MINI MICR USB KEYBOARD EMULATION

WITH OPTIONAL 3-TRACK MSR

TECHNICAL REFERENCE MANUAL

Manual Part Number: 99875371-1

JANUARY 2008

REGISTERED TO ISO 9001:2000

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

REVISIONS

Rev	Date	Notes
1.01	Jan 13, 08	Initial Release

LIMITED WARRANTY

MagTek warrants that the products sold to Reseller 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 original purchaser unless the buyer is authorized by MagTek to resell the products, in which event, this warranty shall apply only to the first repurchase.

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, customer's negligence, Reseller's negligence, or non-MagTek modification of the product. 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 by customers.

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.

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.

EACH PURCHASER UNDERSTANDS THAT THE MAGTEK PRODUCT IS OFFERED AS IS. 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, TO RESELLER OR TO RESELLER'S CUSTOMERS, SHALL IN NO EVENT EXCEED THE TOTAL AMOUNT PAID TO MAGTEK BY RESELLER UNDER THIS AGREEMENT. IN NO EVENT WILL MAGTEK BE LIABLE TO THE RESELLER OR THE RESELLER'S CUSTOMER 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 THE PRODUCTS.

MAGTEK MAKES NO OTHER WARRANTIES WITH RESPECT TO THE PRODUCTS, 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 PRODUCTS, 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 found to comply with the limits for 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 to 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 for 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 las classe B prescrites dans le Réglement sur le brouillage radioélectrique édicté par les ministère des Communications du Canada.

CE STANDARDS

Testing for compliance to CE and FCC requirements was performed by an independent laboratory. The unit under test was found compliant to Class B.

UL/CSA

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

TABLE OF CONTENTS

FIGURES

TABLES

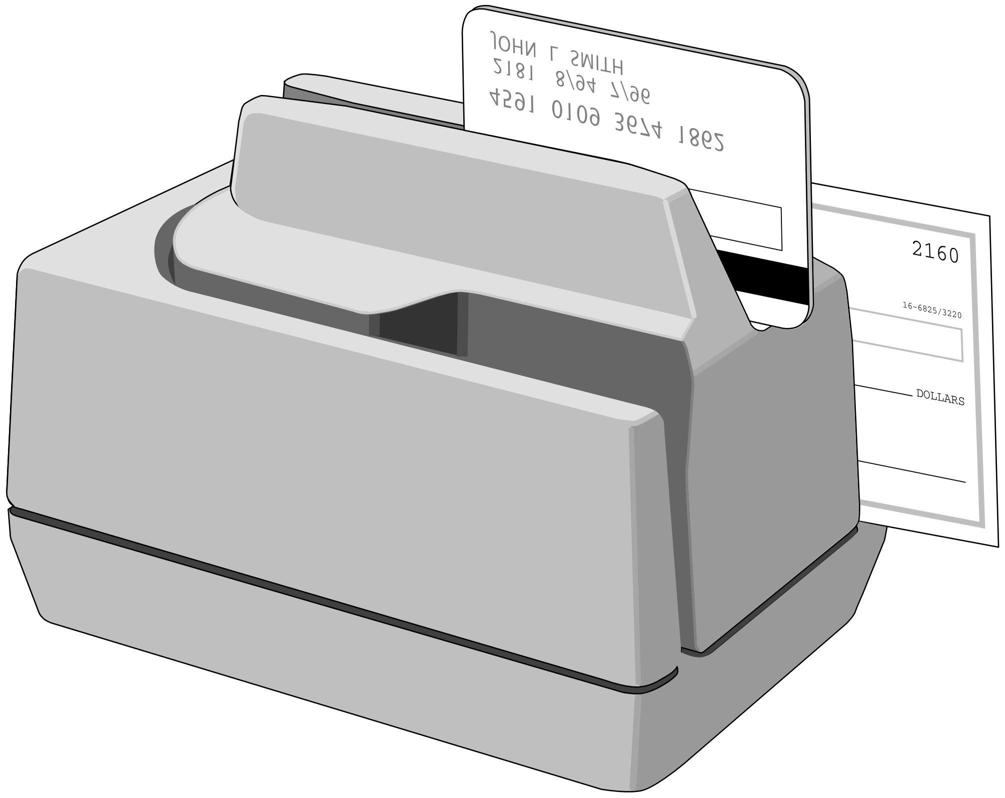


Figure 1-1. MINI MICR USB with 3-Track MSR

SECTION 1. OVERVIEW

The MINI MICR USB KEYBOARD EMULATION With Optional 3-Track MSR is both a MICR (Magnetic Ink Character Recognition) Check Reader and an MSR (Magnetic Stripe Reader).

