

	TECHNICAL SPECIFICATION	PTR - USB	MOD.: MP-04-10	
			EMISS.: 01-12-2000	
			EDIZ.: 01	REV.: 01
RIF.: PO-04-01				

PTR – USB

TECHNICAL REFERENCE MANUAL

	TECHNICAL SPECIFICATION	PTR - USB	MOD.: MP-04-10	
			EMISS.: 01-12-2000	
			EDIZ.: 01	REV.: 01
			RIF.: PO-04-01	

SECTION 1. FEATURES AND SPECIFICATIONS.....	3
FEATURES	3
SPECIFICATIONS	4
SECTION 2. INSTALLATION	5
USB CONNECTION	5
WINDOWS PLUG AND PLAY SETUP.....	5
SECTION 3. OPERATION.....	6
CARD READ	6
SECTION 4. USB COMMUNICATIONS.....	7
HID USAGES.....	7
REPORT DESCRIPTOR	8
CARD DATA	9
TRACK 1 DECODE STATUS	9
TRACK 2 DECODE STATUS	10
TRACK 3 DECODE STATUS	10
TRACK 1 DATA LENGTH.....	10
TRACK 2 DATA LENGTH.....	10
TRACK 3 DATA LENGTH.....	10
CARD ENCODE TYPE.....	10
TRACK DATA.....	11
TRACK 1 DATA.....	11
TRACK 2 DATA.....	11
TRACK 3 DATA.....	11
COMMANDS.....	11
COMMAND NUMBER.....	12
DATA LENGTH	12
DATA.....	12
RESULT CODE.....	12
GET AND SET PROPERTY COMMANDS.....	13
SOFTWARE_ID PROPERTY	14
SERIAL_NUM PROPERTY	15
POLLING_INTERVAL PROPERTY	16
MAX_PACKET_SIZE PROPERTY	17
TRACK_ID_ENABLE PROPERTY	18
INTERFACE_TYPE PROPERTY.....	19
RESET_DEVICE COMMAND.....	20

	TECHNICAL SPECIFICATION	PTR - USB	MOD.: MP-04-10	
			EMISS.: 01-12-2000	
			EDIZ.: 01	REV.: 01
		RIF.: PO-04-01		

SECTION 1. FEATURES AND SPECIFICATIONS

The USB Swipe Reader is a compact magnetic stripe card reader that conforms to ISO standards. The Reader is compatible with any device with a USB host interface. A card is read by sliding it past the head either forward or backward.

The Reader conforms to the USB HID Class specification Version 1.1. This allows host applications designed for most versions of Windows to easily communicate to the device using standard Windows API calls that communicate to the device through the HID driver that comes with Windows.

Unlike HID keyboard emulation readers, this device does not use keyboard emulation. It behaves like a vendor-defined HID device so that a direct communication path can be established between the Host application and the device without interference such as keystrokes from other HID devices.

FEATURES

Major features of the PTR - USB are as follows:

- Powered through the USB – no external power supply required
- Hardware Compatible with PC or any computer or terminal with a USB interface
- Bi-directional card reading
- Reads encoded data that meets ANSI/ISO/AAMVA standards and others such as ISO track 1 format on track 2 or 3
- Reads up to three tracks of card data
- Compatible with USB specification Revision 1.1
- Compatible with HID specification Version 1.1
- Can use standard Windows HID driver for communications. No third part device driver is required.
- Programmable USB serial number descriptor
- Programmable USB Interrupt In Endpoint polling interval
- Non-volatile memory for configuration storage
- Ability to convert to Keyboard Emulation mode of operation

	TECHNICAL SPECIFICATION	PTR - USB	MOD.: MP-04-10	
			EMISS.: 01-12-2000	
			EDIZ.: 01	REV.: 01
RIF.: PO-04-01				

SPECIFICATIONS

Table 1-1 lists the specifications for the USB IntelliHead.

