 00h = 不允许停止时钟 01h = 时钟信号为低时停止 02h = 时钟信号为高时停止 03h = 时钟信号为高或为低时停止
15	*bIFSC*	1	-	商定的 IFSC 的大小
16	*bNadValue*	1	00h	只支持 NAD = 00h

此消息的响应是 RDR_to_PC_Parameters 消息。

3.3.1.8. PC_to_RDR_Escape

使能小节 **3.5** 定义的 ACR89 的扩展特性。

偏移	字段	大小	值	说明
0	*bMessageType*	1	6Bh	-
1	*DwLength*	4	-	此消息的 abData 字段的大小
5	*Bslot*	1	-	标识命令的插槽号
6	*Bseq*	1	-	命令的序号
7	*AbRFU*	3	-	保留为将来使用
10	*AbData*	字节型数组	-	小节 **3.5.2 定义的命令**

此消息的响应是 RDR_to_PC_Escape 消息。

此消息能返回以下 ACR89 特有的错误。错误的更多信息包含在扩展响应里。

bmICCStatus	bmCommand Status	bError	说明
3	1	ACR89_ERROR	ACR89 特有错误。参见 ACR89 响应的 wReturnCode。
3	1	INVALID_MODE	ACR89 当前运行的模式不支持该命令。
3	1	DEVICE_VOID	ACR89 未初始化。

3.3.1.9. PC_to_RDR_Secure (RFU)

保留为将来所用。

用户可用此命令在智能卡上直接验证或修改 PIN 码。

偏移	字段	大小	值	说明
0	*bMessageType*	1	69h	-
1	*DwLength*	4	-	此消息的额外字节的大小
5	*BSlot*	1	-	标识命令的插槽号
6	*BSeq*	1	-	命令的序号

偏移	字段	大小	值	说明
7	*BBWI*	1	-	用于延长当前传输的 CCID 块超时等待时间。“该数值乘以块等待时间”的时间段过去后，CCID 将设置该块超时。此参数仅用于字符级转换。
8	*wLevelParameter*	2	0000h	短 APDU 级，保留为将来所用
10	*bPINOperation*	1	-	用于表示 PIN 操作： 00h = PIN 码校验 01h = PIN 码修改 02h = 从安全的 CCID 设备传输 PIN 码缓存 03h = 等待 ICC 响应 04h = 取消 PIN 功能 05h = 重新发送上一个 I-Block，只有 T=1 协议在用时有意义。 06h = 发送 APDU 的下一部分，只有 T=1 协议在用时有意义。
11	*abPINDataStructure*	字节型数组	-	PIN 码验证数据结构或 PIN 码更改数据结构

此消息的响应是 RDR_to_PC_DataBlock 消息。

注：*关于 PIN 码验证/更改数据结构的更多细节，请参考 CCID 协议的 6.1.11 部分。*

3.3.1.10. PC_to_RDR_Abort

与控制通道中止的请求联用，通知 CCID 停止在特定插槽进行的传输，使插槽回到可以接受新命令通道 Bulk-OUT 消息的状态。

偏移	字段	大小	值	说明
0	*bMessageType*	1	72h	-
1	*DwLength*	4	00000000h	此消息的额外字节的大小
5	*BSlot*	1	-	标识命令的插槽号
6	*BSeq*	1	-	命令的序号
7	*AbRFU*	3	000000h	RFU

此消息的响应是 RDR_to_PC_SlotStatus 消息。

3.4. CCID 命令通道 Bulk-In 消息

Bulk-IN 消息用于响应 Bulk-OUT 消息。ACR89 遵循 CCID 标准（1.1 版本）第 6.2 部分有关 Bulk-IN 消息的规定。此节列举了 ACR89 支持的 CCID 类 Bulk-IN 消息。

3.4.1. 消息汇总

3.4.1.1. RDR_to_PC_DataBlock

此消息是 ACR89 对 PC_to_RDR_IccPowerOn、PC_to_RDR_XfrBlock 和 PC_to_RDR_Secure 消息的响应。

偏移	字段	大小	值	说明
0	*bMessageType*	1	80h	表示 CCID 正在发送一个数据块.
1	*dwLength*	4	-	此消息的 abData 字段的大小
5	*BSlot*	1	-	与 Bulk-OUT 消息中的值相同
6	*BSeq*	1	-	与 Bulk-OUT 消息中的值相同
7	*bStatus*	1	-	小节 3.7 定义了插槽状态和错误寄存器
8	*bError*	1	-	小节 3.7 定义了插槽状态和错误寄存器
9	*bChainParameter*	1	00h	短 APDU 级，保留为将来所用
10	*AbData*	字节型数组	-	本数据域包含由 CCID 返还的数据

3.4.1.2. RDR_to_PC_SlotStatus

此消息是 ACR89 对 PC_to_RDR_IccPowerOff、PC_to_RDR_GetSlotStatus 和 PC_to_RDR_Abort 消息，以及类特定 ABORT 请求的响应。

偏移	字段	大小	值	说明
0	*bMessageType*	1	81h	-
1	*dwLength*	4	00000000h	消息特定的数据长度。
5	*BSlot*	1	-	与 Bulk-OUT 消息中的值相同
6	*BSeq*	1	-	与 Bulk-OUT 消息中的值相同
7	*bStatus*	1	-	小节 3.7 定义了插槽状态和错误寄存器
8	*bError*	1	-	小节 3.7 定义了插槽状态和错误寄存器

偏移	字段	大小	值	说明
9	*bClockStatus*	1	-	值： 00h = 时钟运行 01h = 时钟停于 L 状态 02h = 时钟停于 H 状态 03h = 时钟停于未知状态 所有其他值保留为将来使用。

3.4.1.3. RDR_to_PC_Parameters

此消息是 ACR89 对 PC_to_RDR_GetParameters、PC_to_RDR_ResetParameters 和 PC_to_RDR_SetParameters 消息的响应。

偏移	字段	大小	值	说明
0	*bMessageType*	1	82h	-
1	*dwLength*	4	-	此消息的 abProtocolDataStructure 字段的大小
5	*bSlot*	1	-	与 Bulk-OUT 消息中的值相同
6	*bSeq*	1	-	与 Bulk-OUT 消息中的值相同
7	*bStatus*	1	-	小节 3.7 定义了插槽状态和错误寄存器
8	*bError*	1	-	小节 3.7 定义了插槽状态和错误寄存器
9	*bProtocolNum*	1	-	指定采用的协议数据结构。 00h：T=0 协议的结构 01h：T=1 协议的结构 以下值保留为将来使用 80h：2 线协议结构 81h：3 线协议结构 82h：I2C 协议结构
10	*abProtocolDataStructure*	字节型数组	-	协议数据结构如 CCID 标准（1.1 版本） 6.2.3 节汇总

3.4.1.4. RDR_to_PC_Escape

此消息是 ACR89 对 PC_to_RDR_Escape 消息的响应。

偏移	字段	大小	值	说明
0	*bMessageType*	1	83h	-
1	*dwLength*	4	-	此消息的 abData 字段的大小
5	*bSlot*	1	-	与 Bulk-OUT 消息中的值相同
6	*bSeq*	1	-	与 Bulk-OUT 消息中的值相同
7	*bStatus*	1	-	小节 3.7 定义了插槽状态和错误寄存器
8	*bError*	1	-	小节 3.7 定义了插槽状态和错误寄存器
9	*bRFU*	1	00h	RFU
10	*abData*	字节型数组	-	ACR89 对不同的扩展命令有不同的响应数据，具体细节如小节 3.5.4 所述。

3.5. ACR89 兼容的扩展命令通道消息

本节描述了 CCID 不涵盖，但 ACR89 兼容的扩展命令，用于操作附加功能。这些命令总是在 PC_to_RDR_Escape Bulk-Out 消息下执行，并响应 RDR_to_PC_Escape Bulk-IN 消息。

PC 请求消息	代码	ACR89 响应消息	代码
PC_to_ACR89_InputKey	12h	ACR89_to_PC_DataBlock	81h
PC_to_ACR89_SetCursor	18h	ACR89_to_PC_DisplayStatus	83h
PC_to_ACR89_SetBacklight	19h	ACR89_to_PC_DisplayStatus	83h
PC_to_ACR89_DisplayMessage	1Bh	ACR89_to_PC_DisplayStatus	83h
PC_to_ACR89_DisplayRowGraphic	23h	ACR89_to_PC_DisplayStatus	83h
PC_to_ACR89_SetContrast	1Ch	ACR89_to_PC_DisplayStatus	83h
PC_to_ACR89_ClearDisplay	1Dh	ACR89_to_PC_DisplayStatus	83h
PC_to_ACR89_ReadRTC	08h	ACR89_to_PC_TimeStamp	84h
PC_to_ACR89_SetRTC	09h	ACR89_to_PC_TimeStamp	84h
PC_to_ACR89_Buzzer	0Ah	ACR89_to_PC_Echo	90h
PC_to_ACR89_AccessEeprom	21h	ACR89_to_PC_Datablock	81h
PC_to_ACR89_SetLED	22h	ACR89_to_PC_Echo	90h
PC_to_ACR89_EraseSPIFlash	30h	ACR89_to_PC_ExMemStatus	B0h
PC_to_ACR89_ProgramSPIFlash	33h	ACR89_to_PC_MemoryStatus	B0h
PC_to_ACR89GetSPIFlash	34h	ACR89_to_PC_MemoryPage	B1h
PC_to_ACR89_GetVersion	36h	ACR89_to_PC_VersionInfo	B2h
PC_to_ACR89_AuthoInfo	38h	ACR89_to_PC_AuthInfo	B4h