The MICR Reader, in a typical application, reads the magnetic data encoded on the bottom of checks or magnetic stripe cards and transmits this data to a Host device. The Host device then uses a specific authorization or verification process to validate a business transaction.

The use of the MICR Reader improves accuracy and speed because there is no manual data entry; therefore there are no keying errors or unwanted delays.

This device emulates a USB keyboard. This device is compatible with PCs or hosts that support USB keyboards.

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 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 this product should only be used if the application requires data to be provided via the keyboard input. If previous applications were based upon RS-232 serial interface on a Windows operating system, it is recommended that you use MagTek's MINI MICR USB Virtual COM Port product. (Refer to Technical Manual 99875252 for further information regarding the MINI MICR USB Virtual COM Port product.)

FEATURES

- Available with MICR Reader only or with 3-Track or 2-Track MSR.
- Three track MSR autodiscriminates different card formats: ISO (International Standards Organization), CDL (California Drivers License), or AAMVA (American Association of Motor Vehicle Administrators).
- Small footprint.
- Automatic parsing of MICR fields: transit, account, etc.
- Extensive list of formats to transmit MICR data.
- Optional error/status reporting for check reading.
- Reads E13-B and CMC-7 MICR fonts.

- EMF noise detection
- In addition to the USB interface, the MICR Reader is also available with other interfaces.
- 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.

ACCESSORIES

Accessories available for the MICR Reader are as follows:

- Interface Cable, 9-pin Mini Din, Male, USB A Plug, 6', Beige, Part Number 22517582, or
- Interface Cable, 9-pin Mini Din, Male, USB A Plug, 6', MT Gray, Part Number 22517583
- AC Power Adapter with Cable, 120VAC to 12 VAC, 1 Amp, Part Number 64300050
- MICR Reader Cleaning Card, Part Number 96700006
- Sample Checks, Part Number 96530005
- USB MSR Demo Program with Source Code (disk) 21042806
- USB MSR Demo Program with Source Code (WEB) 99510026

SOFTWARE DRIVERS REQUIRED

The standard HID and Keyboard drivers that come with an operating system are usually all that is needed. For example, the Windows operating system provides all the drivers needed to communicate to the device.

REFERENCE DOCUMENTS

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

USB Human Interface Device (HID) Class Specification Version 1.1

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

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

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

SPECIFICATIONS

Table 1-1 lists the specifications for the MICR Reader.

Table 1-1. Specifications

OPERATING	
Reference Standards	ISO/CDL/AAMVA
Power Input	120 VAC, 50/60 Hz
Output Signal Levels	12 VAC, 1 Amp
Check Read/Decode/Transit Time	1 second
MICR fonts supported	E13-B CMC-7
MSR supported	Tracks 1, 2, and 3; or Tracks 1 and 2
MECHANICAL	
Dimensions	Length 6.0”, Width 4.0”, Height 4.25”
Weight:	3.0 lbs. MSR and Adapter included
Cable length	6’
Connectors	DIN-9, USB A
ENVIRONMENTAL	
Temperature	
Operating	0^oC to 50^oC (32^oF to 122^oF)
Storage	-30^oC to 70^oC (-22^oF to 158^oF)
Humidity	
Operating	10% to 90% noncondensing
Storage	Up to 100% noncondensing

SECTION 2. INSTALLATION

The installation for the MICR Reader is as follows:

REQUIREMENTS

The following is required for the Installation:

- MINI MICR USB With Optional MSR
- Interface Cable, 9-pin Mini Din, Male, USB A Plug, 6’, Beige, Part Number 22517582, or
- Interface Cable, 9-pin Mini Din, Male, USB A Plug, 6’, MT Gray, Part Number 22517583
- AC Power Adapter with Cable, 120VAC to 12 VAC, 1 Amp, Part Number 64300050