Table 1-1. Specifications

Reference Standards	ISO 7810 and ISO 7811 and AAMVA*
Power Input	5V from USB bus
Recording Method	Two-frequency coherent phase (F2F)
Message Format	ASCII
Card Speed	3 to 60 ips (7.62 – 152.4 cm/s)

ELECTRICAL

Current	
Normal Mode	15mA
Suspend Mode	200 μ A

MECHANICAL

Weight	400 gr.
--------	---------

ENVIRONMENTAL

Temperature	
Operating	-40 oC to +70 oC (-40 oF to 158 oF)
Storage	-40 oC to +70 oC (-40 oF to 158 oF)
Humidity	
Operating	10% to 90% noncondensing
Storage	10% to 90% noncondensing
Altitude	
Operating	0-10,000 ft. (0-3048 m.)
Storage	0-50,000 ft. (0-15240 m.)

* ISO (International Standards Organization) and AAMVA (American Association of Motor Vehicle Administrators).

SECTION 2. INSTALLATION

This section describes the cable connection, the Windows Plug and Play Setup, and the physical mounting of the unit.

USB CONNECTION

Since the USB IntelliHead is supplied as an OEM product, the installation and system integration will be unique for each application. The reader module must be attached to an appropriate connector which, in turn, connects to the USB port or hub. The pin numbers for the 5-pin connector are shown in Figure 2-1.

Pin numbers and signal descriptions for the cable shown in the illustration are listed in Table 2-1.

Table 2-1. 5-Pin Connector

Pin Number	Signal	Cable Color
1	VBUS	Red
2	- Data	White
3	+Data	Green
4	Ground	Black
5	Head Case	Brown

WINDOWS PLUG AND PLAY SETUP

On hosts with the Windows operating system, the first time the device is plugged into a specific USB port, Windows will pop up a dialog box, which will guide you through the process of installing a device driver for the device. After this process is completed once, Windows will no longer request this process as long as the device is plugged into the same USB port. The device driver that Windows will install for this device is the driver used for HID devices and it is part of the Windows operating system. When the dialog box pops up, follow the instructions given in the dialog box. Sometimes Windows will find all the files it needs on its own without giving any prompts. Other times Windows will need to know the location of the files it needs. If Windows prompts for the file locations, insert the CD that was used to install Windows on your PC and point Windows to the root directory of the CD. Windows should find all the files it needs there.

	TECHNICAL SPECIFICATION	PTR - USB	MOD.: MP-04-10	
			EMISS.: 01-12-2000	
			EDIZ.: 01	REV.: 01
RIF.: PO-04-01				

SECTION 3. OPERATION

CARD READ

A card may be swiped past the read head at any time. The magnetic stripe must face toward the head and may be swiped in either direction. If there is data encoded on the card, the device will attempt to decode the data and then send the results to the host via a USB HID input report. After the results are sent to the host, the device will be ready to read the next card.

	TECHNICAL SPECIFICATION	PTR - USB	MOD.: MP-04-10	
			EMISS.: 01-12-2000	
			EDIZ.: 01	REV.: 01
			RIF.: PO-04-01	

SECTION 4. USB COMMUNICATIONS

This device conforms to the USB specification revision 1.1. This device also conforms with the Human Interface Device (HID) class specification version 1.1. The device communicates to the host as a vendor-defined HID device. The details about how the card data and commands are structured into HID reports follow later in this section. The latest versions of the Windows operating systems come with a standard Windows USB HID driver. Windows applications that communicate to this device can be easily developed. These applications can communicate to the device using standard windows API calls that communicate to the device using the standard Windows USB HID driver. These applications can be easily developed using compilers such as Microsoft's Visual Basic or Visual C++.

It is recommended that application software developers become familiar with the HID specification the USB specification before attempting to communicate with this device. This document assumes that the reader is familiar with these specifications. These specifications can be downloaded free from www.usb.org.

