

# **USB KB INTELLIHEAD FOR SWIPE READERS TECHNICAL REFERENCE MANUAL**

**Manual Part Number 99875321-10**

**JANUARY 2012**

**MAGTEK<sup>®</sup>**

**REGISTERED TO ISO 9001:2008**

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

Information in this document is subject to change without notice. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of MagTek, Inc.

MagTek is a registered trademark of MagTek, Inc.  
IntelliHead™ is a trademark of MagTek, Inc.

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

[Appendix A](#) is taken from Universal Serial Bus HID Usage Tables, Version 1.12, Section 10, Keyboard/Keypad Page (0x07) ©1996-2005 USB Implementers' Forum.

[Appendix B](#) is taken from Section 8.3 Report Format for Array Items, Device Class Definition for Human Interface Devices (HID) Version 1.11, ©1996-2001 USB Implementers' Forum, [hidcomments@usb.org](mailto:hidcomments@usb.org).

## REVISIONS

Rev Number	Date	Notes
1	28 Jan 05	Initial Release
2	10 Jun 05	Added examples to some of the commands; editorial throughout
3	8 Aug 05	Sec 1 and Appendices B, C, and D: Included additional models: 2103008, 21040132, and 21040133. Sec 4, To, ASCII to Keypress Conversion Type Property, added Active Keymap Property statement. To Get KeyMap Item Command and Set KeyMap Item Command, added the paragraph, "Starting with the firmware release..."
4	12 Sep 05	Added new Warranty Statement and Appendices G and H, USB and HID Usage Table and Device Class Definitions Table
5	24 Oct 05	Added new models: 21040137 and 21046004; Updated other drawing Revs
6	14 Sep 07	Corrected default setting for polling interval. Added 21030016.
7	8 Oct 07	Added ES TK1, ES TK2 and ES TK3 properties.
8	14 Oct 08	Added JIS type 2 decoding option; updated company address.
9	15 Jun 09	Updated Limited Warranty and Agency approvals. Added Pan Name Date Enable and Post Tk Char Enable Property
10	4 Jan 12	Added Host Poll Timeout Property (0x52).

## 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 Règlement sur le brouillage radioélectrique édicté par le ministère des Communications du Canada.

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

Cet appareil numérique de la classe B est conforme 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 ()

# TABLE OF CONTENTS

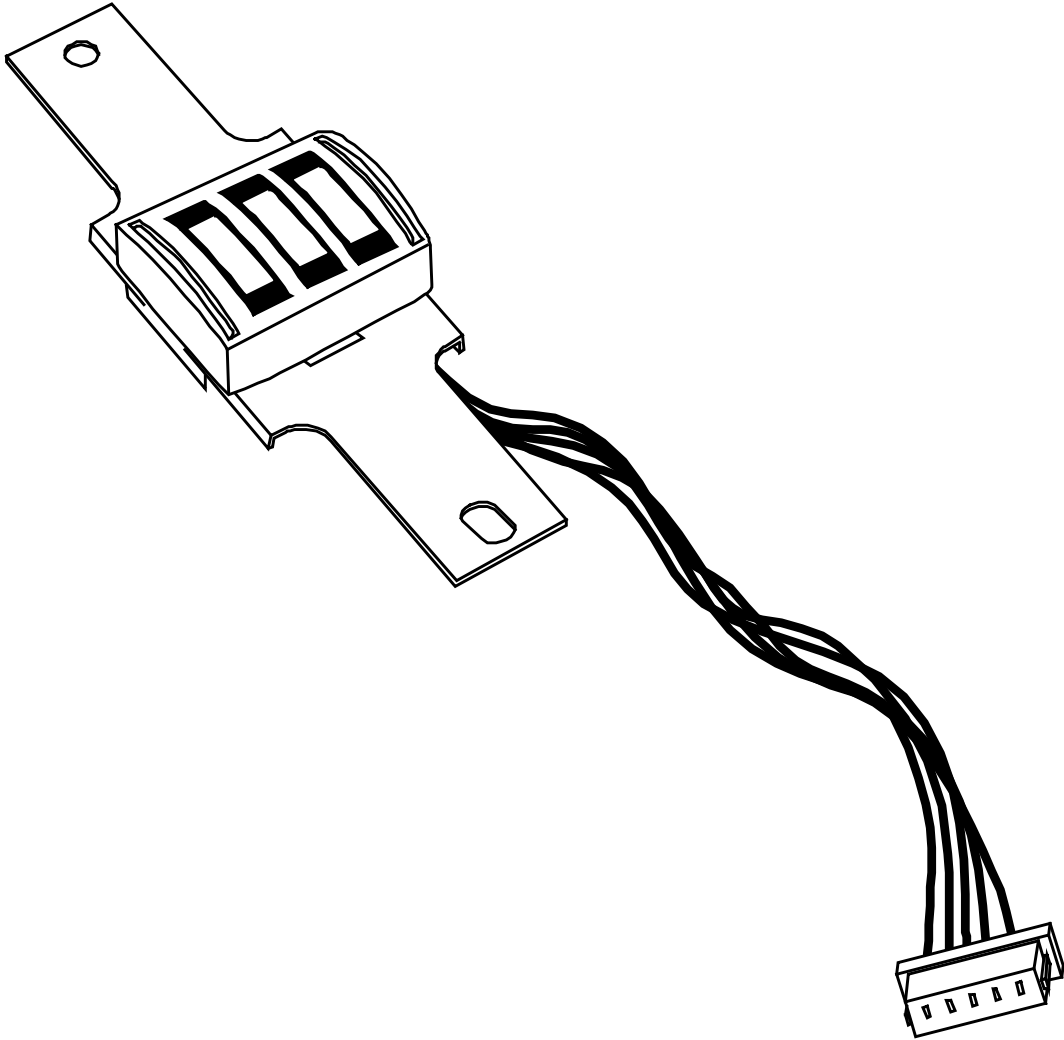
<b>SECTION 1. FEATURES AND SPECIFICATIONS.....</b>	<b>1</b>
FEATURES.....	1
HARDWARE CONFIGURATIONS .....	2
ACCESSORIES.....	2
REFERENCE DOCUMENTS.....	2
SPECIFICATIONS.....	3
<b>SECTION 2. INSTALLATION.....</b>	<b>5</b>
USB CONNECTION .....	5
WINDOWS PLUG AND PLAY SETUP .....	5
MOUNTING.....	5
<b>SECTION 3. OPERATION.....</b>	<b>7</b>
CARD READ.....	7
<b>SECTION 4. USB COMMUNICATIONS.....</b>	<b>9</b>
HOST APPLICATIONS.....	9
CARD DATA .....	9
PROGRAMMABLE CONFIGURATION OPTIONS.....	11
LOW LEVEL COMMUNICATIONS.....	11
HID USAGES.....	12
REPORT DESCRIPTOR .....	12
COMMANDS.....	13
COMMAND NUMBER .....	14
DATA LENGTH.....	14
DATA.....	14
RESULT CODE.....	14
GET AND SET PROPERTY COMMANDS.....	15
SOFTWARE ID PROPERTY .....	16
SERIAL NUM PROPERTY .....	17
POLLING INTERVAL PROPERTY .....	18
TRACK ID ENABLE PROPERTY .....	19
TRACK DATA SEND FLAGS PROPERTY .....	20
TERMINATION CHAR PROPERTY .....	21
SS TK2 7BITS PROPERTY.....	21
SS TK3 ISO ABA PROPERTY .....	22
SS TK3 AAMVA PROPERTY .....	22
SS TK3 7BITS PROPERTY.....	22
PRE CARD CHAR PROPERTY .....	23
POST CARD CHAR PROPERTY .....	23
PRE TK CHAR PROPERTY .....	23
POST TK CHAR PROPERTY.....	24
ASCII TO KEYPRESS CONVERSION TYPE PROPERTY .....	24
INTERFACE TYPE PROPERTY .....	25
ACTIVE KEYMAP PROPERTY .....	26
PRE CARD STRING PROPERTY.....	27
POST CARD STRING PROPERTY.....	28
SS TK1 ISO ABA PROPERTY .....	28
SS TK2 ISO ABA PROPERTY .....	29
ES PROPERTY.....	29
ES TK1 PROPERTY.....	29
ES TK2 PROPERTY.....	30
ES TK3 PROPERTY.....	30
DECODE ENABLE PROPERTY.....	31
SS JIS TYPE 2 PROPERTY.....	32
ES JIS TYPE 2 PROPERTY.....	32
PAN NAME DATE ENABLE PROPERTY .....	32
POST TK CHAR ENABLE PROPERTY .....	34
HOST POLL TIMEOUT PROPERTY.....	34
RESET DEVICE COMMAND.....	36
GET KEYMAP ITEM COMMAND .....	36
SET KEYMAP ITEM COMMAND .....	38
SAVE CUSTOM KEYMAP COMMAND.....	40

<b>SECTION 5. DEMO PROGRAM</b> .....	<b>41</b>
INSTALLATION.....	41
OPERATION.....	41
SOURCE CODE .....	42
<b>APPENDIX A. USAGE ID DEFINITIONS</b> .....	<b>43</b>
<b>APPENDIX B. MODIFIER BYTE DEFINITIONS</b> .....	<b>51</b>
<b>APPENDIX C. DRAWINGS</b> .....	<b>53</b>

## LIST OF FIGURES AND TABLES

Figure 1-1. 3-Track USB Keyboard Emulation IntelliHead.....	viii
Table 1-1. Specifications .....	3
Table 2-1. 5-Pin Connector .....	5
Table A-1. Keyboard/Keypad .....	43
Table B-1. Modifier Byte.....	51
Figure C-1. USB KB IntelliHead, 3-Track, 125mm Wire, 5-Pin Connector, Butterfly Spring .....	54
Figure C-2. USB KB IntelliHead, 3-Track, 275mm Wire, USB-A Connector .....	55
Figure C-3. USB KB IntelliHead, 3-Track, 125mm Wire, 5-Pin Connector, Accordion Spring .....	56
Figure C-4. USB KB IntelliHead, 3-Track, 6' Cable, USB-A Connector, 100mm Black Body .....	57
Figure C-5. USB KB IntelliHead, 3-Track, 6' Cable, USB-A Connector, 100mm Black Body .....	58
Figure C-6. USB KB IntelliHead, 3-Track, 5" Right Angle USB-A, 100mm Black Body.....	59
Figure C-7. USB KB IntelliHead, 3-Track, 125mm Wire, 5-pin Molex Connector, 90mm body.....	60
Figure C-8. USB KB IntelliHead, 3-Track, 125mm Wire, 5-pin Molex Connector, 60mm Slim Profile .....	61
Figure C-9. USB KB IntelliHead, 3-Track, 125mm Wire, 5-pin Molex Connector, 90mm Slim Profile .....	62





**Figure 1-1. 3-Track USB Keyboard Emulation IntelliHead**



# SECTION 1. FEATURES AND SPECIFICATIONS

The USB (Universal Serial Bus) Keyboard Emulation Swipe Reader is a compact magnetic stripe card reader that conforms to ISO standards. The Reader is compatible with any device with a USB host interface and emulates the operation of a keyboard. A card is read by sliding it past the head either forward or backward.