PROCEDURE

Perform the following steps:

1. On the interface cable connect the USB A connector to the PC.
2. On the interface cable connect the 9-pin male DIN connector to the MICR Reader.
3. On the AC power adapter, connect the jack to the plug on the MICR Reader.
4. On the AC power adapter, connect the plug to the wall outlet.
5. The first time the Reader is connected to the PC, Windows will need to install the USB driver. See the instructions below.
6. The LED indicator on the MICR Reader should turn on to a steady green. The LED indicator is located below the slot where the check is first inserted for reading.

Caution

Do not place the MICR Reader within 6 inches of a computer monitor or power supply. These devices may cause undesirable interference with the check reading operation.

USB DRIVER INSTALLATION (WINDOWS)

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.

SECTION 3. OPERATION

This section contains check and card reading procedures and LED indicator states.

CHECK READING PROCEDURE

1. Orient the check so the MICR line is down and the printed side faces the center on the MICR Reader as shown in Figure 3-1.

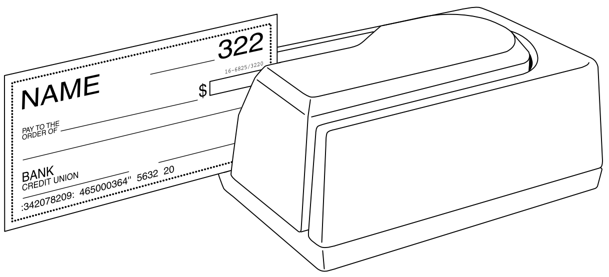


Figure 3-1. Check Orientation

2. Drop the check so the leading edge is in the open slot.

3. When the MICR Reader detects the presence of the check, the motor will turn on. At this time gently urge the check forward until the unit grabs the check. When this happens, release the check. The check will then be transported around the check path and will exit through the other side.

4. After the check is read, the MICR Reader will transmit the data as specified by the parameters described in Section 4, Commands.

CARD SWIPE PROCEDURE

The card may be swiped through the MSR in either direction, but the magnetic stripe must be oriented in only one direction as shown in Figure 1-1. The MSR will transmit raw card data (“as is” on the card) for all tracks that have been enabled using the HW (Hardware) command (Section 4, Commands).

The MSR is capable of reading ISO, AAMVA, and CDL encoded cards. The MSR will autodiscriminate all the card formats when the ID Card Decoding option is enabled using the HW (Hardware) command (Section 4, Commands).

LED INDICATORS

Table 3-1 describes the LED indicator conditions for check and card reading operations. The LED indicator is located below the slot where the check is first inserted for reading.

Table 3-1. LED indicators

LED INDICATOR	DESCRIPTION
OFF	Power off
SOLID GREEN	Ready to read check or card
OFF→ SOLID RED	Check or card read error
OFF→ SOLID GREEN	Good read
FLASH GREEN	Needs initialization*
FLASH RED/GREEN	Data sensor blocked (motor does not run)*
FLASH RED	Motor sensor blocked (motor does not run)*
FLASH GREEN FAST	Monitor mode (factory use only)*

*Refer to "Appendix C. Troubleshooting Guide."

SECTION 4. LEGACY COMMANDS

This section describes the use of commands and programmable options available for the MICR Reader.

Note
All options described below can be factory set as specified by the user when ordering.

To execute the MICR Reader commands, either one of two methods is required: Insta-Change checks or the USBMSR Program for Windows.

INSTA-CHANGE CHECKS

The first method is the use of Insta-Change checks, which is a more practical way of setting up the MICR Reader for most applications. The Insta-Change check is a MICR encoded document that contains commands and options used to reset the parameters of the MICR Reader. Multiple commands and options may be contained on one Insta-Change check. When used, the Insta-Change checks are run through the MICR Reader the same as a standard check, and the options to be used are automatically selected. To obtain Insta-Change checks, notify a MagTek representative and specify what options will be used. To operate Insta-Change checks, install the MICR Reader as described in Section 2, and watch the LED indicator. When the Insta-Change check is run through the MICR Reader and read successfully, the LED indicator will blink green. If the LED indicator turns red, the read is not successful. Try again or use a different Insta-Change check.