3.5.1. 扩展命令通道 Bulk-OUT 消息

本节定义的命令结构 abData 字段用于填充 PC_to_RDR_Escape 消息。

类似于 CCID 消息结构，该命令结构包括固定长度的命令头和长度可变的命令数据部分。命令头固定是 5 个字节。

与 CCID/USB 不同，扩展命令部分采用大端格式。

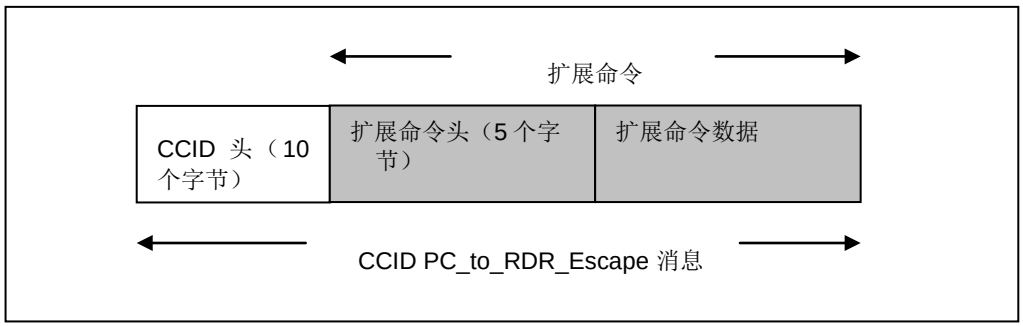

图2： CCID PC_to_RDR_Escape 消息

3.5.2. 命令详情

3.5.2.1. PC_to_ACR89_InputKey

接收键盘输入的数据。此命令的应用场景与插槽无关。

偏移	字段	类型	大小	值	说明
10	*BCmdCode*	16 进制	1	12h	-
11	*wCmdLength*	16 进制	2	0002h	命令数据大小（大端格式）
13	*AbRfu*	16 进制	2	0000h	-
15	*bKeyInputMode*	2 进制	1	-	B0 – 输入模式（b0=0 表示单按键输入, b0=1 表示按键串输入）。按键串输入模式下，按回车键表示完成输入。 B1 – 键盘模式（b1=0 表示数字输入, b1=1 表示字母数字输入） B3 到 b2 – 按键显示（b2=0 表示不显示按键，b2=1 表示显示按键。b2=1 时， b3=0 表示按键显示为明文，b3=1 表示按键显示为星号*。） B4 – 按键输入超时控制（b4=0 表示打开超时设置，b4=1 表示关闭超时设置） B5 – 安全按键传输（b5=0 表示明文传输，b5=1 表示密文传输）。此位保留为将来所用。 B6 – 0/1 – 关闭/开启控制按键 其它 – RFU
16	*bTimeoutValue*	16 进制	1	-	按键输入超时时间的单位是秒。仅在 bKeyInputMode 字段的按键输入超时控制位设置为 0 时有效。

此消息的响应是 ACR89_to_PC_DataBlock 消息。

3.5.2.2. PC_to_ACR89_SetCursor

设置 LCD 光标的新位置。此命令的应用场景与插槽无关。

偏移	字段	类型	大小	值	说明
10	*BcmdCode*	16 进制	1	18h	-
11	*wCmdLength*	16 进制	2	0002h	命令数据大小（大端格式）
13	*AbRfu*	16 进制	2	0000	保留为将来使用
15	*bRowPosition*	16 进制	1	00h 至 07h	新的光标行位置

偏移	字段	类型	大小	值	说明
16	*bColumnPosition*	16 进制	1	00h 至 7Fh	新的光标列位置

此消息的响应是 ACR89_to_PC_DisplayStatus 消息。

3.5.2.3. PC_to_ACR89_SetBacklight

配置 LCD 显示。此命令的应用场景与插槽无关。

偏移	字段	类型	大小	值	说明
10	*BCmdCode*	16 进制	1	19h	-
11	*wCmdLength*	16 进制	2	0001h	命令数据大小（大端格式）
13	*AbRfu*	16 进制	2	0000	保留为将来使用
15	*BBacklight*	16 进制	1	00h 或 01h	00h = 背光关 01h = 背光开 其他值保留为将来所用

此消息的响应是 ACR89_to_PC_DisplayStatus 消息。

3.5.2.4. PC_to_ACR89_DisplayMessage

用 ACR89 内置字体库里的字体从当前光标位置起横向显示字符串。ACR89 将根据字符位置和字符大小自动计算绝对坐标，相应移动光标。此命令的应用场景与插槽有关。

偏移	字段	类型	大小	值	说明
10	*BCmdCode*	16 进制	1	1Bh	-
11	*wCmdLength*	16 进制	2	可变	命令数据大小（大端格式）
13	*AbRfu*	16 进制	2	0000h	保留为将来使用
15	*bCharCoding*	16 进制	1	-	abData 字段的数据编码结构。字符大小取决于数据结构： 00h = ASCII (每个字符占 1 行 x 6 列) 所有其他值保留为将来使用
16	*AbData*	ASCII	字节型数组	-	bCharCoding 字段表明字符串编码结构

此消息的响应是 ACR89_to_PC_DisplayStatus 消息。

3.5.2.5. PC_to_ACR89_DisplayRowGraphic

扫描 LCD 上待显示的一行图形。

偏移	字段	类型	大小	值	说明
10	*bCmdCode*	16 进制	1	23h	-
11	*wCmdLength*	16 进制	2	可变	命令数据大小（大端格式）
13	*abRfu*	16 进制	2	0000h	-
15	*bRowPosition*	16 进制	1	-	起始位置的行索引。行高 8 像素。
16	*bColumnPosition*	16 进制	1	-	起始位置的列索引。
17	*AbData*	16 进制	可变	-	待显示图形的行位图数据

wCmdLength 与 bColumnPosition 的总和不得超过 LCD 列数（128）。

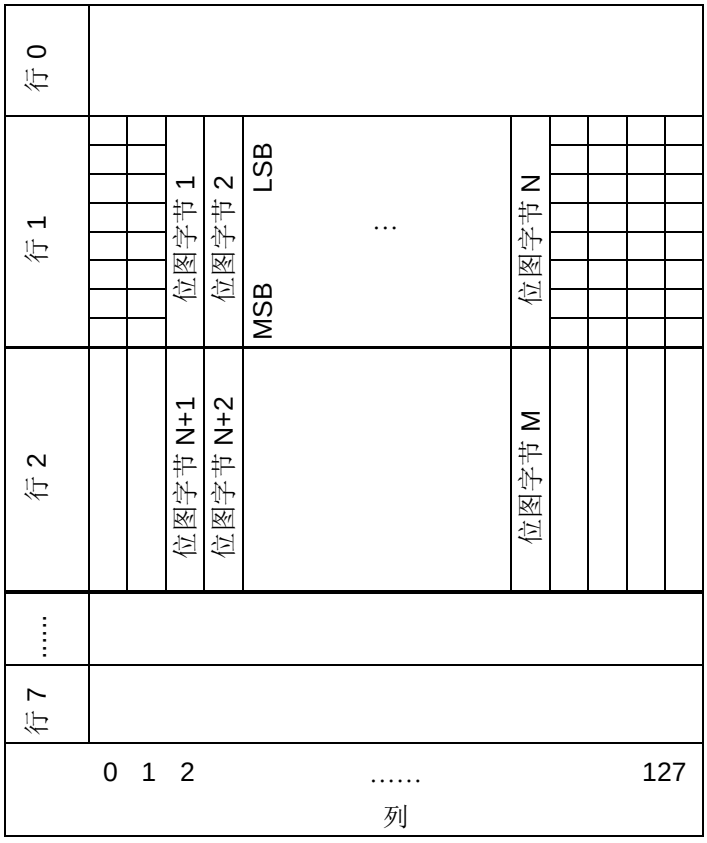

图3： PC_to_ACR89_DisplayGraphic – 位图结构

此消息的响应是 ACR89_to_PC_DisplayStatus 消息。

3.5.2.6. PC_to_ACR89_SetContrast

设置 LCD 对比度级别。此命令的应用场景与插槽无关。

偏移	字段	类型	大小	值	说明
10	*BCmdCode*	16 进制	1	1Ch	-
11	*wCmdLength*	16 进制	2	0001h	命令数据大小（大端格式）
13	*abRfu*	16 进制	2	0000	保留为将来使用
15	*bContrastLevel*	16 进制	1	00h 至 63h	LCD 的新对比度级别

此消息的响应是 ACR89_to_PC_DisplayStatus 消息。

3.5.2.7. PC_to_ACR89_ClearDisplay

清除一行或多行 LCD 显示内容。执行该命令后，光标将移动到所清除块的起始位置。此命令的应用场景与插槽无关。

偏移	字段	类型	大小	值	说明
10	*BcmdCode*	16 进制	1	1Dh	-
11	*wCmdLength*	16 进制	2	0002h	命令数据大小（大端格式）
13	*AbRfu*	16 进制	2	0000h	保留为将来使用
15	*bClearMode*	16 进制	1	00h 或 01h 或 02h	00h = 清除屏幕上的所有内容 01h = 清除光标当前所在的行 02h = 清除光标当前所在行中光标后面的列 所有其他值保留为将来所用
16	*bNumber*	-	1	-	对于 bClearMode = 01h – 待清除的行数 对于 bClearMode = 02h – 待清除的列数 其他情况下无意义

此消息的响应是 ACR89_to_PC_DisplayStatus 消息。

3.5.2.8. PC_to_ACR89_ReadRTC

读取内置实时时钟当前的值。实时时钟每隔半秒钟更新一下时间值。此命令的应用场景与插槽无关。

偏移	字段	类型	大小	值	说明
10	*BCmdCode*	16 进制	1	08h	-
11	*wCmdLength*	16 进制	2	0000h	命令数据大小（大端格式）
13	*AbRFU*	16 进制	2	0000h	-

此消息的响应是 ACR89_to_PC_TimeStamp 消息。

3.5.2.9. PC_to_ACR89_SetRTC

给内置实时时钟设置特定时间值。此命令的应用场景与插槽无关。

偏移	字段	类型	大小	值	说明
10	*BCmdCode*	16 进制	1	09h	-
11	*wCmdLength*	16 进制	2	0006h	命令数据大小（大端格式）
13	*AbRFU*	16 进制	2	0000h	-
15	*bRTCValue*	BCD	6	-	实时时钟的新值格式 YY，MM，DD，HH，MI 和 SS

此消息的响应是 ACR89_to_PC_TimeStamp 消息。

3.5.2.10. PC_to_ACR89_Buzzer

偏移	字段	类型	大小	值	说明
10	*BCmdCode*	16 进制	1	0Ah	-
11	*wCmdLength*	16 进制	2	0002h	命令数据大小（大端格式）
13	*abRfu*	16 进制	2	0000	-
15	*bBuzzerState*	16 进制	1	01h	01h = 蜂鸣器开 00h = 蜂鸣器关
16	*BbuzzerOnDuration*	16 进制	1	-	蜂鸣器开启时间以百分之一毫秒为单位。 仅字段 bBuzzerState 为 01h 时有效。 00h = 激活蜂鸣器并且不关闭 其他值 = 激活蜂鸣器，待其运行所设置数目的百分之一毫秒后，关闭蜂鸣器