The Reader emulates a USB Human Interface Device (HID) United States keyboard or optionally all international keyboards 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***

*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.*

Because of potential “data interleave” issues associated with the USB Keyboard interface, MagTek recommends that the USB Keyboard Emulation MSR product should only be used if the application requires magnetic stripe data to be provided via the keyboard input. If previous applications were based upon RS-232 serial interface magnetic stripe readers, or if this is a brand new development effort, it is recommended that you use MagTek’s USB HID IntelliHead product. (Refer to Technical Manual 99875320 for further information regarding the USB HID IntelliHead.)

## **FEATURES**

Major features of the USB IntelliHead are as follows:

- Powered through the USB – no external power supply required
- Hardware Compatible with PC or any computer or terminal with a USB interface
- Bidirectional card reading
- Reads encoded data that meets ANSI/ISO/AAMVA/JIS Type 2 standards and others such as ISO track 1 format on track 2 or 3
- Reads up to three tracks of card data
- Compatible with USB specification Revision 1.1
- Compatible with HID specification Version 1.1
- Can use standard Windows HID drivers for communications. No third part device driver is required.
- Many programmable configuration options (including the ability to define the Start and End Sentinel characters and to insert a string of characters for the pre- and post-amble characters)
- Non-volatile memory for configuration storage
- Ability to convert to HID mode of operation

**HARDWARE CONFIGURATIONS**

The hardware configurations are as shown in the table below. Drawings of each model are included in Appendix C.

<b>Part Number</b>	<b>Description</b>	<b>Cable Length and Connector Type</b>	<b>Rail or Housing</b>
21030007	USB KB IntelliHead	125mm, 5 pin Molex	Butterfly Spring
21030008	USB KB IntelliHead	275mm,USB-A	Butterfly Spring
21030016	USB KB IntelliHead	125mm, 5 pin Molex	Accordion Spring
21040132	USB KB IntelliHead swipe reader	6 in, USB-A	100mm Black Body
21040133	USB KB IntelliHead swipe reader	6 ft, USB-A	100mm Black Body
21040137	USB KB IntelliHead swipe reader	6 in, USB-A Right Angle	100mm Black Body
21045088	USB KB IntelliHead open rail	125mm, 5 pin Molex	90mm Black rail
21046004	USB KB IntelliHead 60mm Slim rail	125mm, 5 pin Molex	60mm Black slim rail
21047012	USB KB IntelliHead 90mm slim rail	125mm, 5 pin Molex	90mm Black slim rail

**ACCESSORIES**

The accessories are as follows:

<b>Part Number</b>	<b>Description</b>
21042806	USB MSR Demo Program with Source Code (disk)
21051534	Test Cable to convert from IntelliHead to USB-A (6 ft)
99510026	USB MSR Demo Program with Source Code (WEB)

**REFERENCE DOCUMENTS**

MagTek *Magnetic Card Reader Design Kit Technical Specification* (99821002)

MagTek *USB HID IntelliHead for Swipe Readers, Technical Reference Manual* (99875320)

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>

*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<sup>©</sup> 1998 by Compaq Computer Corporation, Intel Corporation, Microsoft Corporation, NEC Corporation

USB Implementers Forum, Inc., [www.usb.org](http://www.usb.org)

**SPECIFICATIONS**

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

**Table 1-1. Specifications**

Reference Standards	ISO 7810, ISO 7811, AAMVA and JIS X 6302*
Power Input	5V from USB bus
Recording Method	Two-frequency coherent phase (F2F)
Message Format	ASCII
Card Speed	3 to 60 ips (7.62 – 152.4 cm/s)
<b>ELECTRICAL</b>	
Current	
Normal Mode	15mA
Suspend Mode	200 $\mu$ A
<b>MECHANICAL</b>	
Weight	1.1 oz. (31 gr.)
Cable length	See related drawing in Appendix C
Connector	See related drawing in Appendix C
<b>ENVIRONMENTAL</b>	
Temperature	
Operating	-40 °C to +70 °C (-40 °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, AAMVA (American Association of Motor Vehicle Administrators) and JIS (Japanese Industrial Standard).



## SECTION 2. INSTALLATION

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

### USB CONNECTION

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

Pin numbers and signal descriptions for the cable shown in the illustration are listed in Table 2-1. The connector is a Molex 52021-0500; one of the recommended mating connectors is Molex 53048-0510.

**Table 2-1. 5-Pin Connector**

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

### WINDOWS PLUG AND PLAY SETUP

On hosts with the Windows operating system, the first time the device is plugged into a specific USB port, Windows will pop up a dialog box, which will guide you through the process of installing a device driver for the device. After this process is completed once, Windows will no longer request this process as long as the device is plugged into the same USB port. The device driver that Windows will install for this device is the driver used for HID keyboard 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. 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

Refer to the appropriate Appendix in this document and/or to the *Magnetic Card Reader Design Kit* for complete mounting details.



## **SECTION 3. OPERATION**

### **CARD READ**

A card may be swiped past the read head at any time. The magnetic stripe must face toward the head and may be swiped in either direction. If there is data encoded on the card, the device will attempt to decode the data and then send the results to the host via a USB HID input report. After the results are sent to the host, all card data will be purged from the device, then 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 with the Human Interface Device (HID) class specification version 1.1. The device communicates to the host as a HID keyboard device. The latest versions of the Windows operating systems come with a standard Windows USB HID keyboard driver.

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

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

This device is powered from the USB bus. The vendor ID is 0x0801 and the product ID is 0x0001.

### HOST APPLICATIONS

This device can be used with existing applications that acquire card data via keyboard input. Also, applications that communicate to this device can be easily developed. These applications can be developed using compilers such as Microsoft's Visual Basic or Visual C++. To demonstrate this device's card reading capabilities any application that accepts keyboard input such as Window's Notepad can be used.

### CARD DATA

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.*

## USB Keyboard Emulation Swipe Reader

---

Because of potential “data interleave” issues associated with the USB Keyboard interface, MagTek recommends that the USB Keyboard Emulation MSR product should only be used by customers who have previously used MagTek’s Keyboard Wedge MSR, or who are interfacing with an existing PC software application which gathers card data from the keyboard port. If previous applications were based upon RS-232 serial interface MSRs, or if this is a brand new development effort, it is recommended that you use the MagTek’s USB HID IntelliHead MSR (Non-Keyboard Emulation Version). Refer to Technical Manual 99875320 for further information regarding the USB IntelliHead HID reader.

The device’s programmable configuration options affect the format of the card data.

The card data format for the default configuration is as follows:

[Tk1 SS] [Tk1 Data] [ES] [Tk2 SS] [Tk2 Data] [ES] [Tk3 SS] [Tk3 Data] [ES] [CR]

where:

- Tk1 SS = % (7-bit start sentinel)
- Tk2 SS = ; (ISO/ABA 5-bit start sentinel)  
@ (7-bit start sentinel)  
DEL (0x7F) (JIS type 2 start sentinel)\*
- Tk3 SS = + (ISO/ABA start sentinel)  
# (AAMVA start sentinel)  
& (7-bit start sentinel)
- ES = ? (end sentinel for all formats except JIS type 2)  
DEL (7F hex) (JIS type 2 end sentinel)\*
- CR = (carriage return) (0x0D)

\* Before the JIS type 2 encode type can be decoded, decoding must be enabled with the decode enable property. The JIS type 2 decoding option was not added until firmware with software ID 21042812K01 was release in October 2008. Some applications may not work well with the JIS type 2 format because this format allows ASCII characters in the non-printable range (0x00 – 0x1F and 0x7F).

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] [Tk1 SS] [Tk1 Data] [ES] [LRC] [P14] [P5] [P13] [Tk2 SS] [Tk2 Data] [ES] [LRC] [P14] [P5] [P13] [Tk3 SS] [Tk3 Data] [ES] [LRC] [P14] [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
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)

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

### **PROGRAMMABLE CONFIGURATION OPTIONS**

This device has a number of programmable configuration properties. These properties are stored in non-volatile memory. These properties can be configured at the factory or by the end user using a program supplied by MagTek. Programming these parameters requires low level communications with the device. 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. 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](http://www.usb.org).

**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](http://www.usb.org). Vendor defined usages must have a usage page in the range 0xff00 – 0xffff. All usages for this device use the standard HID keyboard usages or vendor defined magnetic stripe reader usage page 0xff00. The vendor defined usage IDs for this device are defined in the following table. The usage types are also listed. These usage types are defined in the HID Usage Tables document.

Magnetic Stripe Reader usage page 0xff00:

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

**REPORT DESCRIPTOR**

The HID 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

Item	Value(Hex)
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

## COMMANDS

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.

<b>Value (Hex)</b>	<b>Command Number</b>	<b>Description</b>
00	GET PROPERTY	Gets a property from the device
01	SET PROPERTY	Sets a property in the device
02	RESET DEVICE	Resets the device
03	GET KEYMAP ITEM	Gets a key map item
04	SET KEYMAP ITEM	Sets a key map item
05	SAVE CUSTOM KEYMAP	Saves the custom key map

**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.

<b>Value (Hex)</b>	<b>Result Code</b>	<b>Description</b>
00	SUCCESS	The command completed successfully.
01	FAILURE	The command failed.
02	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 0x00.