USBMSR DEMO PROGRAM FOR WINDOWS

The USBMSR program (P/N 21042806) allows the user to control all the programmable options available in the MICR Reader. Note that only USBMSR version 1.08 or newer with work with this device.

The program allows manual entry of commands and it also displays data from cards and checks that are read. All legacy MICR commands found in this section must be sent to the device using the “Send Legacy Command Command”. Details of this command can be found in the “USB Communications” section of this manual. For details and examples of how to use USBMSR see the USBMSR Demo section of this manual. For more detailed information also refer to the Readme.txt file that comes with this program.

The USBMSR program may also be downloaded from the internet at www.magtek.com under Software/Demo Programs USB Swipe & Insert Reader.

COMMAND FORMAT

When the commands are entered manually, they must use the following format:

`[COMMAND][DATA]<CR>`

where:

- **`[COMMAND]`** is 2 or 3 alpha characters.
- **`[Data]`** is optional as described below for each command.
- **`<CR>`** is always required.
- All characters are ASCII
- No spaces, brackets, or angle brackets required.

All legacy MICR commands found in this section must be sent to the device using the "Send Legacy Command Command". Details of this command can be found in the "USB Communications" section of this manual.

SWA - SWITCH A COMMAND

The SWA command controls the communication parameters, shown in Table 4-1. The data for this command consists of 8 ASCII bits ("0" = hex 30 and "1" = hex 31).

Table 4-1. SWA Command

BITS								PARAMETERS
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
x	x	x	x	x	x	x	x	Not Used

To execute, send the SWA command as follows:

`SWA 01010101<CR>` (with data)
or
`SWA <CR>` (without data)

When sending data, all 8 bits must be provided. The MICR Reader will execute the command but it will not reply. To make this command permanent, use the SA (Save) command described at the end of this section.

If no data is sent, the MICR Reader responds with the current settings for SWA.

SWA PARAMETERS

SWA has no affect on the Reader and is included only to maintain compatibility with our other MICR Readers.

SWB - SWITCH B COMMAND

The SWB command controls the message format, shown in Table 4-2. The data for this command consists of 8 ASCII bits (“0” = hex 30 and “1” = hex 31).

To execute, send the SWB command as follows:

`SWB 01010101<CR>` (with data)

or

`SWB <CR>` (without data)

When sending data, all 8 bits must be provided. The MICR Reader will execute the command but it will not reply. The new settings become effective immediately. To make this command permanent, use the command SA (Save) described at the end of this section.

If no data is sent, the MICR Reader responds with the current settings for SWB.

Table 4-2. SWB Command

BIT								PARAMETERS
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
							0	<LF>: No
							1	<LF>: Yes
						0		<CR>: No
						1		<CR>: Yes
					0			<ETX>: No
					1			<ETX>: Yes
				0				<ESC>: No
				1				<ESC>: Yes
			0					<STX>: No
			1					<STX>: Yes
		0						Send Data After Error?: No
		1						Send Data After Error?: Yes
	0							Send Status After Data?: No
	1							Send Status After Data?: Yes
0			0	0	0	0	0	Comm Mode: 0 - Data Only
1			0	0	0	0	0	Comm Mode: 1 - Data <CR>
0			0	0	0	0	1	Comm Mode: 2 - Data -<LF>
0			0	0	0	1	1	Comm Mode: 3 - Data -<CR><LF>
0			0	1	0	0	0	Comm Mode: 4 - <ESC> Data
0			0	1	0	1	0	Comm Mode: 5 - <ESC> Data<CR>
0			1	0	1	0	0	Comm Mode: 6 - <STX> Data<ETX>
1			0	0	0	0	1	Comm Mode: 7 - <STX>Data<ETX><LRC>

SWB PARAMETERS

The SWB functions are listed in Table 4-2 and described below.

Control Characters and MICR Data

Control Characters may be added to the MICR data message. The characters are always in the following locations:

```
<STX> <ESC> data <ETX> <CR> <LF>
```

The control characters, descriptions, and hex values are shown in Table 4-3.