此命令消息的响应是 ACR89_to_PC_Echo 消息。

3.5.2.11. PC_to_ACR89_AccessEeprom

此命令使用户可以从 EEPROM 读写数据。最大数据长度是 249 字节。

偏移	字段	类型	大小	值	说明
10	*bCmdCode*	16 进制	1	21h	-
11	*wCmdLength*	16 进制	2	可变	命令数据大小（大端格式）
13	*AbRFU*	16 进制	2	0000h	-
15	*bAccessMode*	ASCII	1	-	‘W’ – 写 EEPROM ‘R’ – 读 EEPROM
16	*BDeviceNumber*	16 进制	1	-	00 – 从 EEPROM 01- 中文字体 EEPROM (Rfu)

偏移	字段	类型	大小	值	说明
17	*AbAddress*	16 进制	4	-	EEPROM 地址 （大端格式）
21	*wDataLength*	16 进制	2	可变	数据长度（读/写）（大端格式）
23	*bEeprom Data*	16 进制	可变	-	EEPROM 数据

此消息的响应是 ACR89_to_PC_DataBlock 消息。

3.5.2.12. PC_to_ACR89_SetLED

允许用户打开和关闭智能卡读写器的电源，插槽 1，和插槽 2，开关状态用红/绿色的 LED 灯表示。

偏移	字段	类型	大小	值	说明
10	*BcmdCode*	16 进制	1	22h	-
11	*WcmdLength*	16 进制	2	0003h	命令数据大小（大端格式）
13	*AbRFU*	16 进制	2	0000h	-
15	*Power LED*	16 进制	1	-	Bit0 :1- 选择红色 Bit1 :1- 选择绿色 Bit2 :1- 选择黄色 Bit7 :0- 关/1- 开 例：开启红色 10000001b 关闭绿色 00000010b 忽略 xxxx0000b
16	*插槽 1 LED*	16 进制	1	-	Bit0 :1- 选择红色 Bit1 :1- 选择绿色 Bit2 :1- 选择黄色 Bit7 :0- 关/1- 开
17	*插槽 2 LED*	16 进制	1	-	Bit0 :1- 选择红色 Bit1 :1- 选择绿色 Bit2 :1- 选择黄色 Bit7 :0-关/1-开

对此命令的响应是 ACR89_to_PC_Echo。

3.5.2.13. PC_to_ACR89_EraseSPIFlash

擦除闪存块。

偏移	字段	类型	大小	值	说明
10	*bCmdCode*	16 进制	1	30h	命令代码
11	*bFlashType*	16 进制	1	02h	SPI 闪存
12	*bRFU*	16 进制	1	00h	-
13	*bStartBlockNum*	16 进制	1	-	除 0 以外的任意数，例，01h

偏移	字段	类型	大小	值	说明
14	*bEndBlockNum*	16 进制	1	-	不小于 bStartBlockNum

此消息的响应是 ACR89_to_PC_ExMemStatus 消息。

注：*闪存块的当前是 64k 字节。*

3.5.2.14. PC_to_ACR89_ProgramSPIFlash

将 256 字节的数据写入 SPI 闪存页。

偏移	字段	类型	大小	值	说明
10	*bCmdCode*	16 进制	1	33h	命令代码
11	*AbAddress*	16 进制	4	xxxxxx00h	闪存页的起始地址 （小端格式）
15	*AbData*	16 进制	256	-	写入闪存页的数据
271	*bCheckSum*	16 进制	1	-	AbData 的校验和

对此消息的响应是 ACR89_to_PC_ExMemStatus 消息。

3.5.2.15. PC_to_ACR89_GetSPIFlashPage

从 SPI 闪存页读取 256 字节的数据。

偏移	字段	类型	大小	值	说明
10	*bCmdCode*	16 进制	1	34h	命令代码
11	*AbAddress*	16 进制	4	xxxxxx00h	闪存页的起始地址 （小端格式）

此消息的响应是 ACR89_to_PC_MemoryPage 消息。

3.5.2.16. PC_to_ACR89_GetVersion

读取启动载入器或应用的固件版本信息。

偏移	字段	类型	大小	值	说明
10	*bCmdCode*	16 进制	1	36h	命令代码
11	*bVersionType*	16 进制	1	-	01h = 启动载入器的版本 02h = 应用版本
12	*AbRFU*	16 进制	3	000000h	-

此消息的响应是 ACR89_to_PC_VersionInfo 消息。

3.5.2.17. PC_to_ACR89_ AuthInfo

读取 RomID 和 RomData。

偏移	字段	类型	大小	值	说明
10	bCmdCode	16 进制	1	38h	命令代码
11	AbRFU	16 进制	16	00…00h	-

此消息的响应是 ACR89_to_PC_AuthInfo 消息。

3.5.3. 扩展命令通道 Bulk-IN 消息

本节描述了 CCID 不涵盖，但 ACR89 兼容的扩展命令（用于操作附加功能）的响应消息。对这些消息的响应总是遵循 CCID 协议 4.2.2.4 部分的 RDR_to_PC_Escape Bulk-IN 消息。

本节定义的响应结构是 abData 域，用于填充 RDR_to_PC_Escape 消息。类似于 CCID 消息结构，该响应结构包括固定长度的响应头和长度可变的响应数据部分。响应头固定是 5 个字节。

与 CCID/USB 不同，扩展响应部分采用大端格式。

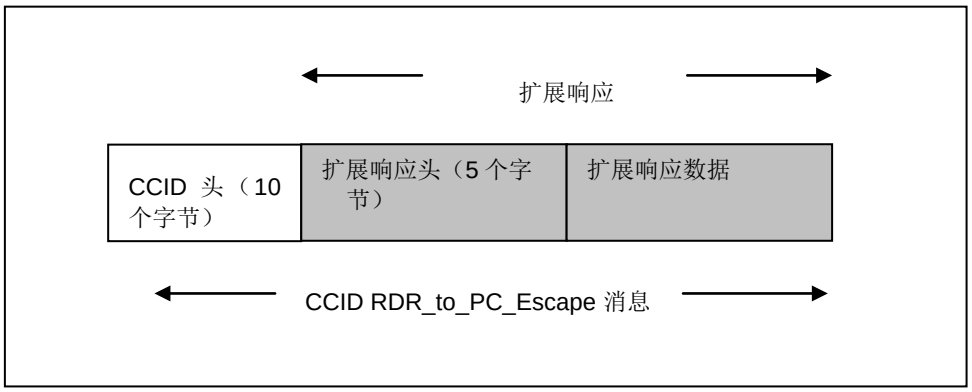

图4：CCID RDR_to_PC_Escape 消息

3.5.4. 消息详情

3.5.4.1. ACR89_to_PC_DataBlock

此消息是 ACR89 对 PC_to_ACR89_InputKey 命令的响应。

对于 PC_to_ACR89_InputKey 命令，将根据选择的按键输入模式返回从键盘获取的单个按键或按键串。

偏移	字段	大小	值	说明
10	*BrespType*	1	81h	-
11	*WReturnCode*	2	-	命令响应码（大端格式）
13	*WRespLength*	2	可变	响应数据大小（大端格式）
15	*Bdata*	可变	-	本字段包含由 ACR89 返还的数据

3.5.4.2. ACR89_to_PC_DisplayStatus

此消息是 ACR89 对以下命令的响应：

PC_to_ACR89_DisplaySetCursor

PC_to_ACR89_DisplayMessage

PC_to_ACR89_DisplayRowGraphic

PC_to_ACR89_ClearDisplay

偏移	字段	大小	值	说明
10	*BrespType*	1	83h	-
11	*wReturnCode*	2	-	命令响应码（大端格式）
13	*wRespLength*	2	0002h	响应数据大小（大端格式）
15	*bRowPosition*	1	00h 至 07h	光标的行位置
16	*bColumnPosition*	1	00h 至 83h	光标的列位置

3.5.4.3. ACR89_to_PC_TimeStamp

此消息是 ACR89 对 PC_to_ACR89_ReadRTC 和 PC_to_ACR89_SetRTC 命令的响应。

偏移	字段	大小	值	说明
10	*BRespType*	1	84h	-
11	*wReturnCode*	2	-	命令响应码（大端格式）
13	*wRespLength*	2	0006h	响应数据大小（大端格式）
15	*bTimeStamp*	6	-	实时时钟当前的值格式 YY，MM，DD，HH，MI 和 SS

3.5.4.4. ACR89_to_PC_Echo

此消息是 ACR89 对以下命令的响应：

PC_to_ACR89_Buzzer

PC_to_ACR89_SetLED

PC_to_ACR89_ExitScriptMode

偏移	字段	大小	值	说明
10	*bRespType*	1	90h	-
11	*wReturnCode*	2	9000h	命令响应码，如果命令运行成功，会返回 90 00h （大端格式）
13	*wRespLength*	2	0000	响应数据大小（大端格式）

3.5.4.5. ACR89_to_PC_ExMemStatus

此消息是 ACR89 对以下命令的响应：

PC_to_ACR89_EraseSPIFlash

PC_to_ACR89_ProgramSPIFlash

偏移	字段	大小	值	说明
10	*bRespType*	1	B0h	-
11	*bReturnState*	1	-	命令返还状态（请参考后面的小节）
12	*bErrorCode*	1	-	错误码（请参考后面的小节）
13	*AbRFU*	2	0000h	-

3.5.4.6. ACR89_to_PC_MemoryPage

此消息是 ACR89 对 PC_to_ACR89_GetSPIFlashPage 命令的响应。

偏移	字段	大小	值	说明
10	*bRespType*	1	B1h	-
11	*bReturnState*	1	-	命令返还状态（请参考后面的小节）
12	*bErrorCode*	1	-	错误码（请参考后面的小节）
13	*AbRFU*	2	0000h	-
15	*AbData*	256	-	从闪存页读取的数据
271	*bCheckSum*	16 进制	1h	AbData 的校验和

注：*命令运行失败时，没有 AbData 和 bCheckSum。*

3.5.4.7. ACR89_to_PC_VersionInfo

此消息是 ACR89 对 PC_to_ACR89_GetVersion 命令的响应。

偏移	字段	大小	值	说明
10	*bRespType*	1	B2h	-
11	*bReturnState*	1	-	命令返还状态（请参考后面的小节）
12	*bErrorCode*	1	-	错误码（请参考后面的小节）
13	*wInfoLength*	2	可变	bInfoData 的大小（小端格式）
15	*bInfoData*	可变	-	固件版本信息（ASCII）