The Set Property command sets a property in the device. The Set Property command number is 0x01.

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 (Hex)	Property ID	Description
00	SOFTWARE ID	The device's software identifier
01	SERIAL NUM	The device's serial number
02	POLLING INTERVAL	The interrupt pipe's polling interval
03	TRACK ID ENABLE	Track enable / ID enable
04	TRACK DATA SEND FLAGS	Track data send flags
05	TERMINATION CHAR	Terminating char / per track or card flag
06	SS TK2 7BITS	Start sentinel char for track 2 – 7 bit data
08	SS TK3 ISO ABA	Start sentinel char for track 3 – ISO/ABA
09	SS TK3 AAMVA	Start sentinel char for track 3 - AAMVA
0A	SS TK3 7BITS	Start sentinel char for track 3 – 7 bit data
0B	PRE CARD CHAR	Pre card char
0C	POST CARD CHAR	Post card char
0D	PRE TK CHAR	Pre track char
0E	POST TK CHAR	Post track char
0F	ASCII TO KEYPRESS CONVERSION TYPE	Type of conversion performed when converting ASCII data to key strokes

Value (Hex)	Property ID	Description
10	INTERFACE TYPE	Type of USB interface
11	ACTIVE KEYMAP	Selects which key map to uses
12	PRE CARD STRING	Pre card string
13	POST CARD STRING	Post card string
14	SS TK1 ISO ABA	Start sentinel char for track 1 – ISO/ABA
16	SS TK2 ISO ABA	Start sentinel char for track 2 – ISO/ABA
16	ES	End sentinel char for all tracks/formats except JIS type 2
17	ES TK1	End sentinel char for track 1
18	ES TK2	End sentinel char for track 2 except JIS type 2
19	ES TK3	End sentinel char for track 3
1A	DECODE ENABLE	Enables decoding for certain formats
1B	SS JIS TYPE 2	Start sentinel char for JIS type 2
1C	ES JIS TYPE 2	End sentinel char for JIS type 2
1D	PAN NAME DATE ENABLE	Enables PAN name data format
1E	POST TK CHAR ENABLE	Enables the post track character per track
52	HOST POLL TIMEOUT	Allows USB poll timeout to be adjusted

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: 0x00  
 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



**SERIAL NUM PROPERTY**

Property ID:	0x01
Property Type:	String
Length:	0 – 15 bytes
Get Property:	Yes
Set Property:	Yes
Default Value:	The default value is no string with a length of zero.
Description:	The value is an ASCII string that represents the device's serial number. This string can be 0 – 15 bytes long. The value of this property, if any, will be sent to the host when the host requests the USB string descriptor.

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

Example Set **Serial Num** property Request (Hex):

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

Example Set **Serial Num** property Response (Hex):

Result Code	Data Len	Data
00	00	

Example Get **Serial Num** property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	01

Example Get **Serial Num** property Response (Hex):

Result Code	Data Len	Prp Value
00	03	31 32 33

**POLLING INTERVAL PROPERTY**

Property ID: 0x02  
 Property Type: Byte  
 Length: 1 byte  
 Get Property: Yes  
 Set Property: Yes  
 Default Value: 1  
 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.

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

**TRACK ID ENABLE PROPERTY**

Property ID: 0x03  
 Property Type: Byte  
 Length: 1 byte  
 Get Property: Yes  
 Set Property: Yes  
 Default Value: 0x95  
 Description: This property is defined as follows:

id	0	T <sub>3</sub>	T <sub>3</sub>	T <sub>2</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>1</sub>
----	---	----------------	----------------	----------------	----------------	----------------	----------------

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

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

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

Example Set **Track ID Enable** property Request (Hex):

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

Example Get **Track ID Enable** property Response (Hex):

Result Code	Data Len	Prp Value
00	01	95

### TRACK DATA SEND FLAGS PROPERTY

Property ID: 0x04  
Property Type: Byte  
Length: 1 byte  
Get Property: Yes  
Set Property: Yes  
Default Value: 0x63  
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

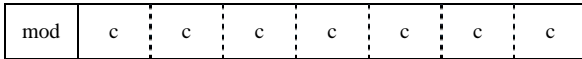
Note that the state of the Caps Lock key on the host keyboard has no affect on what case the card data is transmitted in unless the ICL bit in this property is set to 1.

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**

Property ID: 0x05  
 Property Type: Byte  
 Length: 1 byte  
 Get Property: Yes  
 Set Property: Yes  
 Default Value: 0x0D (carriage return)  
 Description: This property is defined as follows:



- 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**

Property ID: 0x06  
 Property Type: Byte  
 Length: 1 byte  
 Get Property: Yes  
 Set Property: Yes  
 Default Value: 0x40 '@'  
 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

Property ID: 0x08  
Property Type: Byte  
Length: 1 byte  
Get Property: Yes  
Set Property: Yes  
Default Value: 0x2B '+'  
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

Property ID: 0x09  
Property Type: Byte  
Length: 1 byte  
Get Property: Yes  
Set Property: Yes  
Default Value: 0x23 '#'  
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

Property ID: 0x0A  
Property Type: Byte  
Length: 1 byte  
Get Property: Yes  
Set Property: Yes  
Default Value: 0x26 '&'  
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

Property ID: 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

Property ID: 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

Property ID: 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

Property ID: 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

Property ID: 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, an 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: 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

Property ID: 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

Property ID: 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

Property ID: 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

Property ID: 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

Property ID: 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

Property ID: 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 except JIS type 2. 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 TK1 PROPERTY

Property ID: 0x17  
Property Type: Byte  
Length: 1 byte  
Get Property: Yes  
Set Property: Yes  
Default Value: 0xFF (use ES property)  
Description: This character is sent as the end sentinel for track 1 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. If the value is 0xFF then the value of the ES property will be used instead of this property. This property was not present until firmware revision with software ID 21042812H01.  
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 TK2 PROPERTY

Property ID: 0x18  
Property Type: Byte  
Length: 1 byte  
Get Property: Yes  
Set Property: Yes  
Default Value: 0xFF (use ES property)  
Description: This character is sent as the end sentinel for track 2 with any format except JIS type 2. 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. If the value is 0xFF then the value of the ES property will be used instead of this property. This property was not present until firmware revision with software ID 21042812H01.

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 TK3 PROPERTY

Property ID: 0x19  
Property Type: Byte  
Length: 1 byte  
Get Property: Yes  
Set Property: Yes  
Default Value: 0xFF (use ES property)  
Description: This character is sent as the end sentinel for track 3 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. If the value is 0xFF then the value of the ES property will be used instead of this property. This property was not present until firmware revision with software ID 21042812H01.

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.

**DECODE ENABLE PROPERTY**

Property ID: 0x1A  
 Property Type: Byte  
 Length: 1 byte  
 Get Property: Yes  
 Set Property: Yes  
 Default Value: 0x00  
 Description: This property is defined as follows:

Bit Position	7	6	5	4	3	2	1	0
Decode Type	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	JIS Type 2

When a decode type bit is set to 1 (true), the decode type represented by that bit is enabled. When a decode type bit is set to 0 (false), the decode type represented by that bit is disabled. The reserved decode type bits should always be set to zero.

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.

**Note**

*The JIS type 2 decoding option was not added until firmware with software ID 21042812K01 was release in October 2008. Some applications may not work well with the JIS type 2 format because this format allows ASCII characters in the non-printable range (0x00 – 0x1F and 0x7F).*

Example Set property Request (Hex):

Cmd Num	Data Len	Prp ID	Prp Value
01	02	1A	01 (enable JIS Type 2 decode type)

Example Set property Response (Hex):

Result Code	Data Len	Data
00	00	

Example Get property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	1A

Example Get property Response (Hex):

Result Code	Data Len	Prp Value
00	01	01

**SS JIS TYPE 2 PROPERTY**

Property ID: 0x1B  
Property Type: Byte  
Length: 1 byte  
Get Property: Yes  
Set Property: Yes  
Default Value: 0x7F 'DEL'  
Description: This character is sent as the start sentinel for cards that are encoded in the JIS type 2 format. If the value is in the range 0 – 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 JIS TYPE 2 PROPERTY**

Property ID: 0x1C  
Property Type: Byte  
Length: 1 byte  
Get Property: Yes  
Set Property: Yes  
Default Value: 0x7F 'DEL'  
Description: This character is sent as the end sentinel for cards that are encoded in the JIS type 2 format. If the value is in the range 0 – 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.

**PAN NAME DATE ENABLE PROPERTY**

Property ID: 0x1D  
Property Type: Byte  
Length: 1 byte  
Get Property: Yes  
Set Property: Yes  
Default Value: 0x00 (DISABLED)  
Description: When this property is set to 0, the reader data transmission will use the standard mode of sending track information as described in SECTION 4. When this value is set to non-zero, and Track 1 or Track 2 is determined to be in a financial data format, the reader will transmit the PAN (Primary Account Number), Name and Expiration date using the following format:

If Track 1 is available:  
<PAN><TAB><Name><TAB><MM><TAB><YY><Termination  
Character>  
Track 2 data will not be transmitted



If only Track 2 is available:

<PAN><TAB><TAB><MM><TAB><YY><Termination Character>

If Track 1 and Track 2 are not in financial data format or cannot be decoded, the reader will revert back to standard 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.

Rules used for determining that a card is in financial data format:

- PAN length should be between 13 and 19
- Name length should be between 0 and 26 characters for Track 1
- Only 2 field separators ('^') for Track 1
- Only 1 field separator ('=') for Track 2
- Format code shall be the character 'B' for Track 1

Example Set **PAN Name Date Enable** property to 01 Request (Hex):

Cmd Num	Data Len	Prp ID	Prp Value
01	02	1D	01

Example Set **PAN Name Date Enable** property Response (Hex):

Result Code	Data Len	Data
00	00	

Example Get **PAN Name Date Enable** property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	1D

Example Get **PAN Name Date Enable** property Response (Hex):

Result Code	Data Len	Prp Value
00	01	01

\*This property was not added until firmware version 21042812L02.

