USB MAGNEPRINT SWIPE READER WITH ENCRYPTION

TECHNICAL REFERENCE MANUAL

PART NUMBER 99875338-3

MARCH 2009

MAGTEK®

REGISTERED TO ISO 9001:2000

1710 Apollo Court
Seal Beach, CA 90740
Phone: (562) 546-6400
FAX: (562) 546-6301
Technical Support: (651) 415-6800
www.magtek.com

REVISIONS

Rev Number	Date	Notes
1	5 May 06	Initial Release
2	14 Sep 07	Corrected default setting for polling interval
3	9 Mar 09	Updated MagnePrint Status; updated Warranty and Agency information

LIMITED WARRANTY

MagTek warrants that the products sold pursuant to this Agreement will perform in accordance with MagTek's published specifications. This warranty shall be provided only for a period of one year from the date of the shipment of the product from MagTek (the "Warranty Period"). This warranty shall apply only to the "Buyer" (the original purchaser, unless that entity resells the product as authorized by MagTek, in which event this warranty shall apply only to the first repurchaser).

During the Warranty Period, should this product fail to conform to MagTek's specifications, MagTek will, at its option, repair or replace this product at no additional charge except as set forth below. Repair parts and replacement products will be furnished on an exchange basis and will be either reconditioned or new. All replaced parts and products become the property of MagTek. This limited warranty does not include service to repair damage to the product resulting from accident, disaster, unreasonable use, misuse, abuse, negligence, or modification of the product not authorized by MagTek. MagTek reserves the right to examine the alleged defective goods to determine whether the warranty is applicable.

Without limiting the generality of the foregoing, MagTek specifically disclaims any liability or warranty for goods resold in other than MagTek's original packages, and for goods modified, altered, or treated without authorization by MagTek.

Service may be obtained by delivering the product during the warranty period to MagTek (1710 Apollo Court, Seal Beach, CA 90740). If this product is delivered by mail or by an equivalent shipping carrier, the customer agrees to insure the product or assume the risk of loss or damage in transit, to prepay shipping charges to the warranty service location, and to use the original shipping container or equivalent. MagTek will return the product, prepaid, via a three (3) day shipping service. A Return Material Authorization ("RMA") number must accompany all returns. Buyers may obtain an RMA number by contacting Technical Support at (888) 624-8350.

EACH BUYER UNDERSTANDS THAT THIS MAGTEK PRODUCT IS OFFERED AS IS. MAGTEK MAKES NO OTHER WARRANTY, EXPRESS OR IMPLIED, AND MAGTEK DISCLAIMS ANY WARRANTY OF ANY OTHER KIND, INCLUDING ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

IF THIS PRODUCT DOES NOT CONFORM TO MAGTEK'S SPECIFICATIONS, THE SOLE REMEDY SHALL BE REPAIR OR REPLACEMENT AS PROVIDED ABOVE. MAGTEK'S LIABILITY, IF ANY, SHALL IN NO EVENT EXCEED THE TOTAL AMOUNT PAID TO MAGTEK UNDER THIS AGREEMENT. IN NO EVENT WILL MAGTEK BE LIABLE TO THE BUYER FOR ANY DAMAGES, INCLUDING ANY LOST PROFITS, LOST SAVINGS, OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF, OR INABILITY TO USE, SUCH PRODUCT, EVEN IF MAGTEK HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, OR FOR ANY CLAIM BY ANY OTHER PARTY.

LIMITATION ON LIABILITY

EXCEPT AS PROVIDED IN THE SECTIONS RELATING TO MAGTEK'S LIMITED WARRANTY, MAGTEK'S LIABILITY UNDER THIS AGREEMENT IS LIMITED TO THE CONTRACT PRICE OF THIS PRODUCT.

MAGTEK MAKES NO OTHER WARRANTIES WITH RESPECT TO THE PRODUCT, EXPRESSED OR IMPLIED, EXCEPT AS MAY BE STATED IN THIS AGREEMENT, AND MAGTEK DISCLAIMS ANY IMPLIED WARRANTY, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

MAGTEK SHALL NOT BE LIABLE FOR CONTINGENT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES TO PERSONS OR PROPERTY. MAGTEK FURTHER LIMITS ITS LIABILITY OF ANY KIND WITH RESPECT TO THE PRODUCT, INCLUDING ANY NEGLIGENCE ON ITS PART, TO THE CONTRACT PRICE FOR THE GOODS.

MAGTEK'S SOLE LIABILITY AND BUYER'S EXCLUSIVE REMEDIES ARE STATED IN THIS SECTION AND IN THE SECTION RELATING TO MAGTEK'S LIMITED WARRANTY.

FCC WARNING STATEMENT

This equipment has been tested and was found to comply with the limits for a Class B digital device pursuant to Part 15 of FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a residential environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference with radio communications. However, there is no guarantee that interference will not occur in a particular installation.

FCC COMPLIANCE STATEMENT

This device complies with Part 15 of the FCC Rules. Operation of this device is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

CANADIAN DOC STATEMENT

This digital apparatus does not exceed the Class B limits for radio noise from digital apparatus set out in the Radio Interference Regulations of the Canadian Department of Communications.

Le présent appareil numérique n’émet pas de bruits radioélectriques dépassant les limites applicables aux appareils numériques de la classe B prescrites dans le Réglement sur le brouillage radioélectrique édicté par le ministère des Communications du Canada.

This Class B digital apparatus complies with Canadian ICES-003.

Cet appareil numériqué de la classe B est conformé à la norme NMB-003 du Canada.

CE STANDARDS

Testing for compliance with CE requirements was performed by an independent laboratory. The unit under test was found compliant with standards established for Class B devices.

UL/CSA

This product is recognized per Underwriter Laboratories and Canadian Underwriter Laboratories 1950.

RoHS STATEMENT

When ordered as RoHS compliant, this product meets the Electrical and Electronic Equipment (EEE) Reduction of Hazardous Substances (RoHS) European Directive 2002/95/EC. The marking is clearly recognizable, either as written words like “Pb-free”, “lead-free”, or as another clear symbol (Pb).

TABLE OF CONTENTS

SECTION 1. FEATURES AND SPECIFICATIONS....... 1
FEATURES....... 2
HARDWARE CONFIGURATION 2
ACCESSORIES....... 2
REFERENCE DOCUMENTS 3
SPECIFICATIONS....... 4
SECTION 2. INSTALLATION....... 7
USB CONNECTION 7
WINDOWS PLUG AND PLAY SETUP....... 8
MOUNTING 8
SECTION 3. OPERATION....... 11
LED INDICATOR 11
CARD READ....... 11
SECTION 4. USB COMMUNICATIONS....... 13
HID USAGES....... 13
MAGNETIC STRIPE READER USAGE PAGE (HID) 14
REPORT DESCRIPTOR (HID)....... 14
MAGNETIC STRIPE READER USAGE PAGE (KB)....... 16
REPORT DESCRIPTOR (KB) 17
CARD DATA (HID) 18
Track 1 Decode Status 19
Track 2 Decode Status 19
Track 3 Decode Status 19
Track 1 Data Length 19
Track 2 Data Length 19
Track 3 Data Length 19
Card Encode Type....... 19
Track Data 20
Track 1 Data 20
Track 2 Data 20
Track 3 Data 20
Card Status....... 20
MagnePrint Status 21
MagnePrint Data Length....... 21
MagnePrint Data....... 21
Device Serial Number....... 22
Sequence Counter....... 22
CARD DATA (KB)....... 22
Reader Encryption Status....... 23
PROGRAMMABLE CONFIGURATION OPTIONS 24
Low Level Communications....... 24
COMMANDS 24
COMMAND NUMBER 24
DATA LENGTH....... 25
DATA 25
RESULT CODE 25
GET AND SET PROPERTY COMMANDS 25
SOFTWARE_ID PROPERTY....... 27
USB_SERIAL_NUM PROPERTY 27
POLLING_INTERVAL PROPERTY....... 28
MAX_PACKET_SIZE PROPERTY (HID)....... 29
TRACK_ID_ENABLE PROPERTY....... 30
TRACK_DATA_SEND_FLAGS PROPERTY (KB)....... 31

TERMINATION_CHAR PROPERTY (KB) 32
SS_TK2_7BITS PROPERTY (KB) 32
SS_TK3_ISO_ABA PROPERTY (KB) 33
SS_TK3_AAMVA PROPERTY (KB) 33
SS_TK3_7BITS PROPERTY (KB) 33
PRE_CARD_CHAR PROPERTY (KB) 34
POST_CARD_CHAR PROPERTY (KB) 34
PRE_TK_CHAR PROPERTY (KB) 34
POST_TK_CHAR PROPERTY (KB) 35
ASCII_TO_KEYPRESS_CONVERSION_TYPE PROPERTY (KB) 35
INTERFACE_TYPE PROPERTY 36
ACTIVE_KEYMAP PROPERTY (KB) 37
PRE_CARD_STRING PROPERTY (KB) 38
POST_CARD_STRING PROPERTY (KB) 38
SS_TK1_ISO_ABA PROPERTY (KB) 39
SS_TK2_ISO_ABA PROPERTY (KB) 39
ES PROPERTY (KB) 40
FS PROPERTY (KB) 40
DEVICE_SERIAL_NUM PROPERTY 41
SEQUENCE_COUNTER PROPERTY 41
RESET_DEVICE COMMAND 42
GET_KEYMAP_ITEM COMMAND (KB) 42
SET_KEYMAP_ITEM COMMAND (KB) 43
SAVE_CUSTOM_KEYMAP COMMAND (KB) 45
ENCRYPTION KEYS 46
Load DUKPT Initial Key 46
Reinitialize DUKPT Key 47
Report DUKPT KSN and Counter 48
SECTION 5. DEMO PROGRAM 51
INSTALLATION 51
OPERATION 51
SOURCE CODE 52
APPENDIX A. KEYBOARD USAGE ID DEFINITIONS 53
KEYBOARD/KEYPAD PAGE (0X07) 53
APPENDIX B. MODIFIER BYTE DEFINITIONS 61
APPENDIX C. GUIDE ON DECRYPTING DATA 63

TABLES AND FIGURES

Figure 1-1. USB MagnePrint Swipe Reader with Encryption viii
Table 1-2. Specifications 4
Figure 1-2. Dimensions 5
Figure 2-1. Reader Cable and Connector 7
Table 2-1. 4-Pin Connector 7
Figure 2-2. Mounting Hole Dimensions 9
Table A-1. Keyboard/Keypad 53
Table B-1. Modifier Byte 61