注:*无有效版本信息时，wInfoLength 的长度为 0。*

3.5.4.8. ACR89_to_PC_AuthInfo

此消息是 ACR89 对 PC_to_ACR89_AuthInfo 命令的响应。

偏移	字段	大小	值	说明
10	*bRespType*	1	B4h	-
11	*bReturnState*	1	-	命令返还状态（请参考后面的小节）
12	*bErrorCode*	1	-	错误码（请参考后面的小节）
13	*AbRFU*	2	0000h	-
15	*AbRomID*	8	-	唯一标识符
23	*AbRFU*	48	-	-

注：*命令运行失败时，没有偏移 15 的部分。*

3.5.5. 扩展命令响应码和 返还状态

下表汇总了 ACR89 CCID 扩展命令的响应码和返还状态。

响应码	值	说明
CMD_OKAY	9000h	命令执行成功
INVALID_PARAMETERS	FFFFh	扩展命令的参数错误
INVALID_COMMAND_CODE	FFFEh	扩展命令的命令码（偏移 10）无效
INVALID_COMMAND_LENGTH	FFFDh	扩展命令的长度错误
CANNOT_EXECUTE_COMMAND	FFFCh	扩展命令不能执行
TIMEOUT	FFFBh	执行扩展命令超时
SCRIPT_ERROR	FFFAh	脚本不能执行

返还状态	值	说明
CMD_OK	00h	命令执行成功
CMD_FAIL	01h	命令执行失败

错误码	值	说明
COMMAND_NOT_SUPPORT	00h	不支持扩展命令的命令码（偏移 10）
HARDWARE_ERROR	01h	硬件错误
ACCESS_DENIED	02h	当前配置不允许的函数
ADDRESS_ERROR	03h	地址参数错误
FRAME_ERROR	04h	命令帧格式错误
CHECKSUM_ERROR	05h	数据部分的校验和错误

3.6. CCID Interrupt-IN 消息

Interrupt-IN 端点用于通知主机可能异步发生并且处于主机和 ACR89 的命令-响应交换之外的事件。ACR89 遵循 CCID 标准（1.1 版本）第 6.3 部分关于 Interrupt-IN 消息的规定。此节列举了 ACR89 支持的 CCID 类 Interrupt-IN 消息。

3.6.1. 消息汇总

3.6.1.1. RDR_to_PC_NotifySlotChange

ACR89 检测到 ICC 插槽状态变化时都会发送此消息。

偏移	字段	大小	值	说明
0	*bMessageType*	1	50h	-
1	*bmSlotICCState*	-	-	本字段报告字节粒度。大小(2 比特*插槽数）被向上舍入到最近的字节。每个插槽 2 比特。最低有效位表示插槽的当前状态（0b =无 ICC；1b = 有 ICC）最高有效位表示上一条 RDR_to_PC_NotifySlotChange 消息发出后，插槽的状态是否发生了变化（0b =未变化，1b =变化）。如果指定位置没有插槽，则该字段的这 2 个位返回 00b。 例：一个 3 插槽的 CCID 报告了格式如下的单字节： Bit 0 = 插槽 0 的当前状态 Bit 1 = 插槽 0 的变化状况 Bit 2 = 插槽 1 的当前状态 Bit 3 = 插槽 1 的变化状况 Bit 4 = 插槽 2 的当前状态 Bit 5 = 插槽 2 的变化状况 Bit 6 = 0b Bit 7 = 0b

3.7. CCID 错误码和状态码

本节是对 CCID 标准第 12 部分的扩展，以表格形式列出了 Bulk-IN 消息中有可能与插槽错误寄存器同用的错误码。下表汇总了 CCID 定义的错误码以及为 ACR89 的扩展命令另外定义的错误码。

错误名	错误码	可能原因
CMD_ABORTED	FFh	主机中止了当前活动
ICC_MUTE	FEh	与 ICC 通讯时，CCID 超时
XFR_PARITY_ERROR	FDh	与 ICC 通讯时，奇偶校验错误
XFR_OVERRUN	FCh	与 ICC 通讯时，超限错误
HW_ERROR	FBh	总括性的硬件错误
BAD_ATR_TS	F8h	-
BAD_ATR_TCK	F7h	-
ICC_PROTOCOL_NOT_SUPPORTED	F6h	-
ICC_CLASS_NOT_SUPPORTED	F5h	-
PROCEDURE_BYTE_CONFLICT	F4h	-
DEACTIVATED_PROTOCOL	F3h	-
BUSY_WITH_AUTO_SEQUENCE	F2h	自动序列进行中
PIN_TIMEOUT	F0h	-
PIN_CANCELLED	EFh	-
CMD_SLOT_BUSY	E0h	向正在处理命令的插槽发送另外一个命令
ACR89_ERROR	10h	定义在 ACR89 响应头而不是错误寄存器里的错误码
DEVICE_VOID	11h	ACR89 没有初始化。出厂模式，等待卖主个人化设置或者设备已被篡改。
INVALID_SECRET_KEY	12h	私钥错误
INVALID_MODE	13h	尝试执行命令，当前模式不允许
保留为将来使用	-	（剩余所有未提到的值）

表2：CCID 错误码和状态码

4.0. 动态链接库（DLL）

ACR89 DLL 与 Windows 的 PC/SC 子系统无关。DLL 与 ACR89 的内置附件及应用程序通信时不使用任何 PC/SC。

4.1. ACR89 DLL 的 API 声明

4.1.1. 枚举数

4.1.1.1. 端口号

```
enum
{
   AS_USB1          = 0x00,
   AS_USB2          = 0x01,
   AS_USB3          = 0x02,
   AS_USB4          = 0x03,
   AS_USB5          = 0x04,
   AS_USB6          = 0x05,
   AS_USB7          = 0x06,
   AS_USB8          = 0x07
};
```

被 AS_Open 用于选择连接 ACR89 读写器的 USB 端口。最多选择 8 个 USB 端口。

4.1.1.2. LCD_CLEAR MODE

```
typedef enum _LCD_CLEAR_MODE {
   LCD_CLR_FULL    = 0x00,
   LCD_CLR_ROWS    = 0x01,
   LCD_CLR_COLS    = 0x02
} LCD_CLEAR_MODE;
```

被 AS_ClearLCDDisplay 用于选择 LCD 显示内容的清除方式。

数据成员	值	说明
LCD_CLR_FULL	00h	清除 LCD 全屏
LCD_CLR_ROWS	01h	清除 LCD 显示屏上一行或多行内容
LCD_CLR_COLS	02h	清除 LCD 显示屏上一列或多列内容

```
typedef enum _LED_OPTION {
   LED_UNCHANGED  = 0x00,
   LED_OFF      = 0x01,
   LED_RED      = 0x02,
   LED_GREEN       = 0x03,
   LED_YELLOW      = 0x04
} LED_OPTION;
```

被 AS_SetLED 用于设置 ACR89 的 LED 颜色（共 3 个）。

数据成员	值	说明
LED_UNCHANGED	00h	不改变 LED 颜色。
LED_OFF	01h	关闭 LED。
LED_RED	02h	打开 LED 并设为红色。
LED_GREEN	03h	打开 LED 并设为绿色。
LED_YELLOW	04h	打开 LED 并设为黄色。

4.1.1.3. EEPROM_ACCESS

```
typedef enum _EEPROM_ACCESS {
   READ_EEPROM    = 0x00,
   WRITE_EEPROM   = 0x01
} EEPROM_ACCESS;
```

被 AS_AccessEEProm 用于选择读或写 ACR89 的内部 EEPROM 数据。

数据成员	值	说明
READ_EEPROM	00h	从 EEPROM 读取数据
RITE_EEPROM	01h	向 EEPROM 写入数据

4.1.1.4. SERIAL_ACCESS

```
typedef enum _SERIAL_ACCESS {
   READ_SERIALFLASH  = 0x00,
   WRITE_SERIALFLASH = 0x01,
   ERASE_SERIALFLASH = 0x02
} SERIALFLASH_ACCESS;
```

被 AS_AccessSerialFlash 用于选择读、写或擦除 ACR89 的内部串行闪存数据。

数据成员	值	说明
READ_SERIALFLASH	00h	从串行闪存读取数据
WRITE_SERIALFLASH	01h	向串行闪存写入数据
ERASE_SERIALFLASH	02h	擦除串行闪存的一个块

4.1.2. 读写器命令数据结构

4.1.2.1. KEYPADCONFIG

```
typedef struct _KEYPAD_CONFIG {
   BYTE     cbMaxKeyString;
   BYTE     KeyDisplayRow;
} KEYPADCONFIG, *PKEYPADCONFIG;
```

用于 AS_ConfigureKeyPad。

数据成员	值	说明
cbMaxKeyString	00h 至 0Fh	按键串输入模式下，允许输入的最大按键数（参见小节 **3.5.2.1 - PC_to_ACR89_InputKey** 命令）
KeyDisplayRow	00h 至 03h	显示输入的按键时，LCD 上的起始行号

4.1.2.2. KEYPADINPUT

```
typedef struct _KEYPAD_INPUT{
   BOOLEAN     bEnableKeyString;
   BOOLEAN     bEnableAlphanumeric;
   BOOLEAN     bEnableKeyDisplay;
   BOOLEAN     bEnableMaskedDisplay;
   BOOLEAN     bDisableTimeout;
   BOOLEAN     bEnableKeyEncryption;
   BOOLEAN     bEnableControlKeys;
   BOOLEAN     bReserved2;
   BYTE        cbTimeout;
} KEYPADINPUT, *PKEYPADINPUT;
```

用于 AS_GetKeyInput。

数据成员	值	说明
BEnableKeyString	0 或 1	输入模式 0 – 单按键输入 1 – 按键串输入（按键串输入模式下，按回车键表示完成输入。）
BEnableAlphanumeric	0 或 1	键盘模式 0 – 数字输入 1 – 字母数字输入
BEnableKeyDisplay	0 或 1	按键显示模式 0 – 不显示按键 1 – 显示按键
BEnableMaskedDisplay	0 或 1	按键掩码显示模式 0 – 明文显示按键 1 – 显示按键为“*”
BDisableTimeout	0 或 1	有无按键输入超时设置 0 – 有超时设置 1 – 无超时设置
BEnableKeyEncryption	0 或 1	安全按键传输 0 – 明文传输 1 – 密文传输（保留为将来所用）
BEnableControlKeys	0 或 1	开启/关闭控制键（F1~F4&方向键） 0 – 关闭控制键 1 – 开启控制键