### POST TK CHAR ENABLE PROPERTY

Property ID: 0x1E  
Property Type: Byte  
Length: 1 byte  
Get Property: Yes  
Set Property: Yes  
Default Value: 0x07  
Description: This property is used to enable or disable the post track character for each track individually. The post track character is set separately with the post track character property. To enable the post track character for a given track set its corresponding bit position to one. To disable it, set it to zero. The following table shows how this properties bit positions relate to each track.

Bit position	7	6	5	4	3	2	1	0
Track	Reserved	Reserved	Reserved	Reserved	Reserved	3	2	1

The reserved track fields should always be set to zero. For example, to enable all three tracks set this property to 0x07. To enable only track 1 set this property to 0x01.

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 property was not added until firmware version 21042812L02.

### HOST POLL TIMEOUT PROPERTY

Property ID: 0x52  
Property Type: Byte  
Length: 1 byte  
Get Property: Yes  
Set Property: Yes  
Default Value: 0x02 (2 seconds)  
Description: This property can be used to adjust the device's host poll timeout. The property can be set to 0 to disable the timeout or it can be set to a value in the range of 1 to 60 seconds.

The host poll timeout was added around the year 2010 because if a USB suspend occurred while the reader was in the middle of transmitting card data to the host, the reader would no longer be able to read cards until power cycled. It was given a fixed value of 2 seconds. If a USB suspend now occurred while the reader was transmitting card data, this timeout would occur and the remainder of the card data would be discarded and the reader would be ready to read the next card once it got a USB resume signal. Getting a USB suspend while transmitting card data is not an event that would be expected to occur under normal operating conditions, however a customer was

seeing this occur due to abnormal USB bus activity from other devices on the bus.

Around the year 2012, starting with firmware part number 21042886 Revision C.01, this timeout was made adjustable with this property so that it could be disabled or adjusted. This property was added because some printers, made by HP and used as a host in this application, were occasionally ceasing to poll the reader for more than two seconds which would cause a timeout to occur which would in turn cause the host application to have problems. The timeout was disabled to resolve this problem. The printer was not behaving properly in this case. This problem would not be expected to occur on an error free USB Bus.

Not all readers contain this timeout and not all readers contain this property to adjust it.

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 HOST POLL TIMEOUT property Request (Hex):

Cmd Num	Data Len	Prp ID	Prp Value
01	02	52	02

Example Set HOST POLL TIMEOUT property Response (Hex):

Result Code	Data Len	Data
00	00	

Example Get HOST POLL TIMEOUT property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	52

Example Get HOST POLL TIMEOUT property Response (Hex):

Result Code	Data Len	Prp Value
00	01	02

### RESET DEVICE COMMAND

Command number: 0x02

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

Command number: 0x03

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

Command number: 0x04

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 >).
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

Command number: 0x05

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	



## SECTION 5. DEMO PROGRAM

The purpose of this demo program is not to demonstrate card reading with this Keyboard Emulation device. Use a text editor application such as Windows Notepad to demonstrate card reading for this keyboard emulation device. Any application that allows user input from a keyboard should be sufficient to demonstrate card reading for this device.

The primary purpose of the demo program, when used with this keyboard emulation device, is to allow users to change the device's programmable configuration properties. This is accomplished by sending commands to the device with the demo program. The demo program also comes with source code that can be used as a guide for application developers who want to change the device's programmable configuration properties in an application. However, it is unlikely that application developers will want to change these properties in an application since these properties only need to be set once and can be set at the factory. This program is written in Visual Basic.

Demo programs, version 1.2.0 and newer work on Windows 98, Me, 2000 and XP. Older versions do not support the HID keyboard emulation device on Windows 2000 or XP. These older versions only work on Windows 98 and Me.

When the demo program is run, a button for reading cards is displayed along with a button for sending commands. The card reading option is not supported for this keyboard emulation device. Use a text editor application such as Windows Notepad to demonstrate card reading for this keyboard emulation device.

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 to a USB port on the host
- If this is the first time the device has been plugged into the host, then 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 read cards and view the card data do not use the demo program. Use a text editor program such as Windows Notepad.
- To send commands to the device, click on the send commands button.
- 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.** For example, to send the GET PROPERTY command for property SOFTWARE ID enter 00 00.

- Press Enter or click on Send message 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.

### **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. USAGE ID DEFINITIONS

This appendix is from the following document found on [www.usb.org](http://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) <sup>9</sup>	N/A	√	√	√	4/101/104
1	01	Keyboard ErrorRollOver <sup>9</sup>	N/A	√	√	√	4/101/104
2	02	Keyboard POSTFail <sup>9</sup>	N/A	√	√	√	4/101/104
3	03	Keyboard ErrorUndefined <sup>9</sup>	N/A	√	√	√	4/101/104
4	04	Keyboard a and A <sup>4</sup>	31	√	√	√	4/101/104
5	05	Keyboard b and B	50	√	√	√	4/101/104
6	06	Keyboard c and C <sup>4</sup>	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

## USB Keyboard Emulation Swipe Reader

Usage ID (Dec)	Usage ID (Hex)	Usage Name	Ref: Typical AT-101 Position	PC-AT	Mac	UNIX	Boot
17	11	Keyboard n and N	51	√	√	√	4/101/104
18	12	Keyboard o and O <sup>4</sup>	25	√	√	√	4/101/104
19	13	Keyboard p and P <sup>4</sup>	26	√	√	√	4/101/104
20	14	Keyboard q and Q <sup>4</sup>	27	√	√	√	4/101/104
21	15	Keyboard r and R	20	√	√	√	4/101/104
22	16	Keyboard s and S <sup>4</sup>	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 <sup>4</sup>	18	√	√	√	4/101/104
27	1B	Keyboard x and X <sup>4</sup>	47	√	√	√	4/101/104
28	1C	Keyboard y and Y <sup>4</sup>	22	√	√	√	4/101/104
29	1D	Keyboard z and Z <sup>4</sup>	46	√	√	√	4/101/104
30	1E	Keyboard 1 and ! <sup>4</sup>	2	√	√	√	4/101/104
31	1F	Keyboard 2 and ! <sup>4</sup>	3	√	√	√	4/101/104
32	20	Keyboard 3 and # <sup>4</sup>	4	√	√	√	4/101/104
33	21	Keyboard 4 and \$ <sup>4</sup>	5	√	√	√	4/101/104
34	22	Keyboard 5 and % <sup>4</sup>	6	√	√	√	4/101/104
35	23	Keyboard 6 and ^ <sup>4</sup>	7	√	√	√	4/101/104
36	24	Keyboard 7 and & <sup>4</sup>	8	√	√	√	4/101/104
37	25	Keyboard 8 and * <sup>4</sup>	9	√	√	√	4/101/104
38	26	Keyboard 9 and ( <sup>4</sup>	10	√	√	√	4/101/104
39	27	Keyboard 0 and ) <sup>4</sup>	11	√	√	√	4/101/104
40	28	Keyboard Return (ENTER) <sup>5</sup>	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) <sup>4</sup>	12	√	√	√	4/101/104
46	2E	Keyboard = and + <sup>4</sup>	13	√	√	√	4/101/104
47	2F	Keyboard [ and { <sup>4</sup>	27	√	√	√	4/101/104
48	30	Keyboard ] and } <sup>4</sup>	28	√	√	√	4/101/104
49	31	Keyboard \ and	29	√	√	√	4/101/104
50	32	Keyboard Non-US # and ~ <sup>2</sup>	42	√	√	√	4/101/104
51	33	Keyboard ; and : <sup>4</sup>	40	√	√	√	4/101/104
52	34	Keyboard ' and " <sup>4</sup>	41	√	√	√	4/101/104
53	35	Keyboard Grave Accent and Tilde <sup>4</sup>	1	√	√	√	4/101/104
54	36	Keyboard, and < <sup>4</sup>	53	√	√	√	4/101/104
55	37	Keyboard. and > <sup>4</sup>	54	√	√	√	4/101/104
56	38	Keyboard / and ?	55	√	√	√	4/101/104

**Appendix A. Usage ID Definitions**

Usage ID (Dec)	Usage ID (Hex)	Usage Name	Ref: Typical AT-101 Position	PC-AT	Mac	UNIX	Boot
57	39	Keyboard Caps Lock <sup>11</sup>	30	√	√	√	4/101/104
58	3A	Keyboard F1	112	√	√	√	4/101/104
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 <sup>1</sup>	124	√	√	√	101/104
71	47	Keyboard Scroll Lock <sup>11</sup>	125	√	√	√	4/101/104
72	48	Keyboard Pause <sup>1</sup>	126	√	√	√	101/104
73	49	Keyboard Insert <sup>1</sup>	75	√	√	√	101/104
74	4A	Keyboard Home <sup>1</sup>	80	√	√	√	101/104
75	4B	Keyboard PageUp <sup>1</sup>	85	√	√	√	101/104
76	4C	Keyboard Delete Forward <sup>1;14</sup>	76	√	√	√	101/104
77	4D	Keyboard End <sup>1</sup>	81	√	√	√	101/104
78	4E	Keyboard PageDown <sup>1</sup>	86	√	√	√	101/104
79	4F	Keyboard RightArrow <sup>1</sup>	89	√	√	√	101/104
80	50	Keyboard LeftArrow <sup>1</sup>	79	√	√	√	101/104
81	51	Keyboard DownArrow <sup>1</sup>	84	√	√	√	101/104
82	52	Keyboard UpArrow <sup>1</sup>	83	√	√	√	101/104
83	53	Keypad Num Lock and Clear <sup>1</sup>	90	√	√	√	101/104
84	54	Keypad / <sup>1</sup>	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