Table 4-3. Control Characters

CONTROL CHARACTER	DESCRIPTION	HEX VALUE
<STX>	Start of Text	02
<ESC>	Escape	1B
<ETX>	End of Text	03
<CR>	Carriage Return	0D
<LF>	Line Feed	0A

For example, if <STX> and <CR> are set to YES, the message from the MICR Reader will look like this:

```
MICR Data: <STX>data<CR>
```

Control Characters and Card Data

The control characters are also available for card data but they are applied to each track individually. For example, if the <STX> and <ETX> options are set to YES, the card data message is transmitted as follows:

```
Card Data:  <STX>[TK1 data]<ETX><STX>[TK2 data]<ETX><STX>[TK3
data]<ETX>
```

Communication Modes

The selection of comm modes is a quick way of selecting multiple Control Characters. For instance, to send a carriage return/line feed pair after the data, you can specify Comm Mode 3.

Comm Mode 7, also known as Packet Mode, calculates an LRC (Longitudinal Redundancy Check), and appends it to the data message. Also, if a <NAK> (hex 15) character is received in this mode, the MICR Reader will resend the last message.

Send Data After Error

The request Send Data After Error specifies whether the MICR Reader will return data to the Host after a read error. If YES is selected and the MICR Reader detects a read error, the MICR Reader will still send the data back to the Host. If NO is selected and the MICR Reader finds an error, it will discard the data and nothing will be sent. The error conditions are listed in Table 4-4.

Send Status After Data

The Send Status After Data option makes the MICR Reader append a two-digit error/status code to the end of the MICR data. For most formats (See Appendix A), the error/status code will always be preceded by a forward slash (/). The error/status codes are listed in Table 4-4.

For example, if a Canadian check (code 08) is read and had no errors, and the MICR data is "1234567890", then the message from the MICR Reader will look like this:

```
MICR Data:  1234567890/08
```

The status code is always at the end of the data, not the end of the message. For example, using the above conditions, with the message format set to send <STX> and <ETX>, the message from the MICR Reader will look like this:

```
MICR Data:  <STX>1234567890/08<ETX>
```

Table 4-4. Error and Status Codes

PRIORITY	CODE	TYPE	DESCRIPTION
9	01	Error	No MICR data: no transit and no account found
8	09	Status	Mexican check
7	08	Status	Canadian check
6	05	Error	Transit error: No transit, bad character, bad length, bad check digit
5	07	Error	Account error: No account, bad character
4	04	Error	Check # error: Bad character in check number
4	04	Status	No check number
3	03	Status	Low MICR signal, good read
2	10	Status	Business check
1	11	Status	Amount field present
0	00	Status	Good read

Notes:

- The LED indicator will turn red on all error conditions.
- The absence of a check number is not considered and error.
- If a multiple error condition occurs, the error or status code with the highest priority is reported.
- All unreadable MICR characters are transmitted as an "?" ASCII character (hex 3F), except for Format 00xx (See Appendix A).

SWC - SWITCH C COMMAND

The SWC command controls miscellaneous functions, shown in Table 4-5. The data for this command consists of 8 ASCII bits ("0" = hex 30 and "1" = hex 31).

To execute, send the SWC command as follows:

`SWC 01010101<CR>` (with data)
or
`SWC <CR>` (without data)

When sending data, all 8 bits must be provided. The MICR Reader will execute the command but it will not reply. The new settings become effective immediately. To make this command permanent, use the SA (Save) command described at the end of this section.

If no data is sent, the MICR Reader responds with the current settings for SWC.

Table 4-5. SWC Command

BITS								PARAMETERS
7	6	5	4	3	2	1	0	
							0	CMC-7 Character Set: No
							1	CMC-7 Character Set: Yes
					0	0		Invalid Commands: ?<CR>
					0	1		Invalid Commands : No Reply (Header Required)*
					1	0		Invalid Commands: No Reply (No Header Required)
					1	1		Ignore all Commands
				0				Reserved
			0					Data Header: No
			1					Data Header: Yes
		0						Card Data Message: Multiple
		1						Card Data Message: Single
0	0							These bits are always set to 0 but must be included.

*Header Required means all commands must be preceded by a GS character (Hex 1D).