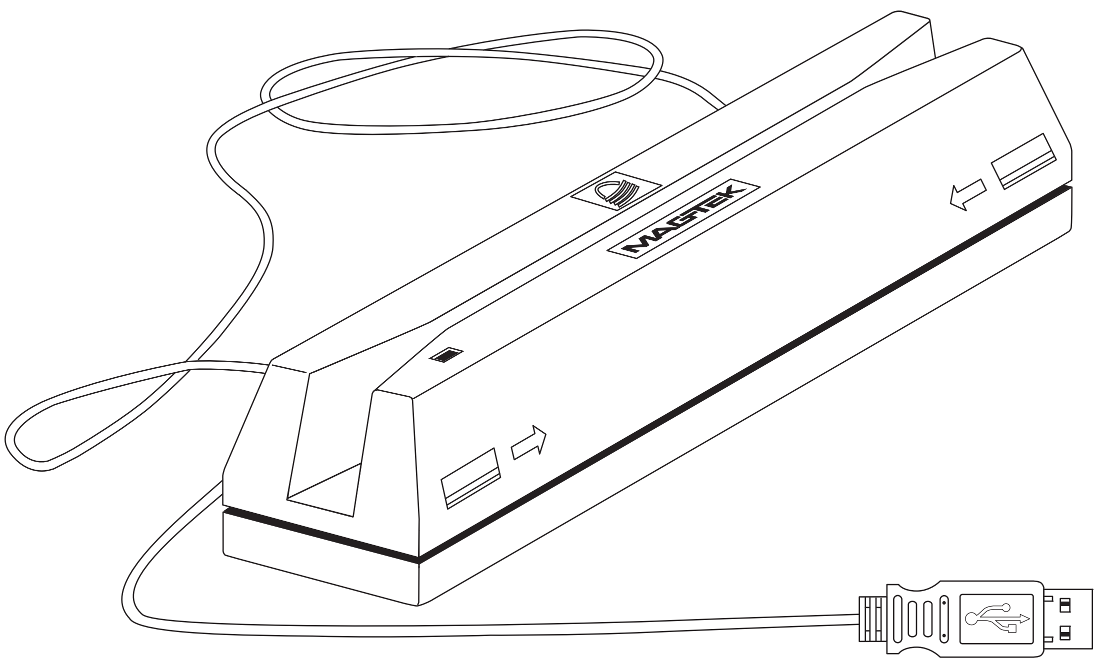


Figure 1-1. USB MagnePrint Swipe Reader with Encryption

SECTION 1. FEATURES AND SPECIFICATIONS

The USB (Universal Serial Bus) Swipe Reader is a compact magnetic stripe card reader that conforms to ISO standards. In addition to reading three tracks of data from a card, this Reader also includes MagnePrint technology. The MagnePrint data will be included with the track data on each transaction. In order to maximize card security, this model of the Reader incorporates data encryption to protect the card contents and MagnePrint information. The Reader is compatible with any device having a host USB interface. A card is read by sliding it, stripe down and facing the LED side, through the slot either forward or backward.

An LED (Light Emitting Diode) indicator on the Reader panel provides the operator with continuous status of the Reader operations.

The reader conforms to the USB HID (Human Interface Device) 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.

The Reader can be operated in two different modes:

- HID (herein referred to as "***HID*** mode") and
- HID with Keyboard Emulation (herein referred to as "***KB*** mode")

When operating in the HID mode, this device will 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 from other HID devices.

When configured for the Keyboard Emulation (KB) mode, the Reader emulates a USB HID United States keyboard or, optionally, any international keyboard using ALT ASCII code keypad key combinations or customizable key maps. This allows host applications designed to acquire card data from keyboard input to seamlessly acquire the card data from the USB swipe reader.

Caution

When in Keyboard Emulation mode, if another keyboard is connected to the same host as this device and a key is pressed on the other keyboard while this device is transmitting, then the data transmitted by this device may get corrupted.

When a card is swiped through the Reader, the track data and MagnePrint information will be TDEA (Triple Data Encryption Algorithm, aka, Triple DES) encrypted using DUKPT (Derived Unique Key Per Transaction) key management. This method of key management uses a base derivation key to encrypt a key serial number that produces an initial encryption key which is injected into the Reader prior to deployment. After each transaction, the encryption key is modified per the DUKPT algorithm so that each transaction uses a unique key. Thus, the data will be encrypted with a different encryption key for each transaction.

FEATURES

Major features of the Swipe Reader are as follows:

- Powered through the USB – no external power supply required
- Hardware Compatible with a PC or any computer or terminal having a USB interface
- Bi-directional card reading
- Reads encoded data that meets ANSI/ISO/AAMVA standards and some custom formats such as ISO track 1 format on track 2 or 3
- Reads up to three tracks of card data
- Red/Green LED for status
- Compatible with USB specification Revision 1.1
- Compatible with HID specification Version 1.1
- Can use standard Windows HID driver for communications; no third party device driver is required
- Programmable USB serial number descriptor
- Programmable USB Interrupt In Endpoint polling interval
- Programmable Keyboard Table to support alternate languages
- Non-volatile memory for property storage
- Built-in 6 foot USB cable
- Supplies 54 byte MagnePrint™ value
- Includes Device serial number and Sequence counter
- Encrypts all track data and the MagnePrint value
- Provides clear text confirmation data including card holder's name, expiration date, and a portion of the PAN

HARDWARE CONFIGURATION

The hardware configuration is as follows:

Part Number	Tracks	Configuration	Cable
21073008	**TK 1,2,3**	**Gray Full Size**	**6' USB-A**
21073023	**TK 1,2,3**	**Black Mini**	**6' USB-A**

ACCESSORIES

The accessories are as follows:

Part Number	Description
21042806	USB MSR Demo Program with Source Code (Diskette)
99510026	USB MSR Demo Program with Source Code (WEB)

REFERENCE DOCUMENTS

Axelson, Jan. *USB Complete, Everything You Need to Develop Custom USB Peripherals*, 1999. Lakeview Research, 2209 Winnebago St., Madison WI 53704, 396pp., *http://www.lvr.com*.

ANS X9.24-2004 Retail Financial Services Symmetric Key Management Part 1: Using Symmetric Techniques

USB Human Interface Device (HID) Class Specification Version 1.1.

Universal Serial Bus (USB): HID Usage Tables Version 1.12 (1/21/2005)

USB (Universal Serial Bus) Specification, Version 1.1, Copyright© 1998 by Compaq Computer Corporation, Intel Corporation, Microsoft Corporation, NEC Corporation.

USB Implementers Forum, Inc., *www.usb.org*.

SPECIFICATIONS

Table 1-2 lists the specifications for the USB Swipe Reader. Figure 1-2 shows the dimensions for the Reader.

Table 1-2. Specifications

Reference Standards	ISO 7810 and ISO 7811/ AAMVA*
Power Input	5V From USB bus
Recording Method	Two-frequency coherent phase (F2F)
Message Format	ASCII
Card Speed	4 to 60 ips (10.1 to 152.4 cm/s)
ELECTRICAL	
Current Normal Mode (including power-up) Suspend Mode	 100mA maximum 500uA maximum
MECHANICAL- Full Size	
Dimensions	Length 6.50” (165.1mm) Width 1.74” (44.2mm) Height 1.50” (38.1mm)
Weight	6.5 oz. (184.3 gr)
Cable length	6 ft.
Connector	USB Type A plug
MECHANICAL – Mini	
Dimensions	Length 3.94” (100.0mm) Width 1.28” (32.5mm) Height 1.23” (31.3mm)
Weight	4.7 oz. (133.2 gr)
Cable length	6 ft.
Connector	USB Type A plug
ENVIRONMENTAL	
Temperature	
Operating	0 °C to 70 °C (32 °F to 158 °F)
Storage	-40 °C to 70 °C (-40 °F to 158 °F)
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).

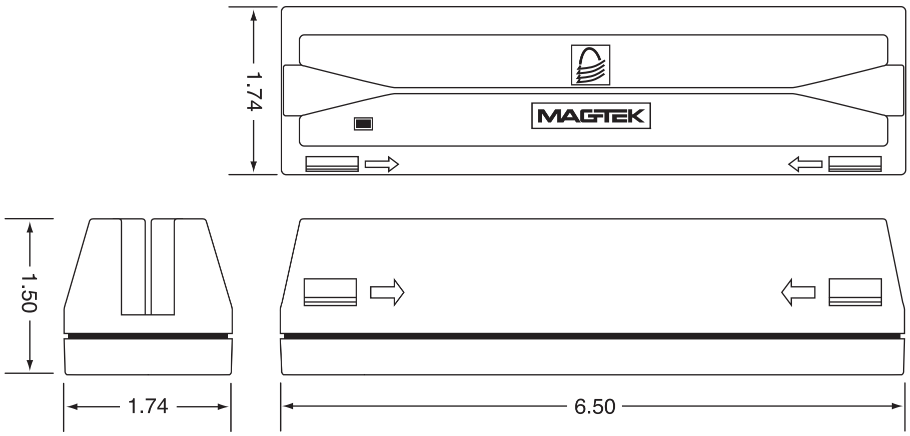


Figure 1-2. Dimensions

SECTION 2. INSTALLATION

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

USB CONNECTION

Connect the USB cable to a USB port on the host. The Reader, LED Indicator, and pin numbers for the 4-pin connector are shown in Figure 2-1.

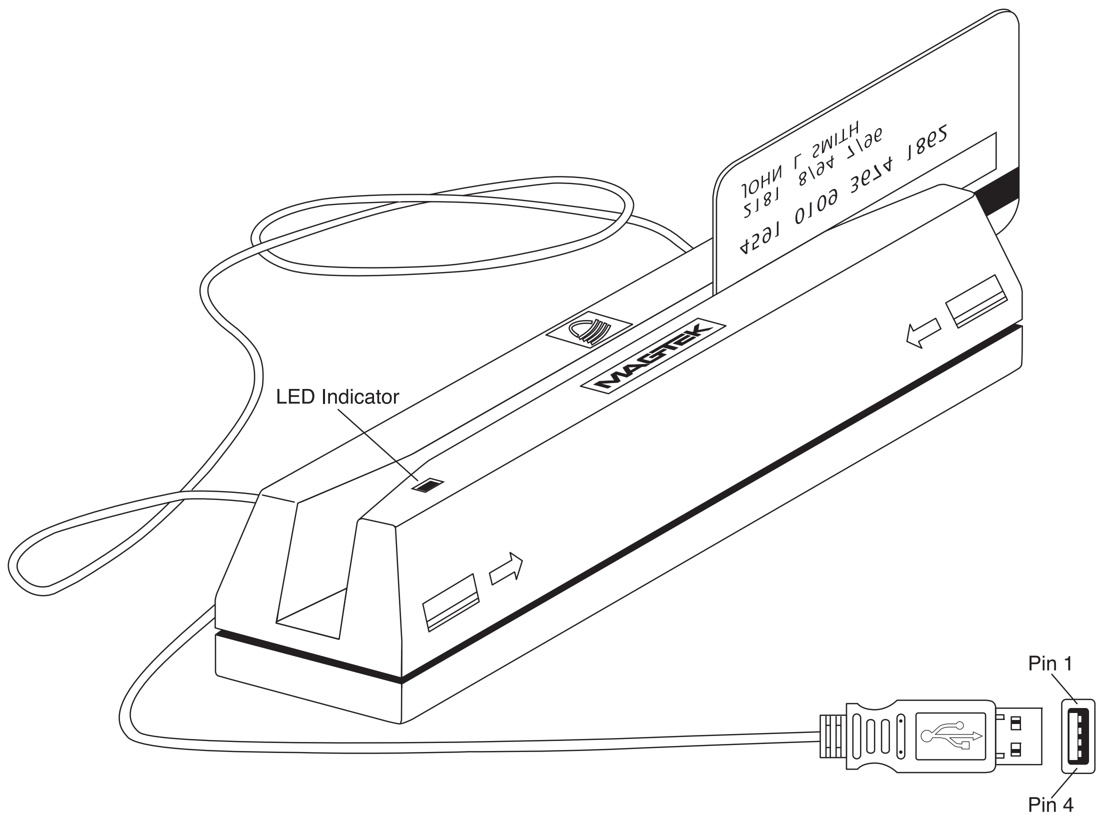


Figure 2-1. Reader Cable and Connector

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

Table 2-1. 4-Pin Connector

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

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.