This is a full speed USB device. This device has a number of programmable configuration properties. These properties are stored in non-volatile memory. These properties can be configured at the factory or by the end user. The device has an adjustable endpoint descriptor polling interval value that can be set to any value in the range of 1ms to 255ms. This property can be used to speed up or slow down the card data transfer rate. The device also has an adjustable serial number descriptor. More details about these properties can be found later in this document in the command section.

The device will go into suspend mode when directed to do so by the host. The device will wake up from suspend mode when directed to do so by the host. The device does not support remote wakeup.

This device is powered from the USB bus. The vendor ID is 0x0801 and the product ID is 0x0002.

HID USAGES

HID devices send data in reports. Elements of data in a report are identified by unique identifiers called usages. The structure of the device's reports and the device's capabilities are reported to the host in a report descriptor. The host usually gets the report descriptor only once, right after the device is plugged in. The report descriptor usages identify the devices capabilities and report structures. For example, a device could be identified as a keyboard by analyzing the device's report descriptor. Usages are four byte integers. The most significant two bytes are called the usage page and the least significant two bytes are called usage IDs. Usages that are related can share a common usage page. Usages can be standardized or they can be vendor defined. Standardized usages such as usages for mice and keyboards can be found in the HID Usage Tables document and can be downloaded free at www.usb.org. Vendor-defined usages must have a usage page in the range 0xffff00 – 0xffff. All usages for this device use vendor-defined magnetic stripe reader usage page 0xffff00. The usage IDs for this device are defined in the following table. The usage types are also listed. These usage types are defined in the HID Usage Tables document.

	TECHNICAL SPECIFICATION	PTR - USB	MOD.: MP-04-10	
			EMISS.: 01-12-2000	
			EDIZ.: 01	REV.: 01
			RIF.: PO-04-01	

Magnetic Stripe Reader usage page 0xff00:

Usage ID (Hex)	Usage Name	Usage Type	Report Type
1	Decoding reader device	Collection	None
20	Track 1 decode status	Data	Input
21	Track 2 decode status	Data	Input
22	Track 3 decode status	Data	Input
28	Track 1 data length	Data	Input
29	Track 2 data length	Data	Input
2A	Track 3 data length	Data	Input
30	Track 1 data	Data	Input
31	Track 2 data	Data	Input
32	Track 3 data	Data	Input
38	Card encode type	Data	Input
20	Command message	Data	Feature

REPORT DESCRIPTOR

The HID report descriptor is structured as follows:

Item	Value (Hex)
Usage Page (Magnetic Stripe Reader)	06 00 FF
Usage (Decoding reader device)	09 01
Collection (Application)	A1 01
Logical Minimum (0)	15 00
Logical Maximum (255)	26 ff 00
Report Size (8)	75 08
Usage (Track 1 decode status)	09 20
Usage (Track 2 decode status)	09 21
Usage (Track 3 decode status)	09 22
Usage (Track 1 data length)	09 28
Usage (Track 2 data length)	09 29
Usage (Track 3 data length)	09 2A
Usage (Card encode type)	09 38
Report Count (7)	95 07
Input (Data, Variable, Absolute, Bit Field)	81 02
Usage (Track 1 data)	09 30
Report Count (110)	95 6E
Input (Data, Variable, Absolute, Buffered Bytes)	82 02 01
Usage (Track 2 data)	09 31
Report Count (110)	95 6E
Input (Data, Variable, Absolute, Buffered Bytes)	82 02 01

Usage (Track 3 data)	09 32
Report Count (110)	95 6E
Input (Data, Variable, Absolute, Buffered Bytes)	82 02 01
Usage (Command message)	09 20
Report Count (24)	95 18
Feature (Data, Variable, Absolute, Buffered Bytes)	B2 02 01
End Collection	C0

CARD DATA

Card data is only sent to the host on the Interrupt In pipe using an Input Report. The device will send only one Input Report per card swipe. If the host requests data from the device when no data is available, the device will send a Nak to the host to indicate that it has nothing to send. When a card is swiped, the Input Report will be sent even if the data is not decodable. The following table shows how the input report is structured.