数据成员	值	说明
bReserved2	-	RFU
CbTimeout	0 至 255	按键输入超时时间值以 100ms 为单位（例如，值 100 表示 10 秒）

4.1.2.3. LCDCURSOR

```
typedef struct _LCD_CURSOR {
   BYTE     cbRowPosition; // 0 - 7
   BYTE     cbColPosition; // 0 - 83
} LCDCURSOR, *PLCDCURSOR;
```

被 AS_SetLcdCursor 用于定位 LCD 显示屏上的光标位置

数据成员	值	说明
cbRowPosition	00h 至 07h	光标的行位置
cbColPosition	00h 至 80h	光标的列位置

4.1.2.4. LCDBACKLIGHT

```
typedef struct _LCD_BACKLIGHT {
   BOOLEAN  bEnableBackLight;
} LCDBACKLIGHT, *PLCDBACKLIGHT;
```

被 AS_SetLcdBacklight 用于开启或关闭 LCD 背光。

数据成员	值	说明
bEnableBackLight	0 或 1	0 - 背光关 1 - 背光开

4.1.2.5. LCDGRAPHICS

```
typedef struct _LCD_GRAPHICS {
   LPCTSTR  szBitmapFile;
} LCDGRAPHICS, *PLCDGRAPHICS;
```

用于 AS_SetLcdDisplayGraphics。

数据成员	值	说明
szBitmapFile	-	待显示的 Windows 位图文件的完整路径。位图尺寸可取 128 像素 （宽） x 64 像素 （高）范围内的任意值。色彩深度可以是 1 位，8 位，或 24 位。 注：ACR89 的 LCD 屏只显示黑白图片。

4.1.2.6. LCDMESSAGE

```
typedef struct _LCD_MESSAGE {
BYTE     cbCharCoding;
LPCTSTR  pMessage;
USHORT   wMessageLen;
} LCDMESSAGE, *PLCDMESSAGE;
```

用于 AS_SetLcdDisplayMessage。

数据成员	值	说明
cbCharCoding	00h	pMessage 字段用的数据编码结构。字符大小取决于数据结构。 00h – ASCII 所有其他值保留为将来使用
pMessage	ASCII 串	cbCharCoding 字段表明字符串编码结构
wMessageLen	正整数	pMessage 内存储的字符数

4.1.2.7. LCDCONTRAST

```
typedef struct _LCD_CONTRAST {
   BYTE     cbContrastLevel;
} LCDCONTRAST, *PLCDCONTRAST;
```

被 AS_SetLcdSetContrast 用于设置 LCD 屏幕的对比度。

数据成员	值	说明
cbContrastLevel	00h 至 63h	LCD 的新对比度级别

4.1.2.8. LCDCLEAR

```
typedef struct _LCD_CLEAR {
   BYTE     cbClearMode;
   BYTE     cbNumber;
} LCDCLEAR, *PLCDCLEAR;
```

被 AS_ClearLcdDisplay 用于清除全部或部分 LCD 显示内容。

数据成员	值	说明
cbClearMode	LCD_CLEAR_MODE	LCD_CLR_FULL = 清除 LCD 全屏内容 LCD_CLR_ROWS = 清除行内容 LCD_CLR_COLS = 清除列内容
cbNumber	LCD_CLR_ROWS	LCD_CLR_ROWS = 待清除的行数 * LCD_CLR_COLS = 待清除的列数 * *LCD_CLR_FULL 模式下无效。

4.1.2.9. LED

```
typedef struct _LED {
   BYTE     cbLedPower;       // see LED_OPTION
   BYTE     cbLedSlot1;       // see LED_OPTION
   BYTE     cbLedSlot2;       // see LED_OPTION
} LED, *PLED;
```

被 AS_SetLED 用于控制 ACR89 的 LED。

数据成员	值	说明
CbLedPower	LED_OPTION	控制电源 LED LED_UNCHANGED – 保持 LED 现状 LED_OFF – 关闭 LED LED_RED – 打开红色 LED LED_GREEN- 打开绿色 LED LED_YELLOW – 打开黄色 LED
cbLedSlot1	LED_OPTION	控制插槽 1 的 LED 可选操作请参见上述 LED_OPTION 内容。
cdLedSlot2	LED_OPTION	控制插槽 2 的 LED 可选操作请参见上述 LED_OPTION 内容。

4.1.2.10. 蜂鸣器

```
typedef struct _BUZZER {
   BYTE     cbBuzzerState;
   BYTE     cbBuzzerOnDuration;
} BUZZER, *PBUZZER;
```

用于 AS_SetBuzzer。

数据成员	值	说明
cbBuzzerState	0 或 1	0 = 蜂鸣器关闭 1 = 蜂鸣器开启
cbBuzzerOnDuration	0 - 255	蜂鸣时间以 100 ms 为单位（例如，值 100 表示 10 秒）

4.1.3. 读写器响应数据

4.1.3.1. AS_STATUS

```
typedef struct _AS_STATUS {
   DLL_ERROR      DllError;
   LONG        W32Error;
} AS_STATUS;
```

数据成员	值	说明
DllError	00h – 20h	包含命令执行期间用 DLL 设置的错误码。另见 Appendix A。
W32Error	Win32 错误码。	包含 Win32 API 执行期间 Windows 系统设置的错误码。

4.1.3.2. INFO

```
typedef struct _INFO {
      CHAR     szUID[8];
      CHAR     szBootloaderVersion[64];
   CHAR     szFirmwareSDKVersion[64];
} INFO, *PINFO;
```

以上信息是由 AS_GetInfo 返回的，包含 ACR89 读写器的详细信息和功能。

数据成员	值	说明
szUID	8 个字节	设备独有的 ID。固定为 8 个字节
szBootloaderVersion	64 个字节的 ASCII	ASCII 格式的 Bootloader 版本号，以空字符结尾
szFirmwareSDKVersion	64 个字节的 ASCII	ASCII 格式的固件 SDK 版本号，以空字符结尾

4.1.3.3. KEYPADSTATUS

```
typedef struct _KEYPAD_STATUS {
   BYTE     cbMaxKeyString;
   BYTE     cbKeyDisplayMode;
} KEYPADSTATUS, *PKEYPADSTATUS;
```

以上信息是由 AS_GetKeyPadConfig 和 AS_ConfigureKeyPad 返回的。

数据成员	值	说明
cbMaxKeyString	0 – 255	按键串输入模式下，允许输入的最大按键数（参见 AS_InputKey 命令）
cbKeyDisplayMode	0 – 7	显示输入的按键时，LCD 上的起始行号

4.1.3.4. DISPLAYSTATUS

```
typedef struct _DISPLAY_STATUS {
   BYTE     cbRowPosition;
   BYTE     cbColumnPosition;
} DISPLAYSTATUS, *PDISPLAYSTATUS;
```

以上信息是由下列函数返回的：

AS_SetLcdCursor

AS_SetLcdBacklight

AS_SetLcdDisplayGraphics

AS_SetLcdDisplayMessage

AS_SetLcdSetContrast

AS_ClearLcdDisplay

数据成员	值	说明
cbRowPosition	0 - 7	光标的行位置
cbColumnPosition	0 - 127	光标的列位置

4.1.3.5. DATABLOCK

```
typedef struct _DATA_BLOCK {
   USHORT   wDataLen;
   PBYTE    pDataBlock;
} DATABLOCK, *PDATABLOCK;
```

以上信息是由 nangi 返回的，包含函数返回的数据。

数据成员	值	说明
wDataLen	-	命令执行前 pDataBlock 的长度。存储 ACR89 执行命令以后返回的数据块的长度。
pDataBlock	-	即将在命令中输入的数据或 ACR89 返回的数据。

4.1.4. 读写器共享命令/响应数据结构

4.1.4.1. TIMESTAMP

```
typedef struct _TIMESTAMP {
   CHAR     szRTCValue[6];
} TIMESTAMP, *PTIMESTAMP;
```

被 AS_ReadRTC 和 AS_SetRTC 用于获取或设置 ACR89 的运行时钟值。

数据成员	值	说明
szRTCValue[0]	00 – 99	年 （小端格式）
szRTCValue[1]	1 – 12	月

数据成员	值	说明
szRTCValue[2]	1 – 31	天
szRTCValue[3]	1 – 23	小时
szRTCValue[4]	0 – 59	分钟
szRTCValue[5]	0 - 59	秒钟

4.1.4.2. ACCESSEEPROM

```
typedef struct _ACCESS_EEPROM {
   BYTE     cbFunction;
   BYTE     cbDeviceNumber;
   DWORD    dwAddress;
   USHORT   wDataLength;
   PBYTE    pData
} ACCESSEEPROM, *PACCESSEEPROM;
```

被 AS_AccessEEPROM 用于读写 ACR89 的 EEPROM 存储。

数据成员	值	说明
cbFunction	EEPROM_ACCESS	00h = READ_EEPROM 01h = WRITE_EEPROM
cbDeviceNumber	00h 或 01h	00h = 从 EEPROM 01h = 中文字体 EEPROM
dwAddress	4 个字节，两个双字（16 进制）	EEPROM 地址
wDataLength	2 个字节，一个双字（16 进制）	数据长度（读/写）
pData	指向 wDataLength 缓存的指针	读 EEPROM：指向用于存储被读 EEPROM 数据的缓存的指针。 写 EEPROM：指向包含将写入 EEPROM 的数据的缓存的指针。

4.1.4.3. ACCESSSERIALFLASH

```
typedef struct _ACCESS_SERIALFLASH {
   BYTE     cbFunction;
   DWORD    dwAddress;
   USHORT   wDataLength;
   PBYTE    pData
} ACCESSSERIALFLASH, *PACCESSSERIALFLASH;
```

被 AS_AccessEEPROM 用于读、写或擦除 ACR89 的串行闪存数据。

数据成员	值	说明
cbFunction	SERIALFLASH_ACCESS	00h = READ_SERIALFLASH 01h = WRITE_SERIALFLASH 02h = ERASE_SERIALFLASH
dwAddress	4 个字节，两个双字（16 进制）	串行闪存的地址
wDataLength	2 个字节，一个双字（16 进制）	读串行闪存：数据长度 写串行闪存：数据长度 擦除串行闪存：忽略
pData	指向 wDataLength 缓存的指针	读串行闪存：指向用于存储被读串行闪存数据的缓存的指针。 写串行闪存：指向包含将写入串行闪存的数据的缓存的指针。 擦除串行闪存：忽略