MOUNTING

The Reader may be mounted with screws or fastening tape as described below.

1. The Reader can be mounted on a surface in various ways:

   - By two screws through the surface attached to the bottom of the unit and running the cable on the top of the surface
   - By two screws through the surface attached to the bottom of the unit and by drilling a hole in the surface for the cable and running the cable through the hole
   - By attaching the unit to the surface with fastening tape and running the cable on the top of the surface

   Note

   The two mounting inserts are 3mm diameter, 0.5mm pitch, 6.4mm deep. The length of the screws used depends on the mounting surface thickness and the thickness of washers (if used).

   The mounting dimensions are shown in Figure 2-2. Determine the method of mounting required.

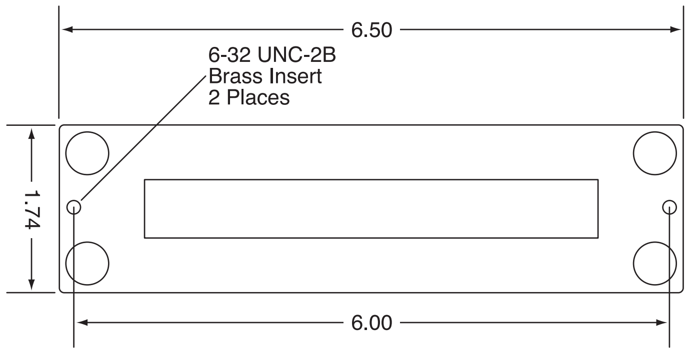


Figure 2-2. Mounting Hole Dimensions

2. Ensure the Reader is positioned on a flat, accessible surface with at least 4 inches clearance on either end for room to swipe a card. Orient the Reader so the side with the LED is facing the direction of intended use.

   If fastening tape is to be used, clean the area that the Reader will be mounted on with isopropyl alcohol. Remove the adhesive protective cover on the fastening tape, and position the Reader and push down firmly.

3. Mount the Reader.

SECTION 3. OPERATION

This section describes the LED Indicator and Card Read operation.

LED INDICATOR

The LED indicator will be either off, red, or green. When the device is not powered, the LED will be off. When the device is first plugged in, the LED will be red. As soon as the device is plugged in, the host will try to enumerate the device. Once the device is enumerated the LED will turn green indicating that the device is ready for use. When a card is being swiped, the LED will turn off temporarily until the swipe is completed. If there are no errors after decoding the card data then the LED will turn green. If there are any errors after decoding the card data, the LED will turn red for approximately two seconds to indicate that an error occurred and then turn green. Anytime the host puts the device into suspend mode, the LED will turn off. Once the host takes the device out of suspend mode, the LED will return to the state it was in prior to entering suspend mode.

The LED will blink green if the MagnePrint circuit is sensing excessive electrical noise in the environment. If this occurs, the reader will still read cards and send card data to the host until it is moved away from the noise source at which time the LED will stop blinking and stay green. When this occurs, re-position the reader away from the noise source. Note that the reader will not check for noise until after a card swipe occurs. So a card has to be swiped to initiate noise detection. If noise is detected after the swipe, the reader will continue to check for noise until the noise is no longer present. If no noise is detected after the swipe, the reader will not check for noise again until after the next swipe.

CARD READ

A card may be swiped through the Reader slot when the LED is green. The magnetic stripe must face toward the front (the side with the LED) 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 or, if in Keyboard Emulation mode, as if the data was being typed on a keyboard. After the results are sent to the host, the device will be ready to read the next card.

SECTION 4. USB COMMUNICATIONS

This device conforms to the USB specification revision 1.1. This device also conforms to the Human Interface Device (HID) class specification version 1.1. The device communicates to the host either as a vendor-defined HID device or as a HID Keyboard Emulation device. (Refer to ***Interface_Type Property*** for information on how to change modes.) The latest versions of the Windows operating system come with standard Windows USB drivers that will support both modes.

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 is a full speed USB device. It is powered from the USB bus. The vendor ID is 0x0801. The product ID is 0x000E when in the HID mode and 0x0001 when in the Keyboard Emulation mode.

Since there are two modes of operation, there are some properties and commands that are exclusive to one of the two modes. Where a property or command is unique, it will be identified with either ***HID*** or ***KB***. Properties and commands that are common to both modes do not include any modifier.

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 0xFF00 – 0xFFFF. All usages for this device use vendor-defined magnetic stripe reader usage page 0xFF00. The usage IDs for this device are defined in the following tables. The usage types are also listed. These usage types are defined in the HID Usage Tables document.

MAGNETIC STRIPE READER USAGE PAGE (HID)

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
23	MagnePrint status	Data	Input
28	Track 1 data length	Data	Input
29	Track 2 data length	Data	Input
2A	Track 3 data length	Data	Input
2B	MagnePrint data length	Data	Input
30	Track 1 data	Data	Input
31	Track 2 data	Data	Input
32	Track 3 data	Data	Input
33	MagnePrint data	Data	Input
38	Card encode type	Data	Input
39	Card status	Data	Input
40	Device serial number	Data	Input
41	Sequence counter	Data	Input
42	Reader Encryption Status	Data	Input
42	Masked PAN	Data	Input
43	Cardholder Name	Data	Input
44	Expiration Date	Data	Input
45	DUKPT serial number/counter	Data	Input
20	Command message	Data	Feature

REPORT DESCRIPTOR (HID)

The 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

Item	Value (Hex)
Report Count (7)	95 07
Input (Data, Variable, Absolute, Bit Field)	81 02
Usage (Track 1 data)	09 30
Report Count (112)	95 70
Input (Data, Variable, Absolute, Buffered Bytes)	82 02 01
Usage (Track 2 data)	09 31
Report Count (112)	95 70
Input (Data, Variable, Absolute, Buffered Bytes)	82 02 01
Usage (Track 3 data)	09 32
Report Count (112)	95 70
Input (Data, Variable, Absolute, Buffered Bytes)	82 02 01
Usage (Card status)	09 39
Report Count (1)	95 01
Input (Data, Variable, Absolute, Bit Field)	81 02
Report Size (32)	75 20
Usage (MagnePrint status)	09 23
Report Count (1)	95 01
Input (Data, Variable, Absolute, Bit Field)	81 02
Report Size (8)	75 08
Usage (MagnePrint data length)	09 2B
Report Count (1)	95 01
Input (Data, Variable, Absolute, Bit Field)	81 02
Usage (MagnePrint data)	09 33
Report Count (128)	95 80
Input (Data, Variable, Absolute, Buffered Bytes)	82 02 01
Usage (Device serial number)	09 40
Report Count (16)	95 10
Input (Data, Variable, Absolute, Buffered Bytes)	82 02 01
Usage (Sequence counter)	09 41
Report Count (8)	95 08
Input (Data, Variable, Absolute, Buffered Bytes)	82 02 01
Usage (Reader Encryption Status)	09 42
Report Count (2)	95 02
Input (Data, Variable, Absolute, Buffered Bytes)	82 02 01
Usage (Masked PAN)	09 43
Report Count (20)	95 14
Input (Data, Variable, Absolute, Buffered Bytes)	82 02 01
Usage (Cardholder Name)	09 44
Report Count (27)	95 1B
Input (Data, Variable, Absolute, Buffered Bytes)	82 02 01
Usage (Expiration Date)	09 45
Report Count (5)	95 05
Input (Data, Variable, Absolute, Buffered Bytes)	82 02 01
Usage (DUKPT Serial Number/Counter)	09 46
Report Count (10)	95 0A

Item	Value (Hex)
Input (Data, Variable, Absolute, Buffered Bytes)	82 02 01
Usage (Command Message)	09 20
Report Count (32)	95 20
Feature (Data, Variable, Absolute, Buffered Bytes)	B2 02 01
End Collection	C0

MAGNETIC STRIPE READER USAGE PAGE (KB)

Magnetic Stripe Reader usage page 0xFF00:

Usage ID (Hex)	Usage Name	Usage Type	Report Type
20	Command message	Data	Feature

REPORT DESCRIPTOR (KB)

The Report Descriptor is structured as follows:

Item	Value(Hex)
Usage Page (Generic Desktop)	05 01
Usage (Keyboard)	09 06
Collection (Application)	A1 01
Usage Page (Key Codes)	05 07
Usage Minimum (224)	19 E0
Usage Maximum (231)	29 E7
Logical Minimum (0)	15 00
Logical Maximum (1)	25 01
Report Size (1)	75 01
Report Count (8)	95 08
Input (Data, Variable, Absolute)	81 02
Report Count (1)	95 01
Report Size (8)	75 08
Input (Constant)	81 03
Report Count (5)	95 05
Report Size (1)	75 01
Usage Page (LEDs)	05 08
Usage Minimum (1)	19 01
Usage Maximum (5)	29 05
Output (Data, Variable, Absolute)	91 02
Report Count (1)	95 01
Report Size (3)	75 03
Output (Constant)	91 03
Report Count (6)	95 06
Report Size (8)	75 08
Logical Minimum (0)	15 00
Logical Maximum (101)	25 66
Usage Page (Key Codes)	05 07
Usage Minimum (0)	19 00
Usage Maximum (101)	29 66
Input (Data, Array)	81 00
Logical Maximum (255)	26 FF 00
Usage Page (vendor defined (MSR))	06 00 FF
Usage (command data)	09 20
Report Count	95 18
Feature (Data, Variable, Absolute, Buffered Bytes)	B2 02 01
End Collection	C0

CARD DATA (HID)

The details about how the card data and commands are structured into HID reports follow later in this section. 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++. A demonstration program and its source code, written in Visual Basic, that communicates with this device is available. This demo program can be used to test the device and it can be used as a guide for developing other applications. More details about the demo program follow later in this document.

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.

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 – 118	Track 1 data
119 – 230	Track 2 data
231 - 342	Track 3 data
343	Card status
344 – 347	MagnePrint status
348	MagnePrint data length
349 - 476	MagnePrint data
477 – 492	Device serial number
493 – 500	Sequence counter
501-502	Reader Encryption Status
503-522	Masked PAN
523-549	Cardholder Name
550-554	Expiration Date
555-564	DUKPT serial number/counter

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 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 112 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.

Card Status

This one byte field is reserved for future use. It is currently not used on this reader.