Offset	Usage Name
0	Track 1 decode status
1	Track 2 decode status
2	Track 3 decode status
3	Track 1 data length
4	Track 2 data length
5	Track 3 data length
6	Card encode type
7 – 116	Track 1 data
117 – 226	Track 2 data
227 - 336	Track 3 data

TRACK 1 DECODE STATUS

Bits	7-1	0
Value	Reserved	Error

This is a one-byte value, which indicates the status of decoding track 1. Bit position zero indicates if there was an error decoding track 1 if the bit is set to one. If it is zero, then no error occurred. If a track has data on it that is not noise, and it is not decodable, then a decode error is indicated. If a decode error is indicated, the corresponding track data length value for the track that has the error will be set to zero and no valid track data will be supplied.

TRACK 2 DECODE STATUS

Bits	7-1	0
Value	Reserved	Error

This is a one-byte value, which indicates the status of decoding track 2. Bit position zero indicates if there was an error decoding track 2 if this bit is set to one. If it is zero, then no error occurred. If a track has data on it that is not noise, and it is not decodable, then a decode error is indicated. If a decode error is indicated, the corresponding track data length value for the track that has the error will be set to zero and no valid track data will be supplied.

TRACK 3 DECODE STATUS

Bits	7-1	0
Value	Reserved	Error

This is a one-byte value, which indicates the status of decoding track 3. Bit position zero indicates if there was an error decoding track 3 if this bit is set to one. If it is zero, then no error occurred. If a track has data on it that is not noise, and it is not decodable, then a decode error is indicated. If a decode error is indicated, the corresponding track data length value for the track that has the error will be set to zero and no valid track data will be supplied.

TRACK 1 DATA LENGTH

This one-byte value indicates how many bytes of decoded card data are in the track 1 data field. This value will be zero if there was no data on the track or if there was an error decoding the track.

TRACK 2 DATA LENGTH

This one-byte value indicates how many bytes of decoded card data are in the track 2 data field. This value will be zero if there was no data on the track or if there was an error decoding the track.

TRACK 3 DATA LENGTH

This one-byte value indicates how many bytes of decoded card data are in the track 3 data field. This value will be zero if there was no data on the track or if there was an error decoding the track.

CARD ENCODE TYPE

This one-byte value indicates the type of encoding that was found on the card. The following table defines the possible values.

Value	Encode Type	Description
0	ISO/ABA	ISO/ABA encode format
1	AAMVA	AAMVA encode format
2	CADL	CADL encode format. Note that this reader can only read track 2 for this format. It cannot read tracks 1 and 3. However, this format is obsolete. There should no longer be any cards in circulation that use this format. California is

	TECHNICAL SPECIFICATION	PTR - USB	MOD.: MP-04-10	
			EMISS.: 01-12-2000	
			EDIZ.: 01	REV.: 01
			RIF.: PO-04-01	

		now using the AAMVA format.
3	Blank	The card is blank.
4	Other	The card has a non-standard encode format. For example, ISO/ABA track 1 format on track 2.
5	Undetermined	The card encode type could not be determined because no tracks could be decoded
6	None	No decode has occurred. This type occurs if no magnetic stripe data has been acquired since the data has been cleared or since the device was powered on. This device only sends an Input report when a card has been swiped so this value will never occur.

TRACK DATA

If decodable track data exists for a given track, it is located in the track data field that corresponds to the track number. The length of each track data field is fixed at 110 bytes, but the length of valid data in each field is determined by the track data length field that corresponds to the track number. Track data located in positions greater than the track data length field indicates are undefined and should be ignored. The HID specification requires that reports be fixed in size, but the number of bytes encoded on a card may vary. Therefore, the Input Report always contains the maximum amount of bytes that can be encoded on the card and the number of valid bytes in each track is indicated by the track data length field. The track data is decoded and converted to ASCII. The track data includes all data starting with the start sentinel and ending with the end sentinel.