SWC PARAMETERS

The SWC functions are listed in Table 4-5 and described below.

CMC-7 Character Set

If NO is selected the MICR Reader will only read E13-B characters. When YES is selected, the MICR Reader will read both CMC-7 and E13-B characters (see Appendix B). However, the MICR Reader will only output raw data ("as is" on the check) for checks with CMC-7 characters.

Invalid Command Response

Invalid command response is the action the MICR Reader takes upon receipt of a command it does not recognize. It can also be used to stop the MICR Reader from receiving any more commands.

The first option "**`?<CR>"`** is the default. If the MICR Reader receives an unrecognized command, it will return a question mark and carriage return to the Host. The MICR Reader will then return to an idle state and wait for further commands or check/credit card reads.

For the second option, "no reply - header required," the MICR Reader will only execute commands preceded by a GS ASCII character (hex 1D). All other commands will be ignored. Also, the MICR Reader will not reply to invalid commands.

For the third option, "no reply," the MICR Reader will execute all valid commands, but it will not reply to invalid commands.

The fourth option, "ignore all commands," causes the MICR Reader to ignore any further commands. Even the SA (Save) command is ignored and therefore this fourth option is only temporary. To make this option permanent or to reset it, you must use an Insta-Change check.

Data Header

If YES is selected, a single character header precedes the data. For MICR data, the message is transmitted as follows:

```
MICR data:     'C'[data]
```

For card data, the header position on the message is controlled by the Card Data Message parameter (see below). Therefore, the message may be transmitted as follows:

If Multiple Message: ‘M’[TK1]‘M’[TK2]’M’[TK3]

If Single Message: ‘M’[TK1] [TK2] [TK3]

It is important to note that the Data Header precedes the data and not the message. For example, if <STX>, <ETX> and Data Header are set to YES, a MICR data message will be transmitted as follows:

MICR data: <STX>‘C’[data]<ETX>

Card Data Message

This option determines the structure of the output message for the individual tracks when a credit card is read. If Multiple is selected, the Control Characters (see SWB, below) and Data Header (see Data Header, above) are added to each track individually. On the other hand, if Single is selected, all available tracks are lumped together into a single message. For example, if <STX>, <ETX> and Data Header are set to YES, the output message may be transmitted as follows:

If Multiple Message:
<STX>‘M’[TK1]<ETX><STX>‘M’[TK2]<ETX><STX>‘M’[TK3]<ETX>

If Single Message: <STX>‘M’[TK1] [TK2] [TK3]<ETX>

HW - HARDWARE COMMAND

This command controls miscellaneous hardware options, shown in Table 4-6. The data for this command consists of 8 ASCII bits (“0” = hex 30 and “1” = hex 31).

To execute, send the HW command as follows:

HW 01010101<CR> (with data)
or
HW <CR> (without data)

When sending data, all 8 bits must be provided. The MICR Reader will execute the command but it will not reply. The new settings become effective immediately. To make this command permanent, use the SA (Save) command described at the end of this section.

If no data is sent, the MICR Reader responds with the current settings for HW.

Table 4-6. HW Command

7	6	5	4	3	2	1	0	PARAMETERS
					0			Track 3: Disable
					1			Track 3: Enable
				0				Track 2: Disable
				1				Track 2: Enable
			0					Track 1: Disable
			1					Track 1: Enable
		0						ID Card decoding: Disable
		1						ID Card decoding: Enable
						0		EMF detect: Yes
						1		EMF detect: No
0	0						0	These bits are always set to 0

HW PARAMETERS

Disable/Enable Tracks

Each Track can be enabled or disabled individually. The tracks are always transmitted in ascending order: TK1, TK2, TK3. For example, if TK1 and TK3 are enabled and TK2 is disabled, the MSR will transmit TK1, TK3.

ID Card Decoding

The MSR has two modes of operation. In the first mode, ID Card decoding disabled, the MSR will only read ISO encoded cards. In the second mode, ID Card decoding enabled, the MSR will read and autodiscriminate ISO, AAMVA, and CDL encoded cards. When a card is swiped, the LED indicator will turn red and indicate an error if any of the enabled tracks read is incompatible with the selected mode of operation. TK2 is a standard track for all types of cards.