MagnePrint Status

This Binary field represents 32 bits of MagnePrint status information. Each character represents 4 bits (hexadecimal notation). For example, suppose the characters are: "A1050000"

Nibble	1				2				3				4				5				6				7				8			
Value	A				1				0				5				0				0				0				0			
Bit	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	23	22	21	20	19	18	17	16	31	30	29	28	27	26	25	24
Value	1	0	1	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Usage	R	R	R	R	R	R	R	M	R	R	R	R	R	R	R	R	X	X	D	X	F	L	N	S	X	X	X	X	X	X	X	X

	Meaning
R	Revision
M	MagnePrint
D	Direction
F	Fast
L	Low
N	Noisy
S	Status
X	Not Used

This four-byte field contains the MagnePrint status. The MagnePrint status is in little endian byte order. Byte 1 is the least significant byte. Byte 1 LSB is status bit 0. Byte 4 MSB is status bit 31. MagnePrint status is defined as follows:

Bit 0	=	This is a MagnePrint-capable product (usage M)
Bits 1-15	=	Product revision & mode (usage R)
Bit 16*	=	STATUS-only state (usage S)
Bit 17*	=	Noise too high or "move me" away from the noise source (used only in STATUS) (usage N)
Bit 18	=	Swipe too slow (usage L)
Bit 19	=	Swipe too fast (usage F)
Bit 20	=	Unassigned (always set to Zero)
Bit 21	=	Actual Card Swipe Direction (0 = Forward, 1 = Reverse) (usage D)
Bits 22-31	=	Unassigned (always set to Zero)

If the Enable/Disable MagnePrint property is set to disable MagnePrint, this field will not be sent.

*Bit 16 & 17 are reserved and should not be used on readers with MagneSafe V5 or later, such as these firmware versions: 21042840, 21042841, 21042846, 21042847, 21042863

MagnePrint Data Length

This one byte field indicates how many bytes of MagnePrint data are in the MagnePrint data field. This field currently only contains a value of 54.

MagnePrint Data

This 128 byte field contains the MagnePrint data. Only the number of bytes specified in the MagnePrint data length field are valid. The least significant bit of the first byte of data in this field corresponds to the first bit of MagnePrint data.

Device Serial Number

This sixteen byte field contains the device serial number. The device serial number is a NUL (zero) terminated string. So the maximum length of the device serial number, not including the null terminator, is 15 bytes. This device serial number can also be retrieved and set with the device serial number property explained in the property section of this document. This field is stored in non-volatile memory, so it will persist when the unit is power cycled.

Sequence Counter

This 8 byte field contains the sequence counter. The sequence counter is in big endian byte order. Byte 1 is the most significant byte. The first four bytes is the counter value, the last four is padding for encryption. The sequence counter is incremented by one every time a card is swiped. The sequence number cannot be reset. This sequence counter can also be retrieved with the sequence number property explained in the property section of this document. This field is stored in non-volatile memory, so it will persist when the unit is power cycled.

CARD DATA (KB)

The card data is converted to ASCII and transmitted to the host as if it had been typed on a keyboard. Any data with ASCII values 0 – 31 or 127 will be transmitted as their equivalent control code combination. For example a carriage return value 13 (0x0D) will be sent as (^M) where ^ represents the Ctrl key on the keyboard.

Caution

If another keyboard is connected to the same host as this device and a key is pressed on the other keyboard while this device is transmitting, then the data transmitted by this device may get corrupted.

The device's programmable configuration options affect the format of the card data. During normal device operation, the device acts like a USB HID keyboard so the host operating system takes care of all low level communications with the device so that the application developer is not burdened with these low level details.

All data will be sent in upper case regardless of the state of the caps lock key on the keyboard. If no data is detected on a track then nothing will be transmitted for that track. If an error is detected on a track, the ASCII character "E" will be sent in place of the track data to indicate an error.

The card data format for all programmable configuration options is as follows:

[P18] [P11] [P13][Reader Encryption Status] [Tk1 SS] [Tk1 Encrypted Data] [ES] [LRC] [P14] [P5] [P13] [Tk2 SS] [Tk2 Encrypted Data] [ES] [LRC] [P14] [P5] [P13] [Tk3 SS] [Tk3 Encrypted Data] [ES] [LRC] [P14] [P23] [MagnePrint status] [P35] [Encrypted MagnePrint data] [P35] [Device serial number] [P35] [Encrypted Sequence counter] [P35] [Masked PAN] [P35] [Cardholder Name] [P35] [Expiration date] [P35] [DUKPT serial number/counter] [P5] [P12] [P19] where:

ES	=	P22 (end sentinel)
LRC	=	Longitudinal redundancy check character
P5	=	Terminating character
P11	=	Pre card character
P12	=	Post card character
P13	=	Pre track character
P14	=	Post track character
P18	=	Pre card string
P19	=	Post card string
P35	=	Programmable field separator; this defaults to the "|" key (0x7C). Note that this key is never found in track data or the default programmable field separators.
Tk1 SS	=	P20 (ISO/ABA start sentinel)
Tk2 SS	=	P21 (ISO/ABA 5-bit start sentinel) P6 (7-bit start sentinel)
Tk3 SS	=	P8 (ISO/ABA start sentinel) P9 (AAMVA start sentinel) P10 (7-bit start sentinel)

Track 1, Track 2 and Track 3 Encrypted Data includes the Start and End Sentinel that were decoded from the card.

All fields with the format P# are programmable configuration property numbers. They are described in detail later in this document.

Reader Encryption Status

This two byte field contains the Encryption Status. The Reader Encryption Status is sent in big endian byte order. Byte 1 is the least significant byte. Byte 1 LSB is status bit 0. Byte 2 MSB is status bit 15. The Reader Encryption status is defined as follows:

Bit 0	=	Encryption Enabled (currently always set)
Bit 1	=	Initial DUKPT key Injected
Bit 2	=	DUKPT Keys exhausted
Bits 3- 15	=	Unassigned (always set to Zero)

Notes:

(1) Encryption will only be performed when Encryption Enabled and Initial DUKPT key Injected are set. Otherwise, data that are normally encrypted are sent in the clear in ASCII HEX format; the DUKPT Serial Number/counter will not be sent.

(2) When DUKPT Keys Exhausted is set, the reader will no longer read cards and after a card swipe, the reader response will be sent as follows:
[P18] [P11] [P13] [Reader Encryption Status] [P5] [P12] [P19]

PROGRAMMABLE CONFIGURATION OPTIONS

This device has a number of programmable configuration properties. Most of the programmable properties deal with the Keyboard Emulation mode but some of the properties deal with the reader regardless of the mode. These properties are stored in non-volatile memory. These properties can be configured at the factory or by the end user using a program supplied by MagTek. Programming these parameters requires low level communications with the device. Details on how to communicate with the device to change programmable configuration properties follows in the next few sections. These details are included as a reference only. Most users will not need to know these details because the device will be configured at the factory or by a program supplied by MagTek. Most users may want to skip over the next few sections on low level communications and continue with the details of the configuration properties.

Low Level Communications

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

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.

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
3	GET_KEYMAP_ITEM	Gets a key map item (KB only)
4	SET_KEYMAP_ITEM	Sets a key map item (KB only)
5	SAVE_CUSTOM_KEYMAP	Saves the custom key map (KB only)
7	LOAD DUKPT INITIAL KEY	Loads the initial DUKPT Key scheme
8	REINITIALIZE DUKPT KEY	Reinitializes the DUKPT Key scheme
9	GET_DUKPT_KSN	Reports DUKKPT KSN and Counter

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 HID mode	Value KB mode	Property ID	Description
0	0	SOFTWARE_ID	The device's software identifier
1	1	SERIAL_NUM	The device's serial number
2	2	POLLING_INTERVAL	The interrupt pipe's polling interval
3	-	MAX_PACKET_SIZE	The interrupt pipe's packet size
4	3	TRACK_ID_ENABLE	Track enable / ID enable
-	4	TRACK_DATA_SEND_FLAGS	Track data send flags
-	5	TERMINATION_CHAR	Terminating char / per track or card flag
-	6	SS_TK2_7BITS	Start sentinel char for track 2 – 7 bit data
-	7	Reserved for future use	
-	8	SS_TK3_ISO_ABA	Start sentinel char for track 3 – ISO/ABA
-	9	SS_TK3_AAMVA	Start sentinel char for track 3 - AAMVA
-	10	SS_TK3_7BITS	Start sentinel char for track 3 – 7 bit data
-	11	PRE_CARD_CHAR	Pre card char
-	12	POST_CARD_CHAR	Post card char
-	13	PRE_TK_CHAR	Pre track char
-	14	POST_TK_CHAR	Post track char
-	15	ASCII_TO_KEYPRESS_CONVERSION_TYPE	Type of conversion performed when converting ASCII data to key strokes
16	16	INTERFACE_TYPE	Type of USB interface
-	17	ACTIVE_KEYMAP	Selects which key map to uses
-	18	PRE_CARD_STRING	Pre card string
-	19	POST_CARD_STRING	Post card string
-	20	SS_TK1_ISO_ABA	Start sentinel char for track 1 – ISO/ABA
-	21	SS_TK2_ISO_ABA	Start sentinel char for track 2 – ISO/ABA
-	22	ES	End sentinel char for all tracks/formats
-	35	FS	Field Separator for additional data

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

USB_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 USB 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. This device must be unplugged for at least 30 seconds to properly power cycle it.

Example Set USB_SERIAL_NUM property Request (Hex):

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

Example Set USB_SERIAL_NUM property Response (Hex):

Result Code	Data Len	Data
00	00	

Example Get USB_SERIAL_NUM property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	01

Example Get USB_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: 1 for Keyboard Emulation interface type or 10 (0A hex) for HID interface type
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 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. This device must be unplugged for at least 30 seconds to properly power cycle it.

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 (HID)

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. This device must be unplugged for at least 30 seconds to properly power cycle it.

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: 3 (KB mode) or 4 (HID mode)
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	T_3	T_3	T_2	T_2	T_1	T_1

Id 0 – Decodes standard ISO/ABA cards only
1 – Decodes AAMV 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. This device must be unplugged for at least 30 seconds to properly power cycle it.

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

TRACK_DATA_SEND_FLAGS PROPERTY (KB)

Property ID: 4
Property Type: Byte
Length: 1 byte
Get Property: Yes
Set Property: Yes
Default Value: 63 (hex)
Description: This property is defined as follows:

ICL	SS	ES	LRC	0	LC	Er	Er

ICL 0 – Changing the state of the caps lock key will not affect the case of the data
1 – Changing the state of the caps lock key will affect the case of the data

SS 0 – Don't send Start Sentinel for each track
1 – Send Start Sentinel for each track

ES 0 – Don't send End Sentinel for each track
1 – Send End Sentinel for each track

LRC 0 – Don't send LRC for each track
1 – Send LRC for each track

Note that the LRC is the unmodified LRC from the track data. To verify the LRC the track data needs to be converted back from ASCII to card data format and the start sentinels that were modified to indicate the card encode type need to be converted back to their original values.

LC 0 – Send card data as upper case
1 – Send card data as lower case

Note that the state of the Caps Lock key on the host keyboard has no affect on what case the card data is transmitted in unless the ICL bit in this property is set to 1.

Er 00 – Don't send any card data if error
01 – Don't send track data if error
11 – Send 'E' for each track error

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.

TERMINATION_CHAR PROPERTY (KB)

Property ID: 5
Property Type: Byte
Length: 1 byte
Get Property: Yes
Set Property: Yes
Default Value: 0D (hex) (carriage return)
Description: This property is defined as follows:

mod	c	c	c	c	c	C	c

mod 0 – Send c after card data
1 – Send c after each track

c 1-127 – 7 bit ASCII char code
0 – send nothing

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.