TRACK 1 DATA

This field contains the decoded track data for track 1.

TRACK 2 DATA

This field contains the decoded track data for track 2.

TRACK 3 DATA

This field contains the decoded track data for track 3.

COMMANDS

Most host applications do not need to send commands to the device. Most host applications only need to obtain card data from the device as described previously in this section. This section of the manual can be ignored by anyone who does not need to send commands to the device.

Command requests and responses are sent to and received from the device using feature reports. Command requests are sent to the device using the HID class specific request Set_Report. The response to a command is retrieved from the device using the HID class specific request Get_Report. These requests are sent over the default control pipe. When a command request is sent, the device will Nak the Status stage of the Set_Report request until the command is completed. This insures that, as soon as the Set_Report request is completed, the Get_Report request can be sent to get the command response. The usage ID for the command message was shown previously in the Usage Table.

	TECHNICAL SPECIFICATION	PTR - USB	MOD.: MP-04-10	
			EMISS.: 01-12-2000	
			EDIZ.: 01	REV.: 01
			RIF.: PO-04-01	

The following table shows how the feature report is structured for command requests:

Offset	Field Name
0	Command Number
1	Data Length
2 – 23	Data

The following table shows how the feature report is structured for command responses.

Offset	Field Name
0	Result Code
1	Data Length
2 – 23	Data

COMMAND NUMBER

This one-byte field contains the value of the requested command number. The following table lists all the existing commands.

Value	Command Number	Description
0	GET_PROPERTY	Gets a property from the device
1	SET_PROPERTY	Sets a property in the device
2	RESET_DEVICE	Resets the device

DATA LENGTH

This one-byte field contains the length of the valid data contained in the Data field.

DATA

This multi-byte field contains command data if any. Note that the length of this field is fixed at 22 bytes. Valid data should be placed in the field starting at offset 2. Any remaining data after the valid data should be set to zero. This entire field must always be set even if there is no valid data. The HID specification requires that Reports be fixed in length. Command data may vary in length. Therefore, the Report should be filled with zeros after the valid data.

RESULT CODE

This one-byte field contains the value of the result code. There are two types of result codes: generic result codes and command-specific result codes. Generic result codes always have the most significant bit set to zero. Generic result codes have the same meaning for all commands and can be used by any command. Command-specific result codes always have the most significant bit set to one. Command-specific result codes are defined by the command that uses them. The same code can have different meanings for different commands. Command-specific result codes are defined in the documentation for the command that uses them. Generic result codes are defined in the following table.

Value	Result Code	Description
0	SUCCESS	The command completed successfully.
1	FAILURE	The command failed.
2	BAD_PARAMETER	The command failed due to a bad parameter or command syntax error.

GET AND SET PROPERTY COMMANDS

The Get Property command gets a property from the device. The Get Property command number is 0.

The Set Property command sets a property in the device. The Set Property command number is 1.

The Get and Set Property command data fields for the requests and responses are structured as follows:

Get Property Request Data:

Data Offset	Value
0	Property ID

Get Property Response Data:

Data Offset	Value
0 – n	Property Value

Set Property Request Data:

Data Offset	Value
0	Property ID
1 – n	Property Value

Set Property Response Data:

None

The result codes for the Get and Set Property commands can be any of the codes list in the generic result code table.

Property ID is a one-byte field that contains a value that identifies the property. The following table lists all the current property ID values:

Value	Property ID	Description
0	SOFTWARE_ID	The device's software identifier
1	SERIAL_NUM	The device's serial number
2	POLLING_INTERVAL	The interrupt pipe's polling interval
3	MAX_PACKET_SIZE	The interrupt pipe's packet size
4	TRACK_ID_ENABLE	Allows tracks to be disabled
16	INTERFACE_TYPE	Type of USB interface

The Property Value is a multiple-byte field that contains the value of the property. The number of bytes in this field depends on the type of property and the length of the property. The following table lists all of the property types and describes them.