注：

1. *将数据写入一个区域前，应先执行擦除操作。*
2. *擦除操作是按块进行的，每一个块是 64 kbytes。*
3. *对于擦除操作，仅用到 2 个最高有效字节。将忽略 2 个最低有效字节。例如，64 kbytes 地址对齐。*

4.2. ACR89 DLL 的 API 函数

4.2.1. 一般描述

所有函数都会返回状态码 AS_STATUS。AS_STATUs 结构含有一个 DLL 定义的错误和一个 WIN32 错误。关于 AS_STATUs 结构的详细信息，另见小节 2.3.1。代码 AS_STATUS.DllError == CMD_SUCCESS 表示命令执行成功。只有 AS_STATUS.DllError != CMD_SUCCESS 时，才会使用 AS_STATUS.W32Error 为开发人员提供附加错误信息。根据所控制附件的类型，API 函数分为 7 类：

- 端口函数
- 设备函数
- LCD 函数
- 键盘函数
- 实时时钟函数
- 脚本函数
- 其它函数

4.2.2. 端口函数

4.2.2.1. AS_Open

打开 ACR89 的逻辑连接。调用任意其它 API 函数前必须先调用该函数。

```
AS_STATUS AS_DECL AS_Open (
          IN INT nReaderType,
          IN INT nPort,
          OUT   INT *nDevId);
```

参数：

nReaderType	[in] 必须是 ACR89（00h, 如 acr89.h 定义）
nPort	[in] USB 端口连接的读写器实例。例： AS_USB1 表示 PC 检测到的第一个 ACR89。对于可能选项，另见小节 **4.1.1.1“端口号”。**
nDevId	[out] 成功建立连接后返回的句柄。后续调用其它所有 API 函数时都会用到该句柄。

返回值

AS_STATUS	根据执行成功或失败的情况返回不同的值。AS_STATUS.DllError 包含 DLL 返回的状态。AS_STATUS.W32Error 包含 DLL 异常相关的 Win32 错误码（如果存在的话）。关于其它可能返回的错误码，另见**附录 A**。

例：

```
INT      nDid;
AS_STATUS   status;

//连接 ACR89。
status = AS_Open(ACR89,AS_USB1,&nDid);
if(status.DllError == CMD_SUCCESS) {
   // 连接成功，用 ACR89 执行操作。
```

```
}
Else {
    // 硬件错误
    return status;
}
```

4.2.2.2. AS_Close

关闭 ACR89 的逻辑连接。

```
AS_STATUS AS_DECL AS_Close (
          IN INT nDevId);
```

参数

nDevId [in] 先前调用 AS_Open 时返回的句柄。

返回值

AS_STATUS 根据执行成功或失败的情况返回不同的值。AS_STATUS.DllError 包含 DLL 返回的状态。AS_STATUS.W32Error 包含 DLL 异常相关的 Win32 错误码（如果存在的话）。关于其它可能返回的错误码，另见**附录 A**。

例：

```
INT       nDid;
AS_STATUS    status;

//连接 ACR89。
status = AS_Open(ACR89, AS_USB1, &nDid);
if(status.DllError == CMD_SUCCESS) {
    //连接成功，执行操作
    .
.
.
//操作完成，断开连接
status = AS_Close(nDid);
}
else {
    //发生异常
    return status;
}
```

4.2.3. 设备函数

设备函数允许初始化、获取发送到或来自于 ACR89 的各种参数。

4.2.3.1. AS_GetInfo

获取 ACR89 的一般信息。

```
AS_STATUS AS_DECL AS_GetInfo (
          IN INT nDevId,
          OUT    PINFO pInfo);
```

参数：

nDevId [in] 先前调用 AS_Open 时返回的句柄。

pInfo [out] 指向 INFO 结构。该结构保存有 ACR89 的一般信息。关于 INFO 结构的详细信息，另见小节 **4.1.3.2**。

返回值

AS_STATUS 根据执行成功或失败的情况返回不同的值。AS_STATUS.DllError 包含 DLL 返回的状态。AS_STATUS.W32Error 包含 DLL 异常相关的 Win32 错误码（如果存在的话）。关于其它可能返回的错误码，另见**附录 A**。

例：

```
INT      nDid;
INFO     Info;
AS_STATUS   status;

//连接 ACR89。
status = AS_Open(ACR89,AS_USB1,&nDid);
if(status.DllError == CMD_SUCCESS) {

   //connection success, get the reader information.
status = AS_GetInfo(nDid,&Info);
   if (status.DLLError == CMD_SUCCESS) {
      //用获取的信息执行操作
}

   //断开连接
   status = AS_Close(nDid);
}
else {
   return status;
}
```

4.2.3.2. AS_AccessEEProm

使用户能读写 EEPROM 数据。最大数据长度是 256 字节。

```
AS_STATUS AS_DECL AS_AccessEEPRom (
         IN INT nDevId,
         IN PACCESSEEPROM pEEPRom);
```

参数：

nDevId [in] 先前调用 AS_Open 时返回的句柄。

pEEPRom [in] 指向 ACCESSEEPROM 结构的指针。该结构包含写入 ACR89 的数据或从 ACR89 读取的数据。更多 ACCESSEEPROM 结构的详细信息，另见小节 **4.1.4.2**。

返回值

AS_STATUS 根据执行成功或失败的情况返回不同的值。AS_STATUS.DllError 包含 DLL 返回的状态。AS_STATUS.W32Error 包含 DLL 异常相关的 Win32 错误码（如果存在的话）。关于其它可能返回的错误码，另见 Appendix A。

例：

```
INT         nDid;
ACCESSEEPROM   eeprom;
BYTE        aData[256];
AS_STATUS      status;

//从地址 0x0000 开始读取 EEPROM 数据。
//假设已经创建连接
eeprom.cbAccessMode  = READ_EEPROM;
eeprom.wAddress    = 0x0000;
eeprom.wDataLength    = 0x0100;
eeprom.pData       = aData;
status = AS_AccessEEPRom(nDid, &eeprom);
if (status.DLLError == CMD_SUCCESS) {
   //用读取的数据执行操作
}
else {
   //发生异常
}
return status;
```

4.2.3.3. AS_AccessSerialFlash

使用户能读写串行闪存数据。最大数据长度是 256 字节。

```
AS_STATUS AS_DECL AS_AccessSerialFlash (
         IN INT nDevId,
         IN PACCESSSERIALFLASH pSerialFlash);
```

参数：

nDevId [in] 先前调用 AS_Open 时返回的句柄。

pEEPRom [in] 指向 ACCESSSERIALFLASH 结构的指针。该结构包含写入 ACR89 的数据或从 ACR89 读取的数据。更多 ACCESSSERIALFLASH 结构的详细信息，另见小节 **4.2.3.3**。

返回值

AS_STATUS **根据**执行成功或失败的情况返回不同的值。AS_STATUS.DllError **包含** DLL **返回的状**态。AS_STATUS.W32Error **包含** DLL **异常相关的** Win32 错误码（如果存在的话）。关于其它可能返回的错误码，另见 **Appendix A**。

例：

```
INT         nDid;
ACCESSSERIALFLASH serialflash;
BYTE        aData[256];
AS_STATUS      status;

//从地址 0x0000 开始读取 EEPROM 数据。
//假设已经创建连接
serialflash.cbAccessMode   = READ_SERIALFLASH;
serialflash.wAddress = 0x0000;
serialflash.wDataLength = 0x0100;
```

```
serialflash.pData    = aData;
status = AS_AccessEEPRom(nDid, &serialflash);
if (status.DLLError == CMD_SUCCESS) {
   //用读取的数据执行操作
}
else {
   //发生异常
}
return status;
```

4.2.4. LCD 函数

LCD 函数控制 LCD 面板的对比度，背光状态和光标位置。也用于在 LCD 面板上显示图片和文字。

4.2.4.1. AS_SetLcdCursor

设置 LCD 光标的新位置。

```
AS_STATUS AS_DECL AS_SetLcdCursor (
         IN INT nDevId,
         IN PLCDCURSOR pLcdCursor,
         OUT   PDISPLAYSTATUS pDisplayStatus);
```

参数：

nDevId	[in] 先前调用 AS_Open 时返回的句柄。
pLcdCursor	[in] 指向 LCDCURSOR 结构的指针。该结构包含待设置的光标位置。更多 LCDCURSOR 结构的详细信息，另见小节 **4.1.2.3**。
pDisplayStatus	[out] 指向 DISPLAYSTATUS 结构的指针。该结构保存新设置的位置参数。更多 DISPLAYSTATUS 结构的详细信息，另见小节 **4.1.3.4**。

返回值

AS_STATUS	**根据**执行成功或失败的情况返回不同的值。AS_STATUS.DllError **包含** DLL **返回的状**态。AS_STATUS.W32Error **包含** DLL **异常相关的** Win32 错误码（如果存在的话）。关于其它可能返回的错误码，另见 **Appendix A**。

例：

```
LCDCURSOR       lcdCursor;
DISPLAYSTATUS   displayStatus;
AS_STATUS       status;

//光标定位在 LCD 左上角
//假设已经创建连接
lcdCursor.cbColPosition = 0;
lcdCursor.cbRowPosition = 0;

status = AS_SetLcdCursor(nDid, &lcdCursor, &displayStatus);
```

4.2.4.2. AS_SetLcdBacklight

打开或关闭 LCD 背光。

```
AS_STATUS AS_DECL AS_SetLcdBacklight (
          IN INT nDevId,
          IN PLCDBACKLIGHT pLcdBacklight,
          OUT   PDISPLAYSTATUS pDisplayStatus);
```

参数：

nDevId [in] 先前调用 AS_Open 时返回的句柄。

pLcdBacklight [in] 指向 LCDBACKLIGHT 结构的指针。该结构包含待设置的光标位置。更多 LCDBACKLIGHT 结构的详细信息，另见小节 **4.1.2.4**。

pDisplayStatus [out] 指向 DISPLAYSTATUS 结构的指针。该结构包含动作完成后的光标位置。更多 DISPLAYSTATUS 结构的详细信息，另见小节 **4.1.3.4**。