SS_TK2_7BITS PROPERTY (KB)

Property ID: 6
Property Type: Byte
Length: 1 byte
Get Property: Yes
Set Property: Yes
Default Value: 40 (hex) '@'
Description: This character is sent as the track 2 start sentinel for cards that have track 2 encoded in 7 bits per character format. If the value is 0 no character is sent. If the value is in the range 1 – 127 then the equivalent ASCII character will be sent.

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.

SS_TK3_ISO_ABA PROPERTY (KB)

Property ID: 8
Property Type: Byte
Length: 1 byte
Get Property: Yes
Set Property: Yes
Default Value: 2B (hex) '+'
Description: This character is sent as the track 3 start sentinel for cards that have track 3 encoded in ISO/ABA format. If the value is 0 no character is sent. If the value is in the range 1 – 127 then the equivalent ASCII character will be sent.

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.

SS_TK3_AAMVA PROPERTY (KB)

Property ID: 9
Property Type: Byte
Length: 1 byte
Get Property: Yes
Set Property: Yes
Default Value: 23 (hex) '#'
Description: This character is sent as the track 3 start sentinel for cards that have track 3 encoded in AAMVA format. If the value is 0 no character is sent. If the value is in the range 1 – 127 then the equivalent ASCII character will be sent.

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.

SS_TK3_7BITS PROPERTY (KB)

Property ID: 10 (0x0A)
Property Type: Byte
Length: 1 byte
Get Property: Yes
Set Property: Yes
Default Value: 26 (hex) '&'
Description: This character is sent as the track 3 start sentinel for cards that have track 3 encoded in 7 bits per character format. If the value is 0 no character is sent. If the value is in the range 1 – 127 then the equivalent ASCII character will be sent.

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.

PRE_CARD_CHAR PROPERTY (KB)

Property ID:	11 (0x0B)
Property Type:	Byte
Length:	1 byte
Get Property:	Yes
Set Property:	Yes
Default Value:	0
Description:	This character is sent prior to all other card data. If the value is 0 no character is sent. If the value is in the range 1 – 127 then the equivalent ASCII character will be sent. 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.

POST_CARD_CHAR PROPERTY (KB)

Property ID:	12 (0x0C)
Property Type:	Byte
Length:	1 byte
Get Property:	Yes
Set Property:	Yes
Default Value:	0
Description:	This character is sent after all other card data. If the value is 0 no character is sent. If the value is in the range 1 – 127 then the equivalent ASCII character will be sent. 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.

PRE_TK_CHAR PROPERTY (KB)

Property ID:	13 (0x0D)
Property Type:	Byte
Length:	1 byte
Get Property:	Yes
Set Property:	Yes
Default Value:	0
Description:	This character is sent prior to the data for each track. If the value is 0 no character is sent. If the value is in the range 1 – 127 then the equivalent ASCII character will be sent. 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.

POST_TK_CHAR PROPERTY (KB)

Property ID:	14 (0x0E)
Property Type:	Byte
Length:	1 byte
Get Property:	Yes
Set Property:	Yes
Default Value:	0
Description:	This character is sent after the data for each track. If the value is 0 no character is sent. If the value is in the range 1 – 127 then the equivalent ASCII character be sent. 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.

ASCII_TO_KEYPRESS_CONVERSION_TYPE PROPERTY (KB)

Property ID:	15 (0x0F)
Property Type:	Byte
Length:	1 byte
Get Property:	Yes
Set Property:	Yes
Default Value:	0 (keymap)
Description:	The value is a byte that represents the devices ASCII to keypress conversion type. The value can be set to 0 for keymap (the active keymap is set with the ACTIVE_KEYMAP property) or to 1 for ALT ASCII code (international keyboard emulation). When the value is set to 0 (keymap), data will be transmitted to the host according to the active keymap which defaults to the United States keyboard keymap. For example, to transmit the ASCII character '?' (063 decimal), the character is looked up in a keymap. For a United States keyboard keymap, the '/' (forward slash) key combined with the left shift key modifier are stored in the keymap to represent the key press combination that is used to represent the ASCII character '?' (063 decimal). When the value is set to 1 (ALT ASCII code), instead of using the key map, a international keyboard key press combination consisting of the decimal value of the ASCII character combined with the ALT key modifier is used. For example, to transmit the ASCII character '?' (063 decimal), keypad '0' is sent combined with left ALT key modifier, next keypad '6' is sent combined with the left ALT key modifier, last keypad '3' is sent combined with the left ALT key modifier. In general, if this device only needs to emulate United States keyboards then this property should be set to 0 (keymap). If this device needs to be able to emulate all country's keyboards then this property should be set to 1 (ALT ASCII code). The tradeoff is that the ALT ASCII code mode is slightly slower than keymap mode because more key presses need to be transmitted. Some applications are not compatible with ALT ASCII code mode.

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 ASCII_TO_KEYPRESS_CONVERSION_TYPE property Request (Hex):

Cmd Num	Data Len	Prp ID	Prp Value
01	02	0F	00

Example Set ASCII_TO_KEYPRESS_CONVERSION_TYPE property Response (Hex):

Result Code	Data Len	Data
00	00	

Example Get ASCII_TO_KEYPRESS_CONVERSION_TYPE property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	0F

Example Get ASCII_TO_KEYPRESS_CONVERSION_TYPE property Response (Hex):

Result Code	Data Len	Prp Value
00	01	00

INTERFACE_TYPE PROPERTY

Property ID: 16 (0x10)
Property Type: Byte
Length: 1 byte
Get Property: Yes
Set Property: Yes
Default Value: 1 (keyboard emulation)
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 HID” Request (Hex):

Cmd Num	Data Len	Prp ID	Prp Value
01	02	10	00

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	00

ACTIVE_KEYMAP PROPERTY (KB)

Property ID: 17 (0x11)
Property Type: Byte
Length: 1 byte
Get Property: Yes
Set Property: Yes
Default Value: 0 (United States)
Description: The value is a byte that represents the device's active key map. The value can be set to 0 for the United States key map or to 1 for the custom key map. The active key map will be used by the device to convert ASCII data into key strokes. The United States key map should be used will all hosts that are configured to use United States keyboards. The custom key map can be used to set up the device to work with hosts that are configured to use other countries keyboards. The default custom key map is the same as the United States key map. The key map can be modified to another countries key map by using commands "Get Key Map", "Set Key Map" and "Save Custom Key Map". See the command section of this manual for a complete description of these commands. To set up a device to use a custom key map, select the appropriate key map to be modified using the active key map property, reset the device to make this change take affect, use the "Get Key Map" and "Set Key Map" commands to modify the active key map, use the "Save Custom Key Map" command to save the active key map as the custom key map, set the active key map property to custom to use the custom key map, reset the device to make these changes take affect.

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 ACTIVE_KEYMAP property Request (Hex):

Cmd Num	Data Len	Prp ID	Prp Value
01	02	11	00

Example Set ACTIVE_KEYMAP property Response (Hex):

Result Code	Data Len	Data
00	00	

Example Get ACTIVE_KEYMAP property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	11

Example Get ACTIVE_KEYMAP property Response (Hex):

Result Code	Data Len	Prp Value
00	01	00

PRE_CARD_STRING PROPERTY (KB)

Property ID: 18 (0x12)
Property Type: String
Length: 0 – 7 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 pre card string. This string can be 0 – 7 bytes long. This string is sent prior to all other card data.

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 PRE_CARD_STRING property Request (Hex):

Cmd Num	Data Len	Prp ID	Prp Value
01	04	12	31 32 33

Example Set PRE_CARD_STRING property Response (Hex):

Result Code	Data Len	Data
00	00	

Example Get PRE_CARD_STRING property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	12

Example Get PRE_CARD_STRING property Response (Hex):

Result Code	Data Len	Prp Value
00	03	31 32 33

POST_CARD_STRING PROPERTY (KB)

Property ID: 19 (0x13)
Property Type: String
Length: 0 – 7 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 post card string. This string can be 0 – 7 bytes long. This string is sent after all other card data.

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 POST_CARD_STRING property Request (Hex):

Cmd Num	Data Len	Prp ID	Prp Value
01	04	12	31 32 33

Example Set POST_CARD_STRING property Response (Hex):

Result Code	Data Len	Data
00	00	

Example Get POST_CARD_STRING property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	12

Example Get POST_CARD_STRING property Response (Hex):

Result Code	Data Len	Prp Value
00	03	31 32 33

SS_TK1_ISO_ABA PROPERTY (KB)

Property ID: 20 (0x14)
Property Type: Byte
Length: 1 byte
Get Property: Yes
Set Property: Yes
Default Value: 0x25 '%'
Description: This character is sent as the track 1 start sentinel for cards that have track 1 encoded in ISO/ABA format. If the value is 0 no character is sent. If the value is in the range 1 – 127 then the equivalent ASCII character will be sent.

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.

SS_TK2_ISO_ABA PROPERTY (KB)

Property ID: 21 (0x15)
Property Type: Byte
Length: 1 byte
Get Property: Yes
Set Property: Yes
Default Value: 0x3B ';'
Description: This character is sent as the track 2 start sentinel for cards that have track 2 encoded in ISO/ABA format. If the value is 0 no character is sent. If the value is in the range 1 – 127 then the equivalent ASCII character will be sent.

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.

ES PROPERTY (KB)

Property ID: 22 (0x16)
Property Type: Byte
Length: 1 byte
Get Property: Yes
Set Property: Yes
Default Value: 0x3F '?'
Description: This character is sent as the end sentinel for all tracks with any format. If the value is 0 no character is sent. If the value is in the range 1 – 127 then the equivalent ASCII character will be sent.

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.

FS PROPERTY (KB)

Property ID: 35 (0x23)
Property Type: Byte
Length: 1 byte
Get Property: Yes
Set Property: Yes
Default Value: 0x7C '|'
Description: This character is sent as the field separator to delimit additional data (MagnePrint info, device info, DUKPT info, etc.). If the value is 0 no character is sent. If the value is in the range 1 – 127 then the equivalent ASCII character will be sent.

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.

DEVICE_SERIAL_NUM PROPERTY

Property ID: 32 (0x20)
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 serial number. This string can be 0 – 15 bytes long. The value of this property, if any, will be sent to the host in the device serial number field of the USB input report when a card is swiped. This is explained in the card data section of this document.

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. This device must be unplugged for at least 30 seconds to properly power cycle it.

Example Set DEVICE_SERIAL_NUM property Request (Hex):

Cmd Num	Data Len	Prp ID	Prp Value
01	04	20	31 32 33

Example Set DEVICE_SERIAL_NUM property Response (Hex):

Result Code	Data Len	Data
00	00	

Example Get DEVICE_SERIAL_NUM property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	20

Example Get DEVICE_SERIAL_NUM property Response (Hex):

Result Code	Data Len	Prp Value
00	03	31 32 33

SEQUENCE_COUNTER PROPERTY

Property ID: 33 (0x21)
Property Type: Double Word
Length: 4 bytes
Get Property: Yes
Set Property: No
Default Value: 0
Description: This 4 byte field contains the sequence counter. The sequence counter is in little endian byte order. Byte 1 is the least significant byte. The sequence counter is incremented by one every time a card is swiped. The sequence number can not be reset.

This property is stored in non-volatile memory, so it will persist when the unit is power cycled.