## USB Keyboard Emulation Swipe Reader

Usage ID (Dec)	Usage ID (Hex)	Usage Name	Ref: Typical AT-101 Position	PC-AT	Mac	UNIX	Boot
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
100	64	Keyboard Non-US \ and   <sup>3,6</sup>	45	√	√	√	4/101/104
101	65	Keyboard Application <sup>10</sup>	129	√		√	104
102	66	Keyboard Power <sup>9</sup> =			√	√	
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 <sup>12</sup>				√	
131	83	Keyboard Locking Num Lock <sup>12</sup>				√	
132	84	Keyboard Locking Scroll Lock <sup>12</sup>				√	
133	85	Keypad Comma <sup>27</sup>	107				
134	86	Keypad Equal Sign <sup>29</sup>					
135	87	Keyboard International1 <sup>15-28</sup>	56				
136	88	Keyboard International2 <sup>16</sup>					

**Appendix A. Usage ID Definitions**

<b>Usage ID (Dec)</b>	<b>Usage ID (Hex)</b>	<b>Usage Name</b>	<b>Ref: Typical AT-101 Position</b>	<b>PC-AT</b>	<b>Mac</b>	<b>UNIX</b>	<b>Boot</b>
137	89	Keyboard International3 <sup>17</sup>					
138	8A	Keyboard International4 <sup>18</sup>					
139	8B	Keyboard International5 <sup>19</sup>					
140	8C	Keyboard International6 <sup>20</sup>					
141	8D	Keyboard International7 <sup>21</sup>					
142	8E	Keyboard International8 <sup>22</sup>					
143	8F	Keyboard International9 <sup>22</sup>					
144	90	Keyboard Lang1 <sup>25</sup>					
145	91	Keyboard Lang2 <sup>26</sup>					
146	92	Keyboard Lang3 <sup>30</sup>					
147	93	Keyboard Lang4 <sup>31</sup>					
148	94	Keyboard Lang5 <sup>32</sup>					
149	95	Keyboard Lang6 <sup>8</sup>					
150	96	Keyboard Lang7 <sup>8</sup>					
151	97	Keyboard Lang8 <sup>8</sup>					
152	98	Keyboard Lang9 <sup>8</sup>					
153	99	Keyboard Alternate Erase <sup>7</sup>					
154	9A	Keyboard Sys/Req Attention <sup>1</sup>					
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 <sup>33</sup>					
179	B3	Decimal Separator <sup>33</sup>					
180	B4	Currency Unit <sup>34</sup>					
181	B5	Currency Sub-unit <sup>34</sup>					
182	B6	Keypad (					
183	B7	Keypad )					
184	B8	Keypad {					
185	B9	Keypad}					
186	BA	Keypad Tab					

## USB Keyboard Emulation Swipe Reader

Usage ID (Dec)	Usage ID (Hex)	Usage Name	Ref: Typical AT-101 Position	PC-AT	Mac	UNIX	Boot
187	BB	Keypad Backspace					
188	BC	Keypad A					
189	BD	Keypad B					
190	BE	Keypad C					
191	BF	Keypad D					
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 <sup>10;23</sup>	127	√	√	√	



Usage ID (Dec)	Usage ID (Hex)	Usage Name	Ref: Typical AT-101 Position	PC-AT	Mac	UNIX	Boot
228	E4	Keyboard RightControl	64	√	√	√	
229	E5	Keyboard RightShift	57	√	√	√	
230	E6	Keyboard RightAlt	62	√	√	√	
231	E7	Keyboard Right GUI <sup>10;24</sup>	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](http://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**

<b>Bit</b>	<b>Key</b>
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. DRAWINGS

The following drawings are provided in this section:

### Part Number Title

21030007	USB KB IntelliHead, 3-Track, 125mm Wire, 5-Pin Molex Connector, Butterfly Spring
21030008	USB KB IntelliHead, 3-Track, 275mm Wire, USB-A Connector
21030016	USB KB IntelliHead, 3-Track, 125mm Wire, 5-Pin Connector, Accordion Spring
21040132	USB KB IntelliHead, 3-Track, 6" wire, USB-A Connector, 100mm Black Body
21040133	USB KB IntelliHead, 3-Track, 6' Cable, USB-A Connector, 100mm Black Body
21040137	USB KB IntelliHead, 3-Track, 5" Cable, Right Angle USB-A, 100mm Black body
21045088	USB KB IntelliHead, 3-Track, 125mm Wire, 5-pin Molex Connector, 90mm body
21046004	USB KB IntelliHead, 3-Track, 125mm Wire, 5-pin Molex Connector, 60mm Slim Profile
21047012	USB KB IntelliHead, 3-Track, 125mm Wire , 5-pin Molex Connector, 90mm Slim Profile

# USB Keyboard Emulation Swipe Reader

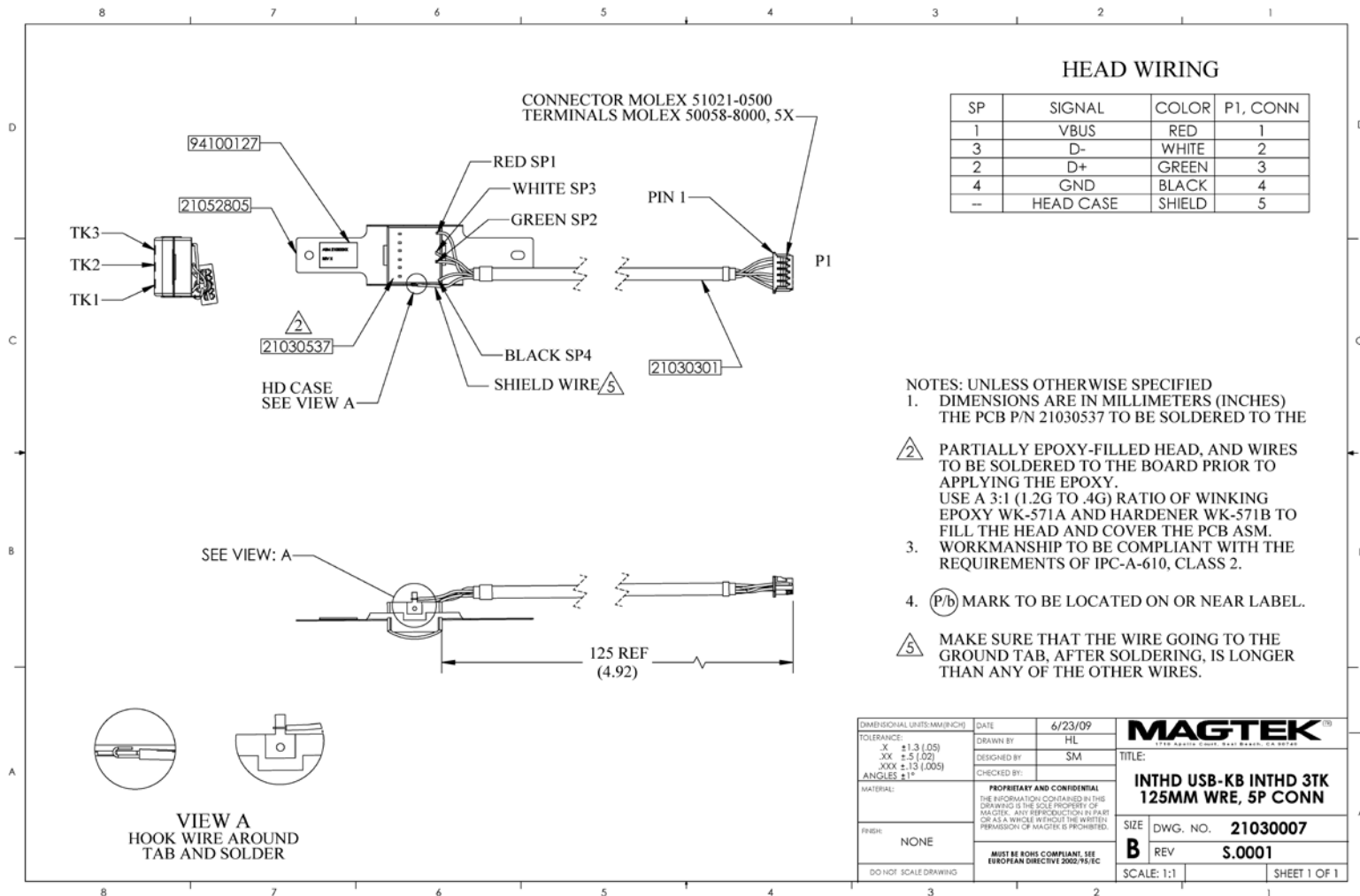


Figure C-1. USB KB IntelliHead, 3-Track, 125mm Wire, 5-Pin Connector, Butterfly Spring