Property Type	Description
Byte	This is a one-byte value. The valid values depend on the property.
String	This is a multiple-byte ASCII string. Its length can be zero to a maximum length that depends on the property. The value and length of the string does not include a terminating NUL character.

SOFTWARE_ID PROPERTY

Property ID: 0
Property Type: String
Length: Fixed at 11 bytes
Get Property: Yes
Set Property: No
Description: This is an 11-byte read only property that identifies the software part number and version for the device. The first 8 bytes represent the part number and the last 3 bytes represent the version. For example this string might be "21042812D01".
Examples follow:

Example Get SOFTWARE_ID property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	00

Example Get SOFTWARE_ID property Response (Hex):

Result Code	Data Len	Prp Value
00	01	32 31 30 34 32 38 31 32 44 30 31

	TECHNICAL SPECIFICATION	PTR - USB	MOD.: MP-04-10	
			EMISS.: 01-12-2000	
			EDIZ.: 01	REV.: 01
			RIF.: PO-04-01	

SERIAL_NUM PROPERTY

Property ID:	1
Property Type:	String
Length:	0 – 15 bytes
Get Property:	Yes
Set Property:	Yes
Default Value:	The default value is no string with a length of zero.
Description:	The value is an ASCII string that represents the device's serial number. This string can be 0 – 15 bytes long. The value of this property, if any, will be sent to the host when the host requests the USB string descriptor.

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect.

Example Set SERIAL_NUM property Request (Hex):

Cmd Num	Data Len	Prp ID	Prp Value
01	04	01	31 32 33

Example Set SERIAL_NUM property Response (Hex):

Result Code	Data Len	Data
00	00	

Example Get SERIAL_NUM property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	01

Example Get SERIAL_NUM property Response (Hex):

Result Code	Data Len	Prp Value
00	03	31 32 33

POLLING_INTERVAL PROPERTY

Property ID: 2
Property Type: Byte
Length: 1 byte
Get Property: Yes
Set Property: Yes
Default Value: 10
Description: The value is a byte that represents the devices polling interval for the Interrupt In Endpoint. The value can be set in the range of 1 – 255 and has units of milliseconds. The polling interval tells the host how often to poll the device for card data packets. For example, if the polling interval is set to 10, the host will poll the device for card data packets every 10ms. This property can be used to speed up or slow down the time it takes to send card data to the host. The trade-off is that speeding up the card data transfer rate increases the USB bus bandwidth used by the device, and slowing down the card data transfer rate decreases the USB bus bandwidth used by the device. The value of this property, if any, will be sent to the host when the host requests the device’s USB endpoint descriptor.

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect.

Example Set POLLING_INTERVAL property Request (Hex):

Cmd Num	Data Len	Prp ID	Prp Value
01	02	02	0A

Example Set POLLING_INTERVAL property Response (Hex):

Result Code	Data Len	Data
00	00	

Example Get POLLING_INTERVAL property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	02

Example Get POLLING_INTERVAL property Response (Hex):

Result Code	Data Len	Prp Value
00	01	0A

MAX_PACKET_SIZE PROPERTY

Property ID: 3
Property Type: Byte
Length: 1 byte
Get Property: Yes
Set Property: Yes
Default Value: 8
Description: The value is a byte that represents the devices maximum packet size for the Interrupt In Endpoint. The value can be set in the range of 1 – 64 and has units of bytes. The maximum packet size tells the host the maximum size of the Interrupt In Endpoint packets. For example, if the maximum packet size is set to 8, the device will send HID reports in multiple packets of 8 bytes each or less for the last packet of the report. This property can be used to speed up or slow down the time it takes to send card data to the host. Larger packet sizes speed up communications and smaller packet sizes slow down communications. The trade-off is that speeding up the card data transfer rate increases the USB bus bandwidth used by the device, and slowing down the card data transfer rate decreases the USB bus bandwidth used by the device. The value of this property will be sent to the host when the host requests the device's USB endpoint descriptor.