Example Get SEQUENCE_COUNTER property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	21

Example Get SEQUENCE_COUNTER property Response (Hex):

Result Code	Data Len	Prp Value
00	04	02 01 00 00 (counter is 258 decimal)

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	

GET_KEYMAP_ITEM COMMAND (KB)

Command number: 3

Description: This command is used to get a key map item from the active key map. The active key map is determined by the active key map property. Data from a magnetic stripe card is a sequence of ASCII characters. These ASCII characters are mapped to key strokes and these key strokes are sent to the host to represent the ASCII character. The key map maps a single ASCII character to a single USB key usage ID and USB key modifier byte. The key usage ID and the key modifier byte are transmitted to the host via USB to represent the ASCII character. The ASCII value is the value of the ASCII character to be transmitted to the host. See an ASCII table for the values of the ASCII character set. The USB key usage ID is a unique value assigned to every keyboard key. For a list of all key usage IDs see Appendix A. The key modifier byte modifies the meaning of the key usage ID. The modifier byte indicates if any combination of the right or left Ctrl, Shift, Alt or GUI keys are pressed at the same time as the key usage ID. For a list and description of the key modifier byte see Appendix B.

Starting with the firmware release with software ID 21042812F01, when both the key usage ID and the key modifier byte are set to 0xFF for a given ASCII value, the ALT ASCII code is sent instead of the key map values. The ALT ASCII code is a key press combination consisting of the decimal value of the ASCII character combined with the ALT key modifier. For example, to transmit the ASCII character '?' (063 decimal), keypad '0' is sent combined with left ALT key modifier, next keypad '6' is sent combined with the left ALT key modifier, last keypad '3' is sent combined with the left ALT key modifier.

Data structure:

Request Data:

Offset	Field Name	Description
0	ASCII value	Value of the ASCII character to be retrieved from the key map. This can be any value between 0 and 127 (0x7F). For example, to retrieve the key map item for ASCII character '?' (card data end sentinel) use the ASCII value of '?' which is 63 (0x3F).

Response Data:

Offset	Field Name	Description
0	Key Usage ID	The value of the USB key usage ID that is mapped to the given ASCII value. For example, for the United States keyboard map, usage ID 56 (0x38) (keyboard / and ?) is mapped to ASCII character '?'.
1	Key Modifier Byte	The value of the USB key modifier byte that is mapped to the given ASCII value. For example, for the United States keyboard map, modifier byte 0x02 (left shift key) is mapped to ASCII character '?'.

Result codes: 0 (success)

Example Request (Hex):

Cmd Num	Data Len	Data
03	01	3F

Example Response (Hex):

Result Code	Data Len	Data
00	02	38 02

SET_KEYMAP_ITEM COMMAND (KB)

Command number: 4

Description: This command is used to set a key map item of the active key map. The active key map is determined by the active key map property. Data from a magnetic stripe card is a sequence of ASCII characters. These ASCII characters are mapped to key strokes and these key strokes are sent to the host to represent the ASCII character. The key map maps a single ASCII character to a single USB key usage ID and USB key modifier byte. The key usage ID and the key modifier byte are transmitted to the host via USB to represent the ASCII character. The ASCII value is the value of the ASCII character to be transmitted to the host. See an ASCII table for the values of the ASCII character set. The USB key usage ID is a unique value assigned to every keyboard key. For a list of all key usage IDs see Appendix A. The key modifier byte modifies the meaning of the key usage ID. The modifier byte indicates if any combination of the right or left Ctrl, Shift, Alt or GUI keys are pressed at the same time as the key usage ID. For a list and description of the key modifier byte see Appendix B. Once a key map item is modified, the changes take affect immediately. However, the changes will be lost if the device is reset or power cycled. To make the changes permanent, the save custom key map command must be issued. To use the new custom key map after a reset or power cycle, the active key map property must be set to custom.

Starting with the firmware release with software ID 21042812F01, when both the key usage ID and the key modifier byte are set to 0xFF for a given ASCII value, the ALT ASCII code is sent instead of the key map values. The ALT ASCII code is a key press combination consisting of the decimal value of the ASCII character combined with the ALT key modifier. For example, to transmit the ASCII character '?' (063 decimal), keypad '0' is sent combined with left ALT key modifier, next keypad '6' is sent combined with the left ALT key modifier, last keypad '3' is sent combined with the left ALT key modifier.

Data structure:

Request Data:

Offset	Field Name	Description
0	ASCII value	Value of the ASCII character to be set in the key map. This can be any value between 0 and 127 (0x7F). For example, to set the key map item for ASCII character '?' (card data end sentinel) use the ASCII value of '?' which is 63 (0x3F).
1	Key Usage ID	The value of the USB key usage ID that is to be mapped to the given ASCII value. For example, for the United States keyboard map, usage ID 56 (0x38) (keyboard / and ?) is mapped to ASCII character '?'. To change this to the ASCII character '>' use usage ID 55 (0x37) (keyboard . and >).

Offset	Field Name	Description
2	Key Modifier Byte	The value of the USB key modifier byte that is to be mapped to the given ASCII value. For example, for the United States keyboard map, modifier byte 0x02 (left shift key) is mapped to ASCII character '?'. To change this to the ASCII character '>' use modifier byte 0x02 (left shift key).

Response Data: None

Result codes: 0 (success)

The following example maps the card ASCII data end sentinel character '?' to the '>' keyboard key.

Example Request (Hex):

Cmd Num	Data Len	Data
04	03	3F 37 02

Example Response (Hex):

Result Code	Data Len	Data
00	00	

SAVE_CUSTOM_KEYMAP COMMAND (KB)

Command number: 5

Description: This command is used to save the active key map as the custom key map in non volatile memory. The active key map is determined by the active key map property. Once a key map item is modified, the changes take affect immediately. However, the changes will be lost if the device is reset or power cycled. To make the changes permanent, the save custom key map command must be issued. To use the new custom key map after a reset or power cycle, the active key map property must be set to custom.

Data structure:

Request Data: None
Response Data: None

Result codes: 0 (success)

Example Request (Hex):

Cmd Num	Data Len	Data
05	00	

Example Response (Hex):

Result Code	Data Len	Data
00	00	

ENCRYPTION KEYS

Load DUKPT Initial Key

This command should only be used in a secure environment.

Command number: 7

Description: This command is used in the Derived Unique Key Per Transaction (DUKPT) Key Management scheme to load the initial key (as two components) in the clear. This command may be used multiple times. Each use completely initializes the DUKPT Key Management scheme, losing all information about the previous scheme.

This command has two parts and the key is not loaded until the second part is executed.

- The first part loads one of the components of the key; the second part loads the other component.
- The first component must be entered first; the second component must be entered within two minutes of the first part.
- There must be no loss of power to the device between the entry of the first and second components.
- The two components are combined by XORing in the unit to create the final key.
- On receipt of the correctly formatted first part, the DUKPT Key Management scheme is initialized, losing all information about previous DUKPT keys, and the new first component is stored in secure memory in anticipation of receipt of the second component.
- On receipt of the second component, both components are combined by XORing and the DUKPT Key Management scheme is completely initialized.

Data structure:

Request Data: First Part:

Offset	Field Name	Description
0	Part Number	Part Number, always a 1
1	Initial Key Component (first part)	This component must be 16 bytes long.

Request Data: Second Part:

Offset	Field Name	Description
0	Part Number	Part Number, always a 2
1	Key Serial Number Register.	This eighty-bit field includes the Initial Key Serial Number in the leftmost 59 bits and a value for the Encryption Counter in the rightmost 21 bits. The value for the Encryption Counter must be 0.
11	Initial Key Component (second part)	This component must be 16 bytes long.

Response Data: None

Result codes: 0x00 (success)
0x02 (Bad Parameters) – The Request Data is not a correct length.
0x95 – First part not loaded (happens only when trying to load second part).

Example Request (Hex): Part 1 (The spaces between bytes are provided for visual clarity; they are not part of the command.)

Cmd Num	Data Len	Data
07	11	01 0F0F 0F0F 0F0F 0F0F 0F0F 0F0F 0F0F 0F0F

Example Request (Hex): Part 2

Cmd Num	Data Len	Data
07	1B	02 FFFF 9876 5432 10E0 0000 65CD 9DF5 AE3E 5442 8A85 BCAC D8DA 9C35

Example Response (Hex):

Result Code	Data Len	Data
00	00	None

Reinitialize DUKPT Key

Command number: 8

Description: This command is used in the Derived Unique Key Per Transaction (DUKPT) Key Management scheme to load a new initial PIN encryption key and/or a new Key Serial Number while the device is in service. This feature allows:

1) Extension of the service life beyond the one million transaction limit.
2) Changing from use of one acquirer's derivation key to another's.
3) Recovery from possible compromise of a derivation key.

This command may be used multiple times. Each use completely initializes the DUKPT Key Management scheme, losing all information about the previous scheme.

The Reader uses the current encryption key to perform the inverse “Triple-DES” function on the encrypted new initial encryption key. This provides the Clear Text new initial encryption key. This key is then used to encrypt, via the “Triple-DES” function, the new key serial number (excluding the 16 rightmost bits). If the leftmost 32 bits of this result match the Check Value, the device performs the initialization and uses the new initial encryption key as the “initial encryption key” and the new Key Serial Number as the Key Serial Number.

If the load is successful, the current key serial number will be based on the new key serial number as requested. If the load is not successful, the current key serial number will not be changed.

This message is secure against “man in the middle” attacks. If any part of the message is modified, the device cannot be used with the intended host. Replay of a message will fail because the encrypted new key will not decrypt correctly (a different key is in the unit at this time).

Data structure:

Request Data:

Offset	Field Name	Description
0	New Key Serial Number (Hex)	Same as for the Load Initial DUKPT Command
10	Key Check Value	Used to validate the new Key is received correctly.
14	New Initial Key	This key must be 16 bytes long.

Response Data:

Offset	Field Name	Description
0	Current Key Serial Number	This eighty-bit field includes the Initial Key Serial Number in the leftmost 59 bits and a value for the Encryption Counter in the rightmost 21 bits.

Result codes:
0x00 (success)
0x02 (Bad Parameters) – The Request Data is not a correct length.
0x84 – There is no current key (for decrypting the new key).
0x93 – Check Value mismatch.

Example Request (Hex): Part 1

Cmd Num	Data Len	Data
08	1E	FFFF 9876 5432 10E0 0000 0102 0304 6AC2 92FA A131 5B4D 858A B3A3 D7D5 933A

Example Response (Hex):

Result Code	Data Len	Data
00	0A	FFFF 9876 5432 10E0 0000

Report DUKPT KSN and Counter

Command number: 9

Description: This command is used to report the Key Serial Number and Encryption Counter.

Data structure: No data is sent with this command.

Response Data:

Offset	Field Name	Description
0	Current Key Serial Number	This eighty-bit field includes the Initial Key Serial Number in the leftmost 59 bits and a value for the Encryption Counter in the rightmost 21 bits.

Result codes: 0x00 (success)
0x02 (Bad Parameters) – The Request Data is not a correct length.

Example Request (Hex):

Cmd Num	Data Len	Data
09	0	none

Example Response (Hex):

Result Code	Data Len	Data
00	0A	FFFF 9876 5432 10E0 0001

SECTION 5. DEMO PROGRAM

The demo program, which is written in Visual Basic, can be used to do the following:

- Send command requests to the device and view the command responses.
- Guide application developers in their application development by providing examples, in source code, of how to properly communicate with the device using the standard Windows APIs.
- Read cards from the device and view the card data (HID mode only).