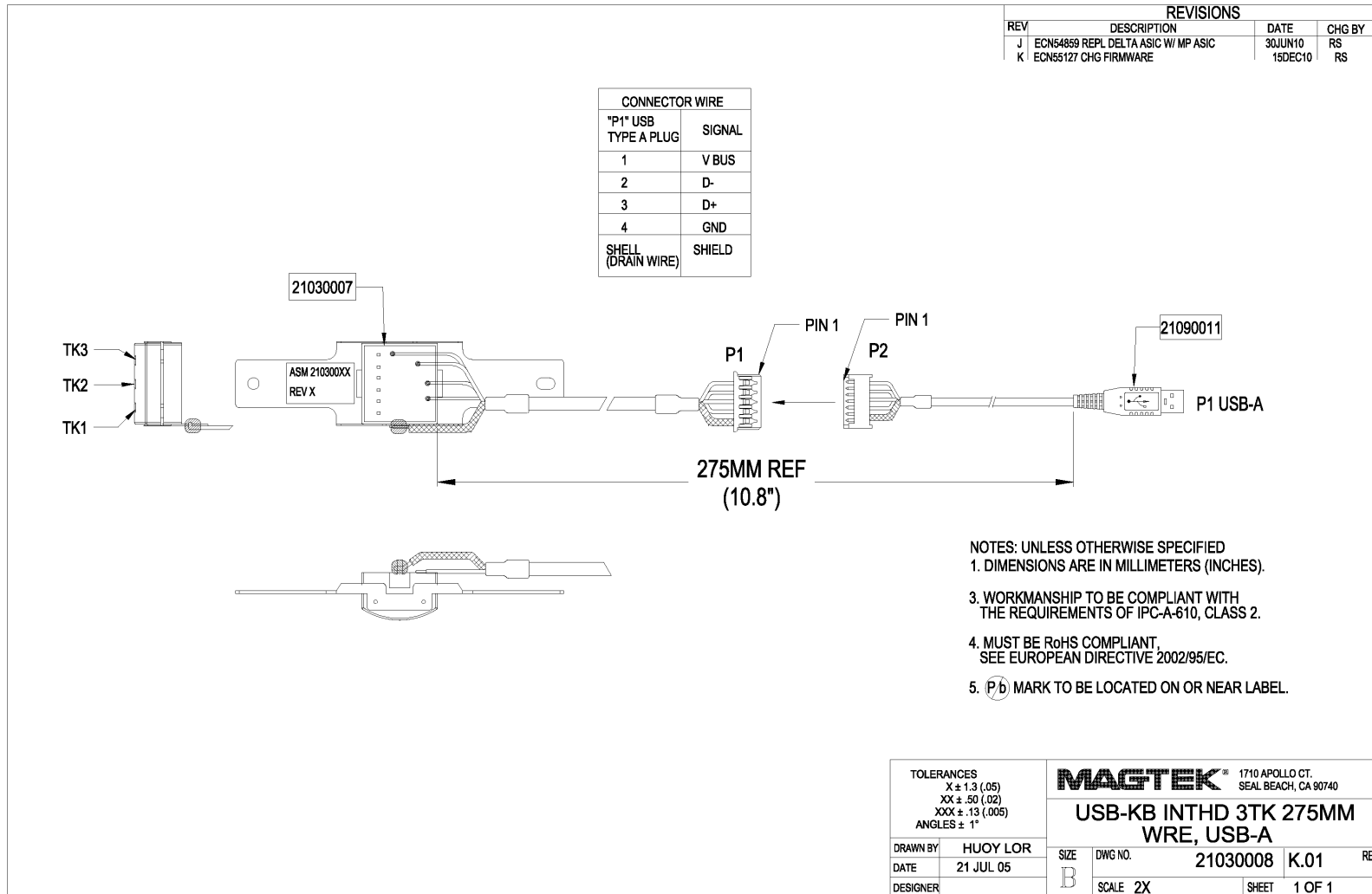


Figure C-2. USB KB IntelliHead, 3-Track, 275mm Wire, USB-A Connector

# USB Keyboard Emulation Swipe Reader

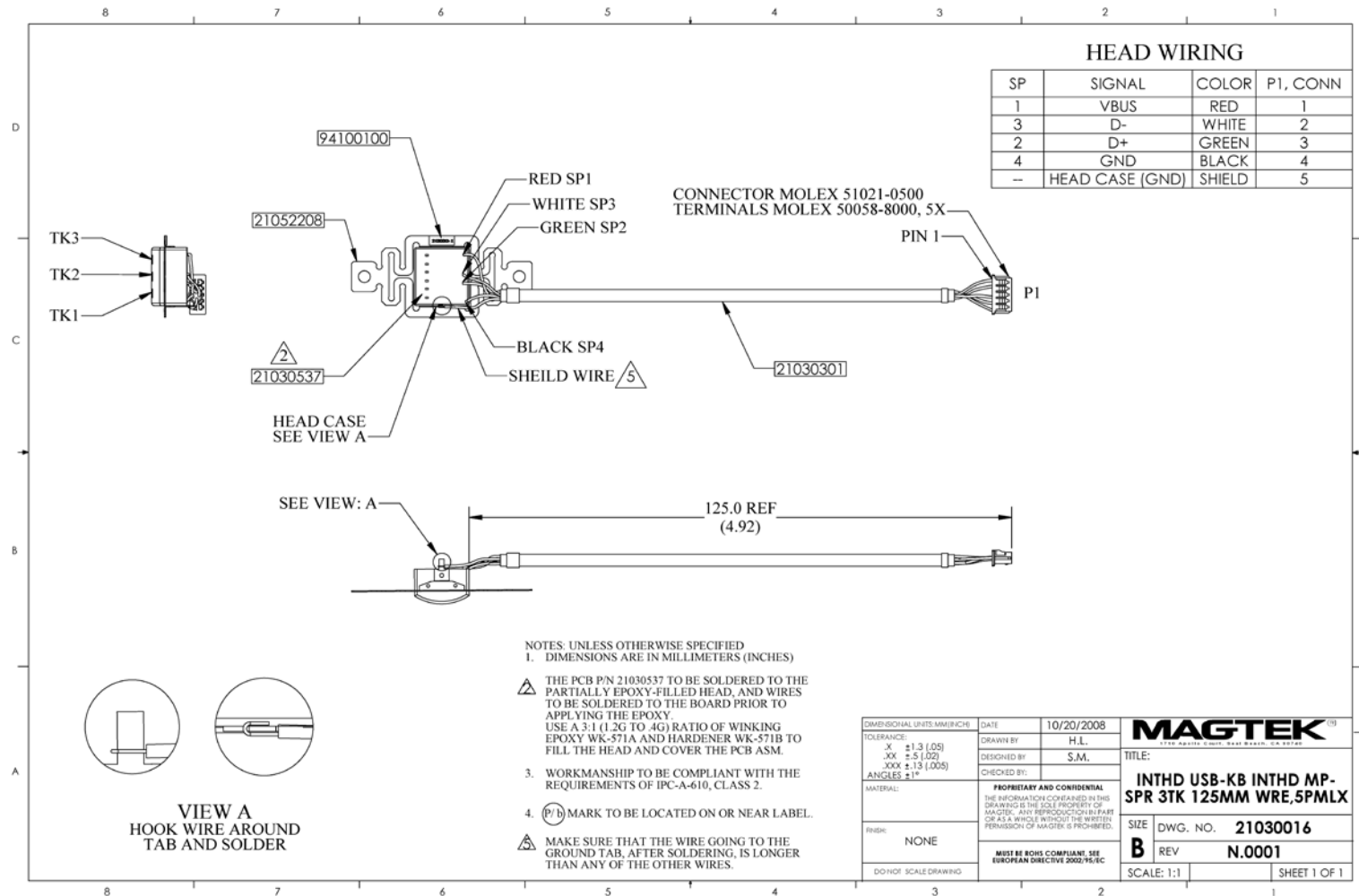


Figure C-3. USB KB IntelliHead, 3-Track, 125mm Wire, 5-Pin Connector, Accordion Spring



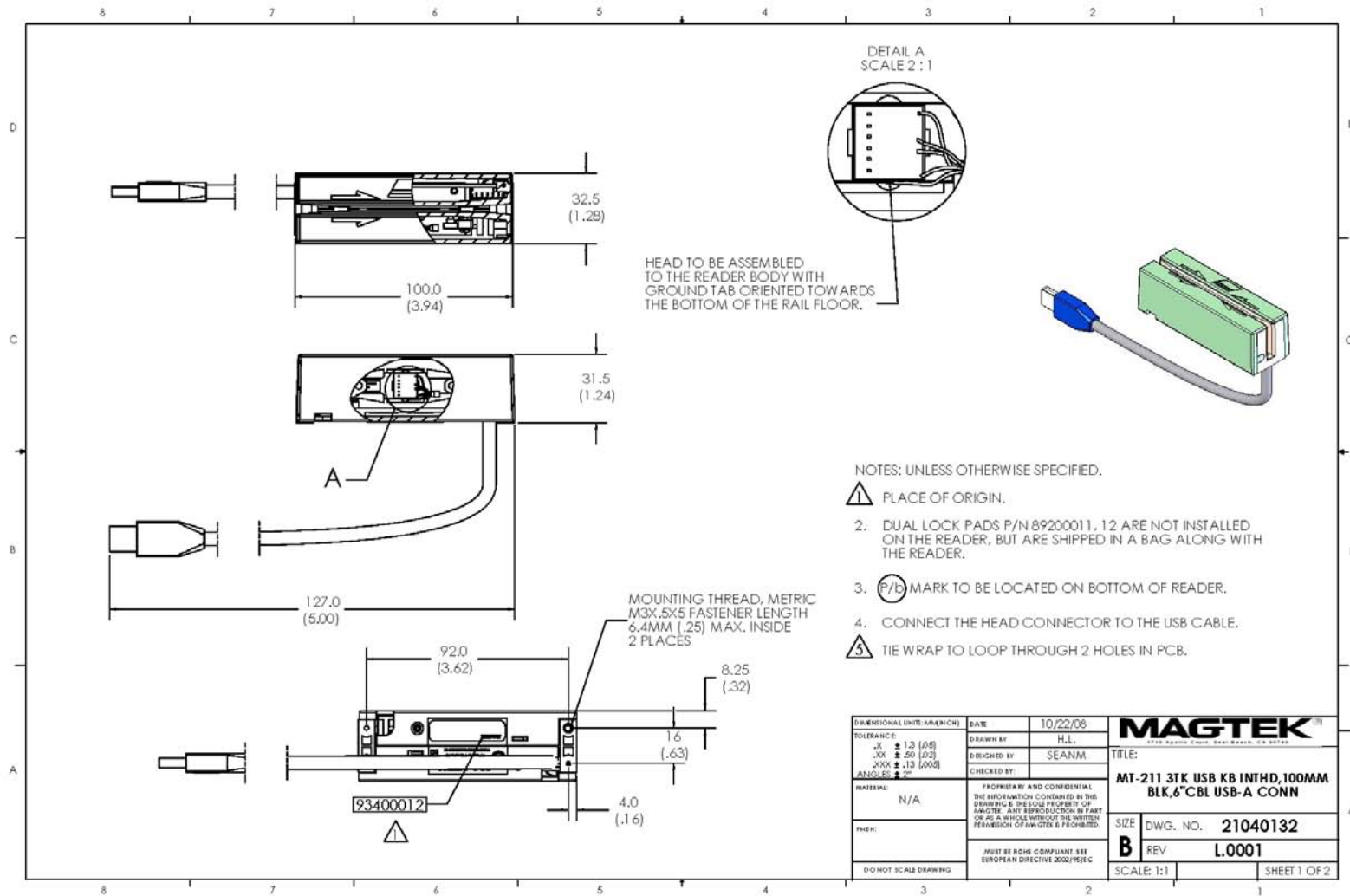


Figure C-4. USB KB IntelliHead, 3-Track, 6" Cable, USB-A Connector, 100mm Black Body