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect.

Example Set MAX_PACKET_SIZE property Request (Hex):

Cmd Num	Data Len	Prp ID	Prp Value
01	02	03	08

Example Set MAX_PACKET_SIZE property Response (Hex):

Result Code	Data Len	Data
00	00	

Example Get MAX_PACKET_SIZE property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	03

Example Get MAX_PACKET_SIZE property Response (Hex):

Result Code	Data Len	Prp Value
00	01	08

TRACK_ID_ENABLE PROPERTY

Property ID: 4
 Property Type: Byte
 Length: 1 byte
 Get Property: Yes
 Set Property: Yes
 Default Value: 95 (hex)
 Description: This property is defined as follows:

id	0	T3	T3	T2	T2	T1	T1
----	---	----	----	----	----	----	----

Id 0 – Decodes standard ISO/ABA cards only
 1 – Decodes AAMVA and 7-bit cards also

T# 00 – Track Disabled
 01 – Track Enabled
 10 – Track Enabled/Required (Error if blank)

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect.

Example Set TRACK_ID_ENABLE property Request (Hex):

Cmd Num	Data Len	Prp ID	Prp Value
01	02	04	95

Example Set TRACK_ID_ENABLE property Response (Hex):

Result Code	Data Len	Data
00	00	

Example Get TRACK_ID_ENABLE property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	04

Example Get TRACK_ID_ENABLE property Response (Hex):

Result Code	Data Len	Prp Value
00	01	95

	TECHNICAL SPECIFICATION	PTR - USB	MOD.: MP-04-10	
			EMISS.: 01-12-2000	
			EDIZ.: 01	REV.: 01
			RIF.: PO-04-01	

INTERFACE_TYPE PROPERTY

Property ID: 16 (10 hex)

Property Type: Byte

Length: 1 byte

Get Property: Yes

Set Property: Yes

Default Value: 0 (HID)

Description: The value is a byte that represents the devices interface type. The value can be set to 0 for the HID interface or to 1 for the keyboard emulation interface. When the value is set to 0 (HID) the device will behave as described in the HID manual. When the value is set to 1 (keyboard emulation) the device will behave as described in the keyboard emulation manual. This property should be the first property changed because it affects which other properties are available. After this property is changed, the device should be power cycled before changing any other properties.

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect.

Example Set INTERFACE_TYPE property to Keyboard Emulation Request (Hex):

Cmd Num	Data Len	Prp ID	Prp Value
01	02	10	01

Example Set INTERFACE_TYPE property Response (Hex):

Result Code	Data Len	Data
00	00	

Example Get INTERFACE_TYPE property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	10

Example Get INTERFACE_TYPE property Response (Hex):

Result Code	Data Len	Prp Value
00	01	01

	TECHNICAL SPECIFICATION	PTR - USB	MOD.: MP-04-10	
			EMISS.: 01-12-2000	
			EDIZ.: 01	REV.: 01
			RIF.: PO-04-01	

RESET_DEVICE COMMAND

Command number: 2

Description: This command is used to reset the device. This command can be used to make previously changed properties take affect without having to unplug and then plug in the device. When the device resets, it automatically does a USB detach followed by an attach. After the host sends this command to the device it should close the USB port, wait a few seconds for the operating system to handle the device detach followed by the attach and then re-open the USB port before trying to communicate further with the device.

Data structure: No data is sent with this command

Result codes: 0 (success)

Example Request (Hex):

Cmd Num	Data Len	Data
02	00	

Example Response (Hex):

Result Code	Data Len	Data
00	00	