The part numbers for the demo program can be found in this document in Section 1 under Accessories.

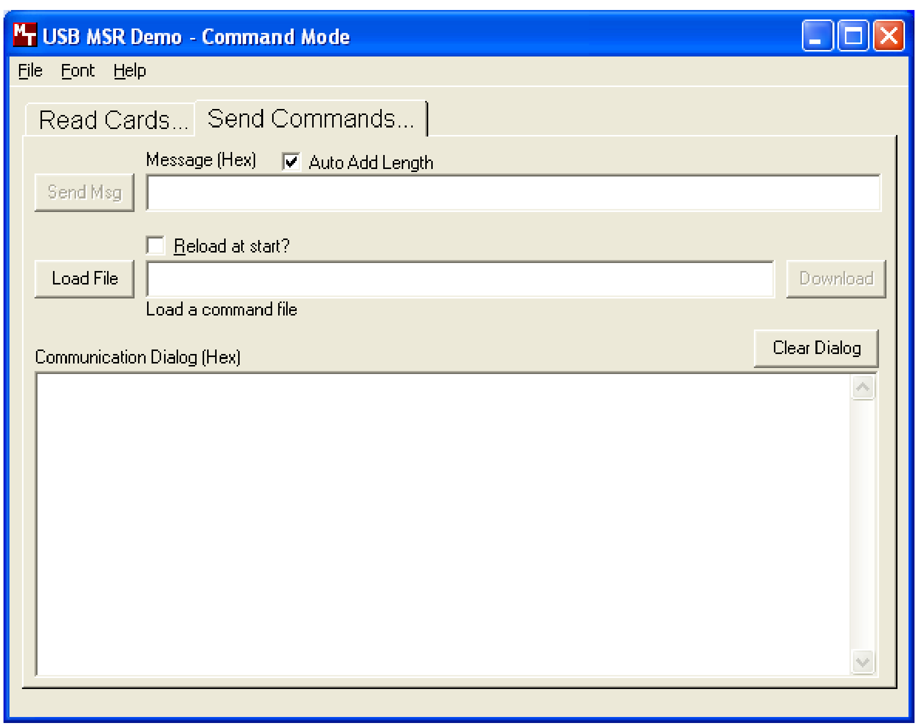
INSTALLATION

To install the demo program, run the setup.exe file and follow the instructions given on the screen.

OPERATION

To operate the demo program perform the following steps:

- Attach the device into a USB port on the host.
- If this is the first time the device has been plugged into the host, follow the instructions on the screen for installing the Windows HID device driver. This is explained in more detail in the installation section of this document.
- Run the demo program.



- To send commands to the device, click the ***Send Commands*** tab (if not already selected).

- Enter a command in the Message edit box. All data entered should be in hexadecimal bytes with a space between each byte. Enter the command number followed by the command data if there is any. **The application will automatically calculate and send the command data length for you** if the *Auto Add Length* box is checked**.** For example, to send the GET_PROPERTY command for property SOFTWARE_ID enter 00 00.
- Press Enter or click ***Send Msg*** to send the command and receive the result.
- The command request and the command result will be displayed in the Communications Dialog edit box.
- The ***Clear Dialog*** button clears the Communication Dialog edit box.
- To read cards and view the card data when in the HID mode, click the ***Read Cards*** tab.
- To read cards and view the card data when in the Keyboard Emulation mode, do not use the demo program. Use a text editor program such as Windows Notepad.

SOURCE CODE

Source code is included with the demo program. It can be used as a guide for application development. It is described in detail, with comments, to assist developers. The book *USB Complete* by Jan Axelson is also a good guide for application developers, especially the chapter on Human Interface Device Host Applications (see “Reference Documents” in Section 1).

APPENDIX A. KEYBOARD USAGE ID DEFINITIONS

This appendix is from the following document found on *www.usb.org*: Universal Serial Bus HID Usage Tables, Version 1.12 and specifically for this manual, Section 10, Keyboard/Keypad Page (0x07).

KEYBOARD/KEYPAD PAGE (0X07)

This section is the Usage Page for key codes to be used in implementing a USB keyboard. A Boot Keyboard (84-, 101- or 104-key) should at a minimum support all associated usage codes as indicated in the "Boot" column below.

The usage type of all key codes is Selectors (Sel), except for the modifier keys Keyboard Left Control (0x224) to Keyboard Right GUI (0x231) which are Dynamic Flags (DV).

Note. A general note on Usages and languages: Due to the variation of keyboards from language to language, it is not feasible to specify exact key mappings for every language. Where this list is not specific for a key function in a language, the closest equivalent key position should be used, so that a keyboard may be modified for a different language by simply printing different keycaps. One example is the Y key on a North American keyboard. In Germany this is typically Z. Rather than changing the keyboard firmware to put the Z Usage into that place in the descriptor list, the vendor should use the Y Usage on both the North American and German keyboards. This continues to be the existing practice in the industry, in order to minimize the number of changes to the electronics to accommodate other languages.

Table A-1. Keyboard/Keypad

Usage ID (Dec)	Usage ID (Hex)	Usage Name	Ref: Typical AT-101 Position	PC-AT	Mac	UNIX	Boot
0	00	Reserved (no event indicated) [9]	N/A	√	√	√	4/101/104
1	01	Keyboard ErrorRollOver[9]	N/A	√	√	√	4/101/104
2	02	Keyboard POSTFail[9]	N/A	√	√	√	4/101/104
3	03	Keyboard ErrorUndefined[9]	N/A	√	√	√	4/101/104
4	04	Keyboard a and A[4]	31	√	√	√	4/101/104
5	05	Keyboard b and B	50	√	√	√	4/101/104
6	06	Keyboard c and C[4]	48	√	√	√	4/101/104
7	07	Keyboard d and D	33	√	√	√	4/101/104
8	08	Keyboard e and E	19	√	√	√	4/101/104
9	09	Keyboard f and F	34	√	√	√	4/101/104
10	0A	Keyboard g and G	35	√	√	√	4/101/104
11	0B	Keyboard h and H	36	√	√	√	4/101/104
12	0C	Keyboard i and I	24	√	√	√	4/101/104
13	0D	Keyboard j and J	37	√	√	√	4/101/104
14	0E	Keyboard k and K	38	√	√	√	4/101/104
15	0F	Keyboard l and L	39	√	√	√	4/101/104
16	10	Keyboard m and M	52	√	√	√	4/101/104
17	11	Keyboard n and N	51	√	√	√	4/101/104
18	12	Keyboard o and O[4]	25	√	√	√	4/101/104

Usage ID (Dec)	Usage ID (Hex)	Usage Name	Ref: Typical AT-101 Position	PC-AT	Mac	UNIX	Boot
19	13	Keyboard p and P[4]	26	√	√	√	4/101/104
20	14	Keyboard q and Q[4]	27	√	√	√	4/101/104
21	15	Keyboard r and R	20	√	√	√	4/101/104
22	16	Keyboard s and S[4]	32	√	√	√	4/101/104
23	17	Keyboard t and T	21	√	√	√	4/101/104
24	18	Keyboard u and U	23	√	√	√	4/101/104
25	19	Keyboard v and V	49	√	√	√	4/101/104
26	1A	Keyboard w and W[4]	18	√	√	√	4/101/104
27	1B	Keyboard x and X[4]	47	√	√	√	4/101/104
28	1C	Keyboard y and Y[4]	22	√	√	√	4/101/104
29	1D	Keyboard z and Z[4]	46	√	√	√	4/101/104
30	1E	Keyboard 1 and ![4]	2	√	√	√	4/101/104
31	1F	Keyboard 2 and ![4]	3	√	√	√	4/101/104
32	20	Keyboard 3 and #[4]	4	√	√	√	4/101/104
33	21	Keyboard 4 and $[4]	5	√	√	√	4/101/104
34	22	Keyboard 5 and %[4]	6	√	√	√	4/101/104
35	23	Keyboard 6 and ^[4]	7	√	√	√	4/101/104
36	24	Keyboard 7 and &[4]	8	√	√	√	4/101/104
37	25	Keyboard 8 and *[4]	9	√	√	√	4/101/104
38	26	Keyboard 9 and ([4]	10	√	√	√	4/101/104
39	27	Keyboard 0 and)[4]	11	√	√	√	4/101/104
40	28	Keyboard Return (ENTER)[5]	43	√	√	√	4/101/104
41	29	Keyboard ESCAPE	110	√	√	√	4/101/104
42	2A	Keyboard DELETE (Backspace)	15	√	√	√	4/101/104
43	2B	Keyboard Tab	16	√	√	√	4/101/104
44	2C	Keyboard Spacebar	61	√	√	√	4/101/104
45	2D	Keyboard - and (underscore)[4]	12	√	√	√	4/101/104
46	2E	Keyboard = and +[4]	13	√	√	√	4/101/104
47	2F	Keyboard [and {[4]	27	√	√	√	4/101/104
48	30	Keyboard] and }[4]	28	√	√	√	4/101/104
49	31	Keyboard \ and |	29	√	√	√	4/101/104
50	32	Keyboard Non-US # and ~[2]	42	√	√	√	4/101/104
51	33	Keyboard ; and :[4]	40	√	√	√	4/101/104
52	34	Keyboard ‘ and “[4]	41	√	√	√	4/101/104
53	35	Keyboard Grave Accent and Tilde[4]	1	√	√	√	4/101/104
54	36	Keyboard, and <[4]	53	√	√	√	4/101/104
55	37	Keyboard. and >[4]	54	√	√	√	4/101/104
56	38	Keyboard / and ?	55	√	√	√	4/101/104
57	39	Keyboard Caps Lock[11]	30	√	√	√	4/101/104
58	3A	Keyboard F1	112	√	√	√	4/101/104