# USB Keyboard Emulation Swipe Reader

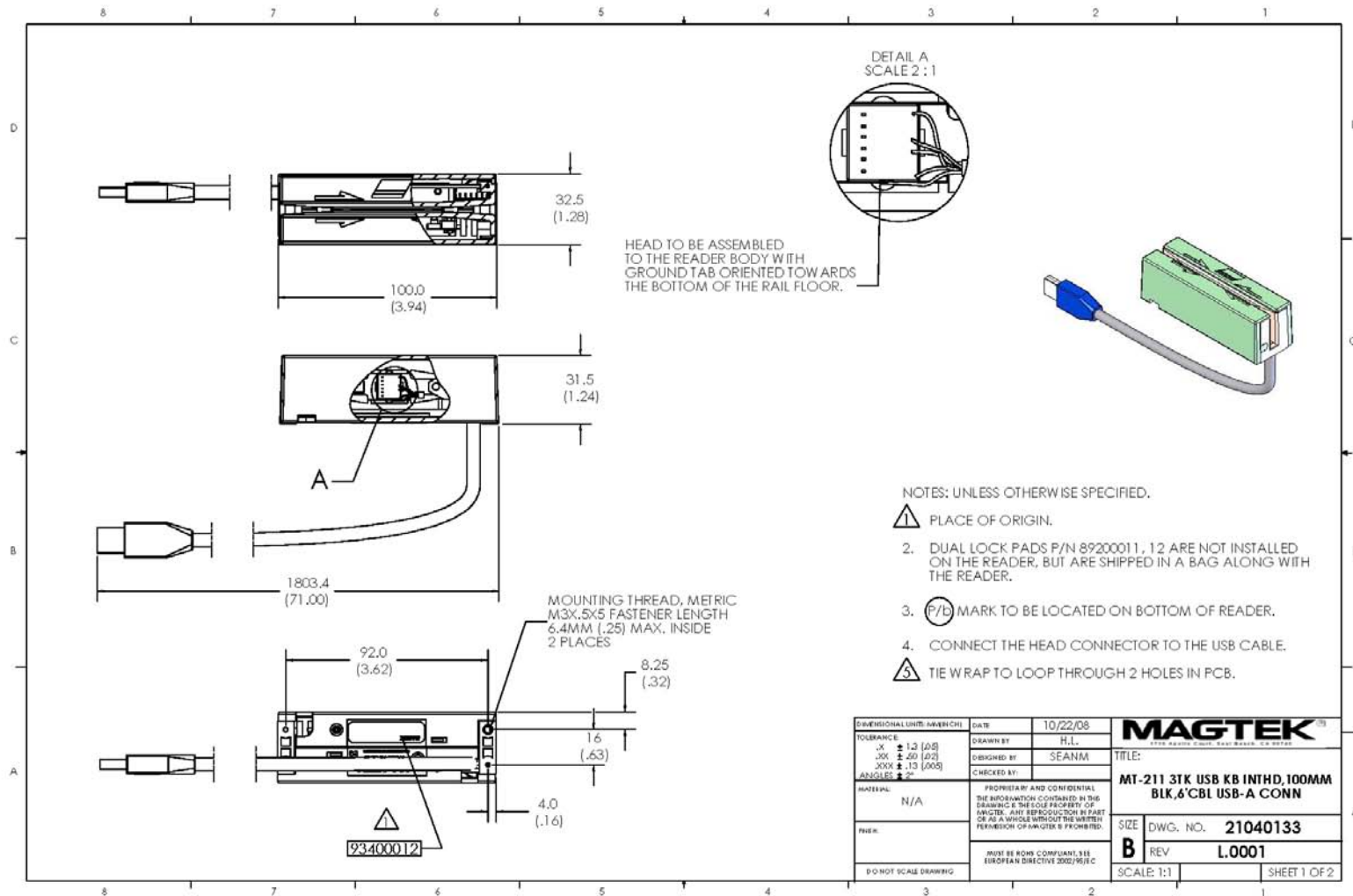


Figure C-5. USB KB IntelliHead, 3-Track, 6' Cable, USB-A Connector, 100mm Black Body