返回值

AS_STATUS 根据执行成功或失败的情况返回不同的值。AS_STATUS.DllError 包含 DLL 返回的状态。AS_STATUS.W32Error 包含 DLL 异常相关的 Win32 错误码（如果存在的话）。关于其它可能返回的错误码，另见 Appendix A。

例：

```
LCDBACKLIGHT    lcdLight;
DISPLAYSTATUS   lcdStatus;
AS_STATUS       status;

//打开 LCD 背光
//假设已经创建连接
lcdLight.bEnableBackLight = TRUE;
status = AS_SetLcdBacklight(nDid, &lcdLight, &lcdStatus);
```

4.2.4.3. AS_SetLcdDisplayGraphic

传输位图图像给 ACR89 并在 LCD 上的当前光标位置显示图像。位图格式如图 5 所示。执行该命令后，光标将移动到图片后面（图片右下角位置）。位图的最大尺寸是 128 像素 （宽）x 64 像素（高）。

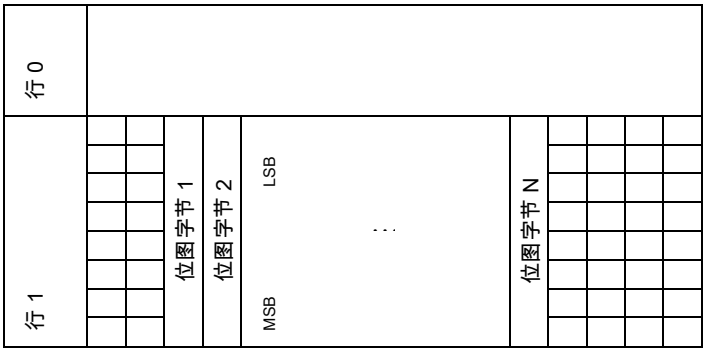

图5：ACR89 读写器的位图格式

```
AS_STATUS AS_DECL AS_SetLcdDisplayGraphics (
          IN INT nDevId,
          IN PLCDGRAPHICS pLcdGraphics,
          OUT   PDISPLAYSTATUS pDisplayStatus);
```

参数：

nDevId [in] 先前调用 AS_Open 时返回的句柄。

pLcdGraphics [in] 指向 LCDGRAPHICS 结构的指针。该结构表明位图文件的存储路径。更多 LCDGRAPHICS 结构的详细信息，另见小节 **4.1.2.5**。

pDisplayStatus [out] 指向 DISPLAYSTATUS 结构的指针。该结构包含显示图片后的光标位置。更多 DISPLAYSTATUS 结构的详细信息，另见小节 **4.1.3.4**。

返回值

AS_STATUS 根据执行成功或失败的情况返回不同的值。AS_STATUS.DllError 包含 DLL 返回的状态。AS_STATUS.W32Error 包含 DLL 异常相关的 Win32 错误码（如果存在的话）。关于其它可能返回的错误码，另见 **Appendix A**。

例：

```
LCDGRAPHICS    lcdGrafx;
DISPLAYSTATUS  lcdStatus;
AS_STATUS      status;
//显示位图文件“MyLogo.bmp”。
//假设已经创建连接
lcdGrafx.szBitmapFile   = “MyLogo.bmp”;
status = AS_SetLcdDisplayGraphics(nDid, &lcdGrafx, &lcdStatus);
```

4.2.4.4. AS_SetLcdDisplayMessage

此函数可用 ACR89 内置字体库里的字体显示字符串。从当前光标位置横向显示字符串。ACR89 将根据字符位置和字符大小自动计算绝对坐标，并相应移动光标。文本占满一行时，文本将被覆盖。

```
AS_STATUS AS_DECL AS_SetLcdDisplayMessage (
          IN INT nDevId,
          IN PLCDMESSAGE pLcdMessage,
          OUT   PDISPLAYSTATUS pDisplayStatus);
```

参数：

nDevId [in] 先前调用 AS_Open 时返回的句柄。

pLcdMessage [in] 指向 LCDMESSAGE 结构的指针。该结构表明 LCD 待显示的字母数字文本。更多 LCDMESSAGE 结构的详细信息，另见小节 **4.1.2.6**。

pDisplayStatus [out] 指向 DISPLAYSTATUS 结构的指针。该结构包含显示消息后的光标位置。更多 DISPLAYSTATUS 结构的详细信息，另见小节 **4.1.3.4**。

返回值

AS_STATUS 根据执行成功或失败的情况返回不同的值。AS_STATUS.DllError 包含 DLL 返回的状态。AS_STATUS.W32Error 包含 DLL 异常相关的 Win32 错误码（如果存在的话）。关于其它可能返回的错误码，另见 **Appendix A**。

例：

```
const char      szText[]=”Welcome to the ACR89”;
LCDMESSAGE      lcdMsg;
DISPLAYSTATUS   lcdStatus;
AS_STATUS       status;

//显示上述文本
//假设已经创建连接
lcdMsg.cbCharCoding  = 0x00;
lcdMsg.pMessage      = szText;
lcdMsg.wMessageLen   = strlen(szText);
status = AS_SetLcdDisplayMessage(nDid, &lcdMsg, &lcdStatus);
```

4.2.4.5. AS_SetLcdSetContrast

设置 LCD 对比度级别。

```
AS_STATUS AS_DECL AS_SetLcdSetContrast (
          IN INT nDevId,
          IN PLCDCONTRAST pLcdContrast,
          OUT   PDISPLAYSTATUS pDisplayStatus);
```

参数：

nDevId	[in] 先前调用 AS_Open 时返回的句柄。
pLcdContrast	[in] 指向 LCDCONTRAST 结构的指针。该结构表明位图文件的存储路径。更多 LCDCONTRAST 结构的详细信息，另见小节 **4.1.2.7**。
pDisplayStatus	[out] 指向 DISPLAYSTATUS 结构的指针。该结构包含显示图片后的光标位置。更多 DISPLAYSTATUS 结构的详细信息，另见小节 **4.1.3.4**。

返回值

AS_STATUS	根据执行成功或失败的情况返回不同的值。AS_STATUS.DllError 包含 DLL 返回的状态。AS_STATUS.W32Error 包含 DLL 异常相关的 Win32 错误码（如果存在的话）。关于其它可能返回的错误码，另见 **Appendix A**。

例：

```
LCDCONTRAST     lcdContrast;
DISPLAYSTATUS   lcdStatus;
AS_STATUS       status;

//将 LCD 对比度设为 100%
//假设已经创建连接
lcdContrast. cbContrastLevel = 0x3f;
status = AS_SetLcdSetContrast (nDid, &lcdContrast, &lcdStatus);
```

4.2.4.6. AS_ClearLcdDisplay

清除一行（列）或多行（列）LCD 显示内容。执行该命令后，光标将移动到所清除块的起始位置。

```
AS_STATUS AS_DECL AS_ClearLcdDisplay (
          IN INT nDevId,
          IN PLCDCLEAR pLcdClear,
          OUT   PDISPLAYSTATUS pDisplayStatus);
```

参数：

nDevId [in] 先前调用 AS_Open 时返回的句柄。

pLcdClear [in] 指向 LCDCLEAR 结构的指针。该结构表明 LCD 的清除模式。更多 LCDCLEAR 结构的详细信息，另见小节 **4.1.2.8**。

pDisplayStatus [out] 指向 DISPLAYSTATUS 结构的指针。该结构包含显示图片后的光标位置。更多 DISPLAYSTATUS 结构的详细信息，另见小节 **4.1.3.4**。

返回值

AS_STATUS 根据执行成功或失败的情况返回不同的值。AS_STATUS.DllError 包含 DLL 返回的状态。AS_STATUS.W32Error 包含 DLL 异常相关的 Win32 错误码（如果存在的话）。关于其它可能返回的错误码，另见 **Appendix A**。

例：

```
LCDCLEAR    lcdClear;
DISPLAYSTATUS  lcdStatus;
AS_STATUS      status;

//清除 LCD 全屏
//假设已经创建连接
lcdClear.cbClearMode = LCD_CLR_FULL;
lcdClear.cbNumber    = 0x00; //忽略

status = AS_ClearLcdDisplay(nDid, &lcdClear, &lcdStatus);
```

4.2.5. 键盘函数

使用户可以配置 ACR89 的键盘及处理按键输入。

4.2.5.1. AS_GetKeyPadConfig (API 定义仍处于初级阶段)

读取 ACR89 键盘的当前配置。

```
AS_STATUS AS_DECL AS_GetKeyPadConfig (
          IN INT nDevId,
          OUT   PKEYPADSTATUS pKeypadStatus);
```

参数：

nDevId [in] 先前调用 AS_Open 时返回的句柄。

pKeypadStatus [out] 指向 KEYPADSTATUS 结构的指针。该结构包含键盘的当前配置。更多 KEYPADSTATUS 结构的详细信息，另见小节 **4.1.3.3** 。

返回值

AS_STATUS 根据执行成功或失败的情况返回不同的值。AS_STATUS.DllError 包含 DLL 返回的状态。AS_STATUS.W32Error 包含 DLL 异常相关的 Win32 错误码（如果存在的话）。关于其它可能返回的错误码，另见 **Appendix A**。

4.2.5.2. AS_ConfigureKeyPad (API 定义仍处于初级阶段)

配置 ACR89 键盘。

```
AS_STATUS AS_DECL AS_ConfigureKeyPad (
          IN INT nDevId,
          IN PKEYPADCONFIG pKeypadConfig,
          OUT   PKEYPADSTATUS pKeypadStatus);
```

参数：

nDevId [in] 先前调用 AS_Open 时返回的句柄。

pKeypadConfig [in] 指向 KEYPADCONFIG 结构的指针。该结构表明 ACR89 键盘的配置。更多 KEYPADCONFIG 结构的详细信息，另见小节 **4.1.2.1**。

pKeypadStatus [out] 指向 KEYPADSTATUS 结构的指针。该结构保存键盘的当前配置。更多 KEYPADSTATUS 结构的详细信息，另见小节 **4.1.3.3**。

返回值

AS_STATUS 根据执行成功或失败的情况返回不同的值。AS_STATUS.DllError 包含 DLL 返回的状态。AS_STATUS.W32Error 包含 DLL 异常相关的 Win32 错误码（如果存在的话）。关于其它可能返回的错误码，另见 **Appendix A**。