Usage ID (Dec)	Usage ID (Hex)	Usage Name	Ref: Typical AT-101 Position	PC-AT	Mac	UNIX	Boot
59	3B	Keyboard F2	113	√	√	√	4/101/104
60	3C	Keyboard F3	114	√	√	√	4/101/104
61	3D	Keyboard F4	115	√	√	√	4/101/104
62	3E	Keyboard F5	116	√	√	√	4/101/104
63	3F	Keyboard F6	117	√	√	√	4/101/104
64	40	Keyboard F7	118	√	√	√	4/101/104
65	41	Keyboard F8	119	√	√	√	4/101/104
66	42	Keyboard F9	120	√	√	√	4/101/104
67	43	Keyboard F10	121	√	√	√	4/101/104
68	44	Keyboard F11	122	√	√	√	101/104
69	45	Keyboard F12	123	√	√	√	101/104
70	46	Keyboard PrintScreen[1]	124	√	√	√	101/104
71	47	Keyboard Scroll Lock[11]	125	√	√	√	4/101/104
72	48	Keyboard Pause[1]	126	√	√	√	101/104
73	49	Keyboard Insert[1]	75	√	√	√	101/104
74	4A	Keyboard Home[1]	80	√	√	√	101/104
75	4B	Keyboard PageUp[1]	85	√	√	√	101/104
76	4C	Keyboard Delete Forward[1;14]	76	√	√	√	101/104
77	4D	Keyboard End[1]	81	√	√	√	101/104
78	4E	Keyboard PageDown[1]	86	√	√	√	101/104
79	4F	Keyboard RightArrow[1]	89	√	√	√	101/104
80	50	Keyboard LeftArrow[1]	79	√	√	√	101/104
81	51	Keyboard DownArrow[1]	84	√	√	√	101/104
82	52	Keyboard UpArrow[1]	83	√	√	√	101/104
83	53	Keypad Num Lock and Clear1[1]	90	√	√	√	101/104
84	54	Keypad /[1]	95	√	√	√	101/104
85	55	Keypad *	100	√	√	√	4/101/104
86	56	Keypad -	105	√	√	√	4/101/104
87	57	Keypad +	106	√	√	√	4/101/104
88	58	Keypad ENTER5	108	√	√	√	101/104
89	59	Keypad 1 and End	93	√	√	√	4/101/104
90	5A	Keypad 2 and Down Arrow	98	√	√	√	4/101/104
91	5B	Keypad 3 and PageDn	103	√	√	√	4/101/104
92	5C	Keypad 4 and Left Arrow	92	√	√	√	4/101/104
93	5D	Keypad 4 and Left Arrow	97	√	√	√	4/101/104
94	5E	Keypad 4 and Left Arrow	102	√	√	√	4/101/104
95	5F	Keypad 7 and Home	91	√	√	√	4/101/104
96	60	Keypad 8 and Up Arrow	96	√	√	√	4/101/104
97	61	Keypad 9 and PageUp	101	√	√	√	4/101/104
98	62	Keypad 0 and Insert	99	√	√	√	4/101/104
99	63	Keypad . and Delete	104	√	√	√	4/101/104

Usage ID (Dec)	Usage ID (Hex)	Usage Name	Ref: Typical AT-101 Position	PC-AT	Mac	UNIX	Boot
100	64	Keyboard Non-US \ and |[3;6]	45	√	√	√	4/101/104
101	65	Keyboard Application[10]	129	√		√	104
102	66	Keyboard Power[9] =			√	√	
103	67	Keypad =			√		
104	68	Keyboard F13	62		√		
105	69	Keyboard F14	63		√		
106	6A	Keyboard F15	64		√		
107	6B	Keyboard F16	65				
107	6C	Keyboard F17					
109	6D	Keyboard F18					
110	6E	Keyboard F19					
111	6F	Keyboard F20					
112	70	Keyboard F21					
113	71	Keyboard F22					
114	72	Keyboard F23					
115	73	Keyboard F24					
116	74	Keyboard Execute				√	
117	75	Keyboard Help				√	
118	76	Keyboard Menu				√	
119	77	Keyboard Select				√	
120	78	Keyboard Stop				√	
121	79	Keyboard Again				√	
122	7A	Keyboard Undo				√	
123	7B	Keyboard Cut				√	
124	7C	Keyboard Copy				√	
125	7D	Keyboard Paste				√	
126	7E	Keyboard Find				√	
127	7F	Keyboard Mute				√	
128	80	Keyboard Volume Up				√	
129	81	Keyboard Volume Down				√	
130	82	Keyboard Locking Caps Lock[12]				√	
131	83	Keyboard Locking Num Lock[12]				√	
132	84	Keyboard Locking Scroll Lock[12]				√	
133	85	Keypad Comma[27]	107				
134	86	Keypad Equal Sign[29]					
135	87	Keyboard International1[15-28]	56				
136	88	Keyboard International2[16]					
137	89	Keyboard International3[17]					
138	8A	Keyboard International4[18]					
139	8B	Keyboard International5[19]					
140	8C	Keyboard International6[20]					

Usage ID (Dec)	Usage ID (Hex)	Usage Name	Ref: Typical AT-101 Position	PC-AT	Mac	UNIX	Boot
141	8D	Keyboard International7[21]					
142	8E	Keyboard International8[22]					
143	8F	Keyboard International9[22]					
144	90	Keyboard Lang1[25]					
145	91	Keyboard Lang2[26]					
146	92	Keyboard Lang3[30]					
147	93	Keyboard Lang4[31]					
148	94	Keyboard Lang5[32]					
149	95	Keyboard Lang6[8]					
150	96	Keyboard Lang7[8]					
151	97	Keyboard Lang8[8]					
152	98	Keyboard Lang9[8]					
153	99	Keyboard Alternate Erase[7]					
154	9A	Keyboard Sys/Req Attention[1]					
155	9B	Keyboard Cancel					
156	9C	Keyboard Clear					
157	9D	Keyboard Prior					
158	9E	Keyboard Return					
159	9F	Keyboard Separator					
160	A0	Keyboard Out					
161	A1	Keyboard Oper					
162	A2	Keyboard Clear/Again					
163	A3	Keyboard Cr/Sel/Props					
164	A4	Keyboard Ex Sel					
165-175	A5-CF	Reserved					
176	B0	Keypad 00					
177	B1	Keypad 000					
178	B2	Thousands Separator[33]					
179	B3	Decimal Separator[33]					
180	B4	Currency Unit[34]					
181	B5	Currency Sub-unit[34]					
182	B6	Keypad (					
183	B7	Keypad)					
184	B8	Keypad {					
185	B9	Keypad}					
186	BA	Keypad Tab					
187	BB	Keypad Backspace					
188	BC	Keypad A					
189	BD	Keypad B					
190	BE	Keypad C					
191	BF	Keypad D					

Usage ID (Dec)	Usage ID (Hex)	Usage Name	Ref: Typical AT-101 Position	PC-AT	Mac	UNIX	Boot
192	C0	Keypad E					
193	C1	Keypad F					
194	C2	Keypad XOR					
195	C3	Keypad ^					
196	C4	Keypad %					
197	C5	Keypad <					
198	C6	Keypad >					
199	C7	Keypad &					
200	C8	Keypad &&					
201	C9	Keypad |					
202	CA	Keypad ||					
203	CB	Keypad :					
204	CC	Keypad #					
205	CD	Keypad Space					
206	CE	Keypad @					
207	CF	Keypad !					
208	D0	Keypad Memory Store					
209	D1	Keypad Memory Recall					
210	D2	Keypad Memory Clear					
211	D3	Keypad Memory Add					
212	D4	Keypad Memory Subtract					
213	D5	Keypad Memory Multiple					
214	D6	Keypad Memory Divide					
215	D7	Keypad +/-					
216	D8	Keypad Clear					
217	D9	Keypad Clear Entry					
218	DA	Keypad Binary					
219	DB	Keypad Octal					
220	DC	Keypad Decimal					
221	DD	Keypad Hexadecimal					
222-223	DE-DF	Reserved					
224	E0	Keyboard LeftControl	58	√	√	√	
225	E1	Keyboard LeftShift	44	√	√	√	
226	E2	Keyboard LeftA;t	60	√	√	√	
227	E3	Keyboard Left GUI[10;23]	127	√	√	√	
228	E4	Keyboard RightControl	64	√	√	√	
229	E5	Keyboard RightShift	57	√	√	√	
230	E6	Keyboard RightAlt	62	√	√	√	
231	E7	Keyboard Right GUI[10;24]	128	√	√	√	
232 – 65535	E8-FFFF	Reserved					

Footnotes

1. Usage of keys is not modified by the state of the Control, Alt, Shift or Num Lock keys. That is, a key does not send extra codes to compensate for the state of any Control, Alt, Shift or Num Lock keys.
2. Typical language mappings: US: \| Belg: µ`£ FrCa: <}> Dan:'* Dutch: <> Fren:*µ Ger: #' Ital: ù§ LatAm: }`] Nor:,* Span: }Ç Swed: ,* Swiss: $£ UK: #~.
3. Typical language mappings: Belg:<\> FrCa:«°» Dan:<\> Dutch:]¦[Fren:<> Ger:<|> Ital:<> LatAm:<> Nor:<> Span:<> Swed:<|> Swiss:<\> UK:\| Brazil: \|.
4. Typically remapped for other languages in the host system.
5. Keyboard Enter and Keypad Enter generate different Usage codes.
6. Typically near the Left-Shift key in AT-102 implementations.
7. Example, Erase-Eaze™ key.
8. Reserved for language-specific functions, such as Front End Processors and Input Method Editors.
9. Reserved for typical keyboard status or keyboard errors. Sent as a member of the keyboard array. Not a physical key.
10. Windows key for Windows 95, and "Compose."
11. Implemented as a non-locking key; sent as member of an array.
12. Implemented as a locking key; sent as a toggle button. Available for legacy support; however, most systems should use the non-locking version of this key.
13. Backs up the cursor one position, deleting a character as it goes.
14. Deletes one character without changing position.

15-20. See additional foot notes in Universal Serial Bus HID Usage Tables, Copyright © 1996-2005, USB Implementers Forum.

21. Toggle Double-Byte/Single-Byte mode.
22. Undefined, available for other Front End Language Processors.
23. Windowing environment key, examples are Microsoft Left Win key, Mac Left Apple key, Sun Left Meta key
24. Windowing environment key, examples are Microsoft® RIGHT WIN key, Macintosh® RIGHT APPLE key, Sun® RIGHT META key.
25. Hangul/English toggle key. This usage is used as an input method editor control key on a Korean language keyboard.
26. Hanja conversion key. This usage is used as an input method editor control key on a Korean language keyboard.
27. Keypad Comma is the appropriate usage for the Brazilian keypad period (.) key. This represents the closest possible match, and system software should do the correct mapping based on the current locale setting.
28. Keyboard International1 should be identified via footnote as the appropriate usage for the Brazilian forward-slash (/) and question-mark (?) key. This usage should also be renamed to either "Keyboard Non-US / and ?" or to "Keyboard International1" now that it's become clear that it does not only apply to Kanji keyboards anymore.
29. Used on AS/400 keyboards.
30. Defines the Katakana key for Japanese USB word-processing keyboards.
31. Defines the Hiragana key for Japanese USB word-processing keyboards.
32. Usage 0x94 (Keyboard LANG5) "Defines the Zenkaku/Hankaku key for Japanese USB word-processing keyboards.
33. The symbol displayed will depend on the current locale settings of the operating system. For example, the US thousands separator would be a comma, and the decimal separator would be a period.
34. The symbol displayed will depend on the current locale settings of the operating system. For example the US currency unit would be $ and the sub-unit would be ¢.

APPENDIX B. MODIFIER BYTE DEFINITIONS

This appendix is from the following document found on www.usb.org: Device Class Definition for Human Interface Devices (HID) Version 1.11, and specifically for this manual, Section 8.3 Report Format for Array Items.

The modifier byte is defined as follows:

Table B-1. Modifier Byte

Bit	Key
0	LEFT CTRL
1	LEFT SHIFT
2	LEFT ALT
3	LEFT GUI
4	RIGHT CTRL
5	RIGHT SHIFT
6	RIGHT ALT
7	RIGHT GUI

APPENDIX C. GUIDE ON DECRYPTING DATA

When a data field consists of more than one block, Cipher Block Chaining (CBC) method is used by the encrypting algorithm.

To decrypt this group of data, follow these steps:

- Start decryption on the last block.
- The result of the decryption is then XORed with the previous block.
- Continue until reaching the first block.
- The first block can skip the XOR operation.