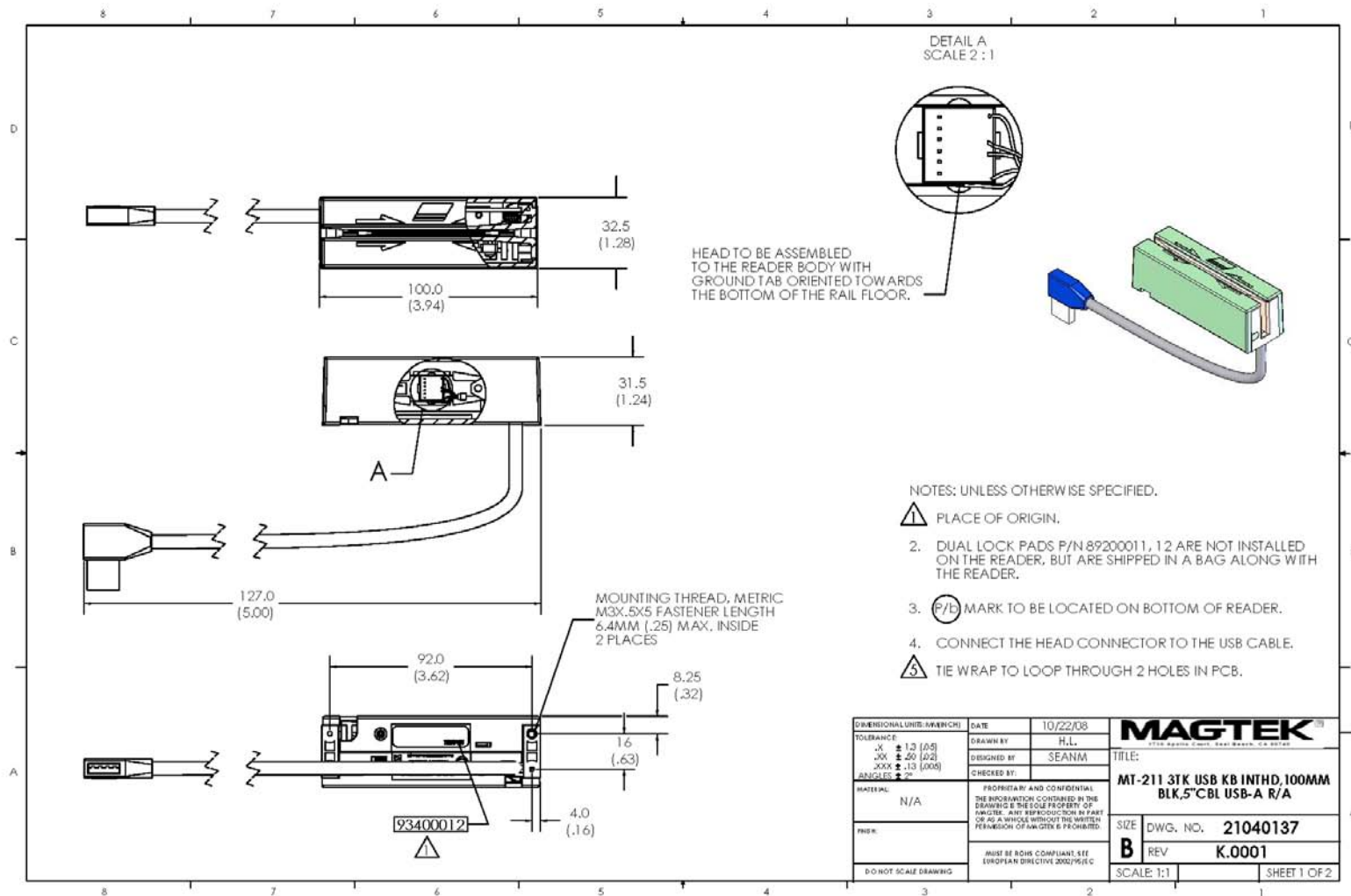


Figure C-6. USB KB IntelliHead, 3-Track, 5" Right Angle USB-A, 100mm Black Body

# USB Keyboard Emulation Swipe Reader

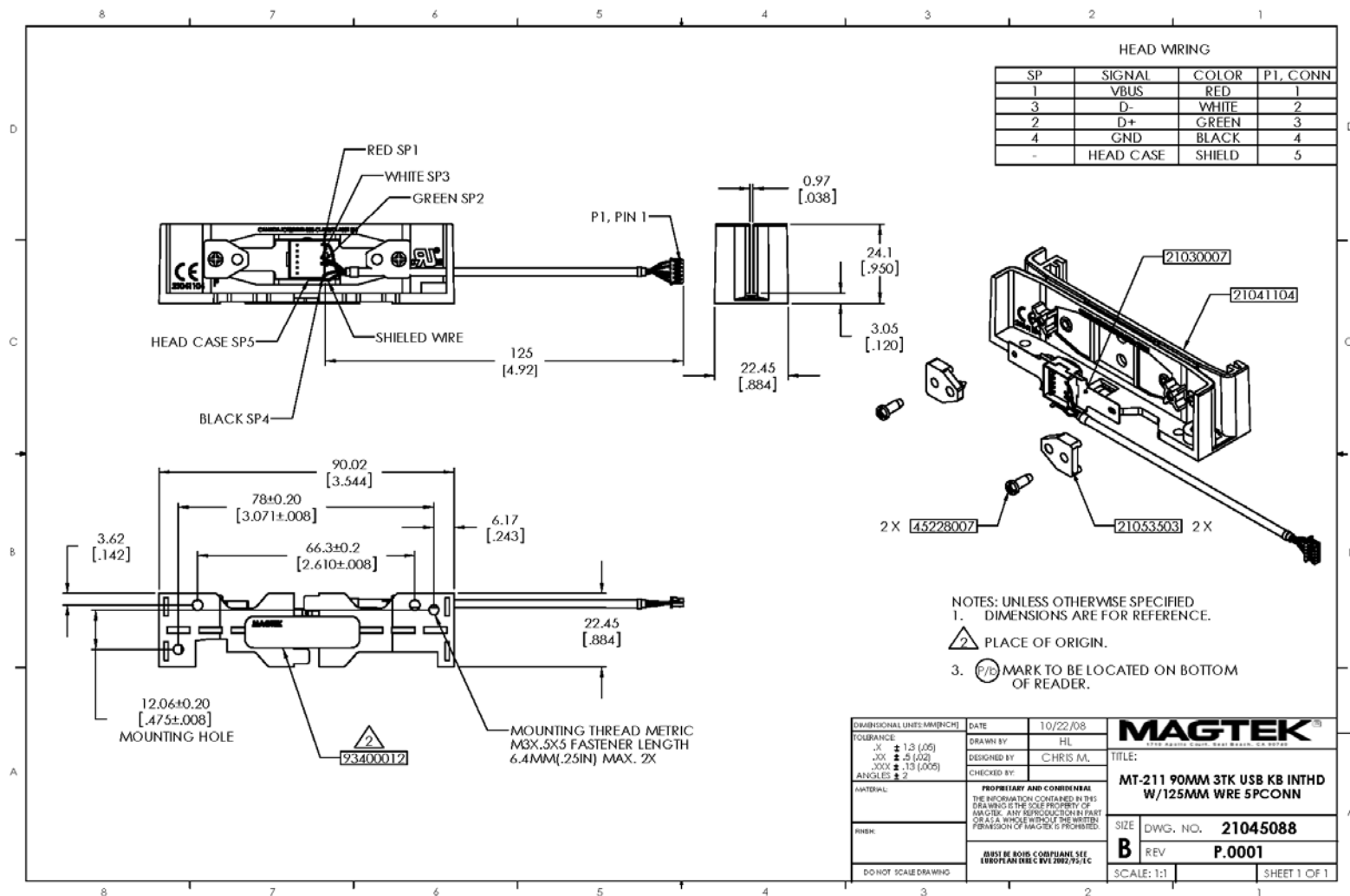


Figure C-7. USB KB IntelliHead, 3-Track, 125mm Wire, 5-pin Molex Connector, 90mm body

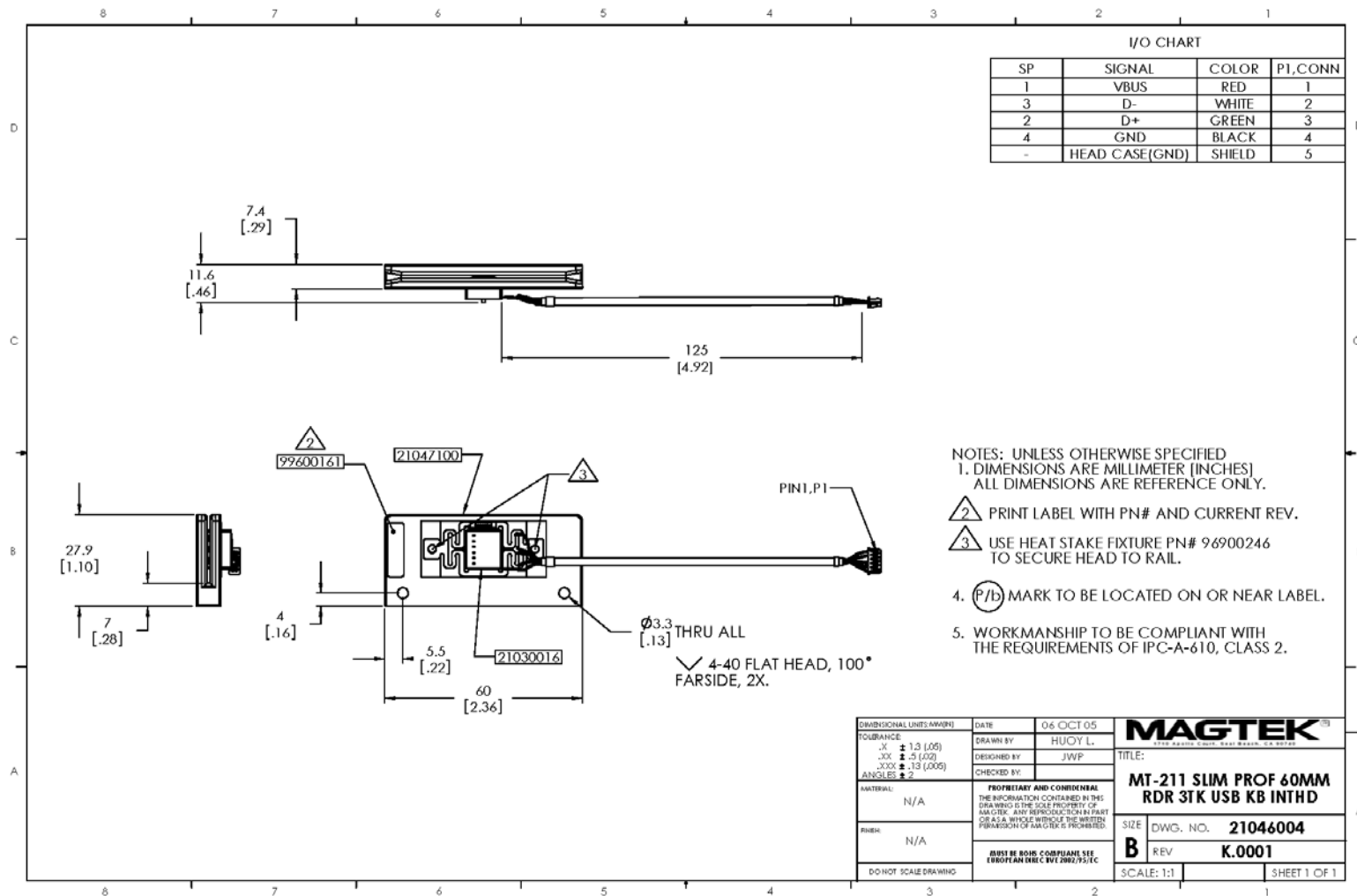


Figure C-8. USB KB IntelliHead, 3-Track, 125mm Wire, 5-pin Molex Connector, 60mm Slim Profile

# USB Keyboard Emulation Swipe Reader

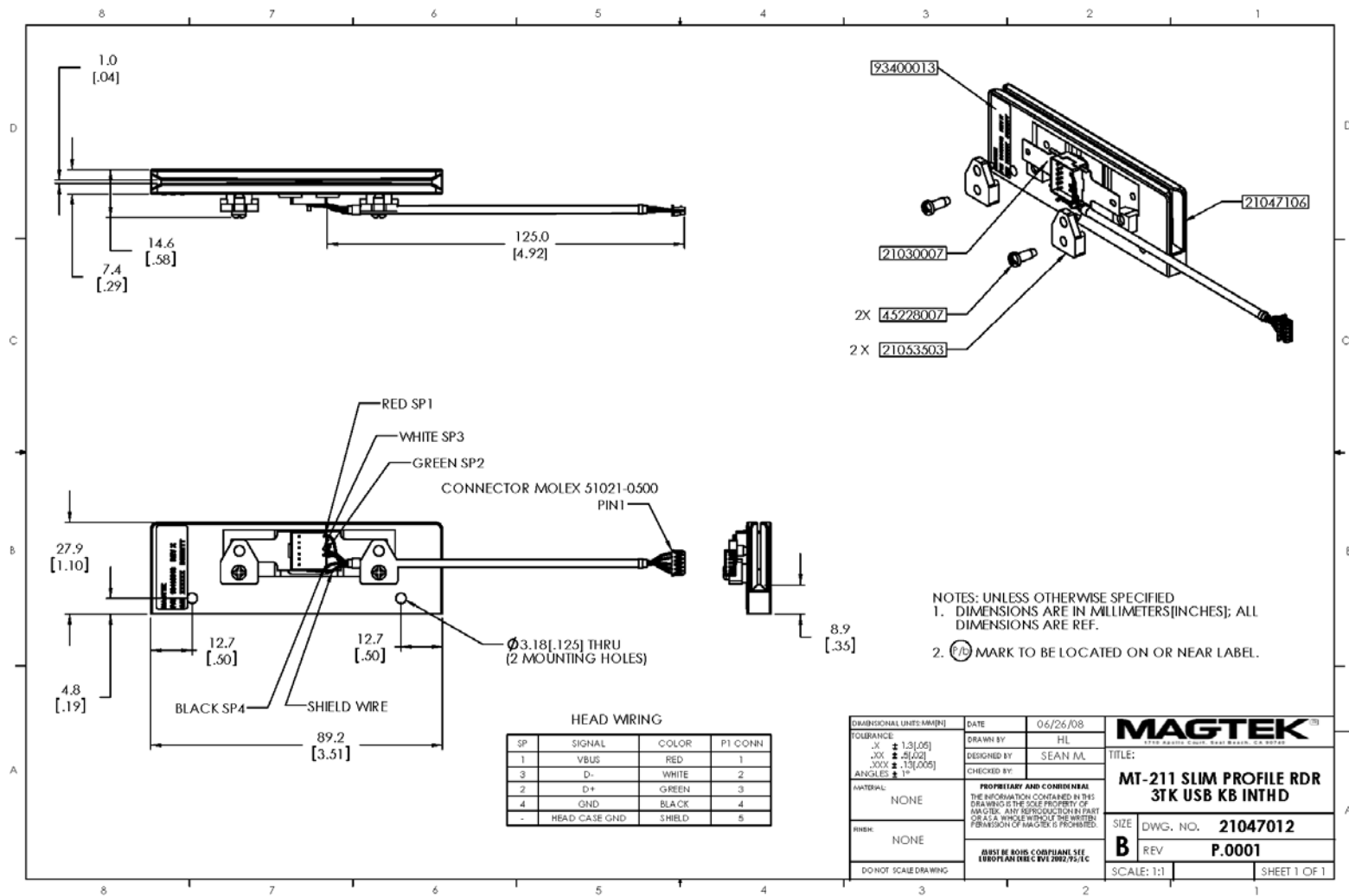


Figure C-9. USB KB IntelliHead, 3-Track, 125mm Wire, 5-pin Molex Connector, 90mm Slim Profile