4.2.5.3. AS_GetKeyInput

使用户可以在 ACR89 上输入按键。根据 KEYPADINPUT 构的设置，可能采用单按键输入或按键串输入模式。采用以下格式返回按键信息：

模式	按键	值
数字	0 ~ 9	0 ~ 9
字母数字	0 ~ 9	ASCII 码
所有模式	清除	10h
	确定	0Dh
	F1	3Dh
	F2	3Eh
	F3	3Fh
	F4	0Ch

表3：键盘输入格式

注：

如果在按键串输入模式下按下单个功能键，输入信息无效，但会返回功能码。

*按键串输入模式下，**Enter** 键将返回按下的按键，而 Clear 键将清除最后一个输入的按键。*

按键串输入模式下，如果输入的字符串全部被清除，AS_KeyInput 将返回空字符串。

输入方向键则不会有返回信息。方向键仅用于在 ACR89 的 LCD 屏上控制方向。

```
AS_STATUS AS_DECL AS_GetKeyInput (
          IN INT nDevId,
          IN PKEYPADINPUT pKeypadInput,
          OUT   PDATABLOCK pDataBlock);
```

参数：

nDevId	[in] 先前调用 AS_Open 时返回的句柄。
pKeypadInput	[in] 指向 KEYPADINPUT 结构的指针。该结构表明 ACR89 接收按键输入信息时可用的选项。 关于 KEYPADINPUT 结构的详细信息，另见小节 **2.2.3**。
pDataBlock	[out] 指向 DATABLOCK 结构的指针。该结构包括输入的按键（如果有的话）。更多 DATABLOCK 结构的详细信息，另见小节 **4.1.3.5**。

返回值

AS_STATUS	根据执行成功或失败的情况返回不同的值。AS_STATUS.DllError 包含 DLL 返回的状态。AS_STATUS.W32Error 包含 DLL 异常相关的 Win32 错误码（如果存在的话）。关于其它可能返回的错误码，另见**附录 A**。

例：

```
BYTE     aKeys[16];
KEYPADINPUT kpInput;
DATABLOCK   dataBlk;
AS_STATUS   status;

//使用户可以输入一个字符串
//假设已经创建连接
dataBlk.pDataBlock = aKeys;
kpInput.bEnableKeyString    = TRUE;  //输入一个字符串
kpInput.bEnableAlphanumeric    = TRUE;  //输入的字符串是字母数字型的
kpInput.bEnableKeyDisplay  = TRUE;  //在 LCD 上显示按键
kpInput.bEnableMaskedDisplay  = FALSE; //按键无掩码
kpInput.bEnableControlKeys    = FALSE; //停用控制键
kpInput.bDisableTimeout    = 1;     //没有超时设置
kpInput.bEnableKeyEncryption  = 0;   //返回按键无加密
status = AS_GetKeyInput(nDid,&kpInput,&data);
```

4.2.6. 实时时钟函数

使用户可以读取和设置 ACR89 的内置运行时钟。

4.2.6.1. AS_ReadRTC

读取内置实时时钟当前的值。实时时钟每隔半秒钟更新一下时间值。

```
AS_STATUS AS_DECL AS_ReadRTC (
          IN INT nDevId,
          OUT   PTIMESTAMP pTimeStamp);
```

参数：

nDevId [in] 先前调用 AS_Open 时返回的句柄。

pTimeStamp [out] 指向 TIMESTAMP 结构的指针。该结构包含 ACR89 内置实时时钟返回的当前时间。更多 TIMESTAMP 结构的详细信息，另见小节 **4.1.4.1** 。

返回值

AS_STATUS 根据执行成功或失败的情况返回不同的值。AS_STATUS.DllError 包含 DLL 返回的状态。AS_STATUS.W32Error 包含 DLL 异常相关的 Win32 错误码（如果存在的话）。关于其它可能返回的错误码，另见 **附录 A** 。

例：

```
AS_STATUS   status;
TIMESTAMP   tsRTC;
char     szTime[81];

//读取 ACR89 的实时时钟。
//假设已经创建连接
status = AS_ReadRTC(nDevId, &tsRTC);
if(status.DllError == CMD_SUCCESS)
   // 显示 ACR89 返回的日期和时间。
sprintf(szTime, “RTC:%d/%d/%d %d:%d:%d (yy/mm/dd hh:mm:ss)”,
  tsRTC. szRTCValue[0], tsRTC. szRTCValue[1],
  tsRTC. szRTCValue[2], tsRTC. szRTCValue[3],
  tsRTC. szRTCValue[4], tsRTC. szRTCValue[5]);
   ::MessageBox(0L, szTime, “”, MB_OK);
}
return status;

return status;
```

4.2.6.2. AS_SetRTC

将内置实时时钟设置为 TIMESTAMP 结构中的值。

```
AS_STATUS AS_DECL AS_SetRTC (
          IN INT nDevId,
          IN PTIMESTAMP pNewTime,
          OUT   PTIMESTAMP pTimeStamp);
```

参数：

nDevId [in] 先前调用 AS_Open 时返回的句柄。

pNewTime [in] 指向 TIMESTAMP 结构的指针。该结构包含内置实时时钟将设置的新时间值。更多 TIMESTAMP 结构的详细信息，另见小节 **4.1.4.1**。

pTimeStamp [out] 指向 TIMESTAMP 结构的指针。该结构保存新设置的时间值。更多 TIMESTAMP 结构的详细信息，另见小节 **4.1.4.1**。

返回值

AS_STATUS 根据执行成功或失败的情况返回不同的值。AS_STATUS.DllError 包含 DLL 返回的状态。AS_STATUS.W32Error 包含 DLL 异常相关的 Win32 错误码（如果存在的话）。关于其它可能返回的错误码，另见 **Appendix A** 。

例：

```
AS_STATUS    status;
TIMESTAMP    newTime;
TIMESTAMP    chkTime;

//用  abTime 数组的值设置 ACR89 的实时时钟值。
//假设已经创建连接
CopyMemory(newTime.szRTCValue, abTime, 6);
Status = AS_SetRTC(pDevId, &newTime, &chkTime);
```

4.2.7. 其它函数

下列函数使用户可以控制 ACR89 的 LED/蜂鸣器。

4.2.7.1. AS_SetBuzzer

启用或停用 ACR89 的蜂鸣器.

```
AS_STATUS AS_DECL AS_ SetBuzzer (
          IN INT nDevId,
          IN PBUZZER pBuzzer);
```

参数：

nDevId [in] 先前调用 AS_Open 时返回的句柄。

pBuzzer [in] 指向 BUZZER 结构的指针。该结构包含蜂鸣器待设置的状态。更多 BUZZER 结构的详细信息，另见小节 **4.1.2.10**。

返回值

AS_STATUS 根据执行成功或失败的情况返回不同的值。AS_STATUS.DllError 包含 DLL 返回的状态。AS_STATUS.W32Error 包含 DLL 异常相关的 Win32 错误码（如果存在的话）。关于其它可能返回的错误码，另见 **Appendix A** 。

例：

```
AS_STATUS   status;
BUZZER   buzStat;

//打开 ACR89 的蜂鸣器 1 秒钟
//假设已经创建连接
buzStat.cbBuzzerState = 1;
buzStat.cbBuzzerOnDuration = 10; // 1 秒
status = AS_SetBuzzer(nDevId, &buzStat);

return status;
```

4.2.7.2. AS_SetLed

启用或停用 ACR89 的 LED。

```
AS_STATUS AS_DECL AS_ SetLED (
         IN INT nDevId,
         IN PLED pLed);
```

参数：

nDevId [in] 先前调用 AS_Open 时返回的句柄。

pLED [in] 指向 LED 结构的指针。该结构包含 LED 待设置的状态。更多 LED 结构的详细信息，另见小节 **4.1.2.9**。

返回值

AS_STATUS 根据执行成功或失败的情况返回不同的值。AS_STATUS.DllError 包含 DLL 返回的状态。AS_STATUS.W32Error 包含 DLL 异常相关的 Win32 错误码（如果存在的话）。关于其它可能返回的错误码，另见 **Appendix A** 。

例：

```
AS_STATUS   status;
LED      ledStat;

//打开 LED 并换一种颜色。
//假设已经创建连接
ledStat.cbLedPower = LED_RED;     // 电源 LED 设为红色
ledStat.cbLedSlot1 = LED_GREEN;   // 插槽 1 的 LED 设为绿色
ledStat.cbLedSlot2 = LED_YELLOW; // 插槽 2 的 LED 设为黄色
status = AS_SetLED(nDevId, &ledStat);

return status;
```

附录A. 错误码(DLL 异常)

以下仅列出 DLL 异常。更多关于 Win32 异常的详情，请参考 MSDN。

错误码	错误
00h	CMD_SUCCESS
01h	CMD_WARNING_BUFFER_OVERFLOW
02h	CMD_ERROR_INVALID_OPTION
03h	CMD_ERROR_INVALID_PARAMETER
04h	CMD_ERROR_INVALID_RESPONSE_TYPE
05h	CMD_ERROR_INVALID_PARAMETER_LENGTH
06h	CMD_ERROR_LCD_INVALID_BITMAP_FILE
07h	CMD_ERROR_LCD_LOAD_BITMAP_FILE
08h	CMD_ERROR_LCD_INVALID_BITMAP_SIZE
09h	CMD_ERROR_BUFFER_TOO_SMALL
0Ah	CMD_ERROR_BUFFER_ALLOCATION_FAILED
0Bh	CMD_ERROR_COMM_PORT_OCCUPIED
0Ch	CMD_ERROR_COMM_PORT_CANNOT_OPEN
0Dh	CMD_ERROR_COMM_PORT_NOT_OPENED
0Eh	CMD_ERROR_COMM_PORT_WRITE
0Fh	CMD_ERROR_COMM_PORT_READ
10h	CMD_ERROR_COMM_DLL_GET_SYSTEMPATH
11h	CMD_ERROR_COMM_DLL_FAILED_LOAD
12h	CMD_ERROR_COMM_DLL_LOCATE_FUNCTION
13h	CMD_ERROR_COMM_DLL_GET_DEVINFO
14h	CMD_ERROR_COMM_DLL_INSUFF_BUFFER
15h	CMD_ERROR_COMM_DLL_GET_DEVDETAIL
16h	CMD_ERROR_COMM_NO_DEVICE_FOUND
17h	CMD_ERROR_SCRIPT_INVALID_FILE
18h	CMD_ERROR_SCRIPT_CANNOT_LOAD
19h	CMD_ERROR_TFM_UNSUPPORTED
20h	CMD_ERROR_SYSTEM_BUFFER_TOO_SMALL

表4： DLL 错误码