EMF Detect

The EMF Detect option allows the MICR Reader, when idle, to monitor EMF interference in its immediate environment. If YES is selected, the LED indicator will blink red/green when the MICR Reader detects a signal with amplitude large enough to affect check reading. If NO is selected, the MICR Reader will not monitor nor indicate the presence of EMF interference.

FC - FORMAT CHANGE COMMAND

Formats are used by the MICR Reader to process and transmit the MICR fields. The format command allows the selection of a format from the Format List, Appendix A. The data for this command consists of 4 digits (ASCII characters 0-9). To execute, send the command as follows:

`FC 6600<CR>` (with data)

or

`FC <CR>` (without data)

When sending data, all 4 digits must be provided. The MICR Reader will execute the command but it will not reply. The new settings become effective immediately. To make this command permanent, use the SA (Save) command described below.

If no data is provided, the MICR Reader will respond with the current format number.

VR - VERSION COMMAND

The Version command gives the current software revision in the MICR Reader. To execute, send the VR command followed by a carriage return as follows:

```
VR<CR>
```

The MICR Reader responds as follows:

```
MICR data:  [software revision]<CR>
```

LE - LED COMMAND

To control the LED, the LE command is sent with a hexadecimal digit (use ASCII characters for the hex digit):

`LE` ***`X`*****`<Enter>`**

Where ***X*** = Hex digit **`0-F`**.

An example of the "Blink Red" command is:

```
LE 9<Enter>
```

The codes and descriptions are shown in Table 4-7. The LE command will control the LED for three seconds and then return it to the normal state. The description column is a common expression of the state of the LED.

Table 4-7. LED Control

Color Cycle	Hex Digit	Description
Off/Off/Off/Off	0	LED Off
Green/Green/Green/Green	1	Steady Green
Red/Red/Red/Red/	2	Steady Red
Amber/Amber/Amber/Amber	3	Steady Amber
Green/Green/Off/Off	4	Blink Green
Red/Red/Off/Off	5	Blink Red
Amber/Amber/Off/Off	6	Blink Amber
Red/Red/Green/Green	7	Blink Red/Green
Green/Off/Green/Off	8	Blink Green
Red/Off/Red/Off	9	Blink Red
Amber/Off/Amber/Off	A	Blink Amber
Red/Green/Red/Green	B	Blink Red/Green
Red/Green/Off/Off	C	Blink Red/Green
Green/Green/Green/Red	D	Blink Red/Green
Red/Red/Red/Green	E	Blink Red/Green
Off/Off/Off/Off	F	Off

SA - SAVE COMMAND

All changes are considered temporary until the Save command is executed. The Save command saves all changes to the MICR Reader memory and makes them permanent. The MICR Reader will execute the command but it will not reply. To execute, send the SA command followed by a carriage return as follows:

SA<CR>

RS - RESET COMMAND

The Reset command resets the MICR firmware to the normal operating state of waiting for a check or card to be read. The command also resets the serial port to the most recent settings provided by the SWA command. To execute, send the RS command followed by a carriage return as follows:

RS<CR>

SECTION 5. 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 keyboard 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 0x2251.

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 AND MICR DATA

The card and MICR data is converted to ASCII and transmitted to the host as if it had been typed on a 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.

Because of potential “data interleave” issues associated with the USB Keyboard interface, MagTek recommends that this product should only be used if the application requires data to be provided via the keyboard input. If previous applications were based upon RS-232 serial interface on a Windows operating system, it is recommended that you use MagTek’s MINI MICR USB Virtual COM Port product. (Refer to Technical Manual 99875252 for further information regarding the MINI MICR USB Virtual COM Port product.)

The device's programmable configuration options affect the format of the card and MICR data. Refer to the legacy commands section for a description of how the card and MICR data is formatted. Some of the properties in this section also affect the format of the card and MICR data.

All data will be sent in upper case regardless of the state of the caps lock key on the keyboard.

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.

HID USAGES

HID devices send data in reports. Elements of data in a report are identified by unique identifiers called usages. The structure of the device's reports and the device's capabilities are reported to the host in a report descriptor. The host usually gets the report descriptor only once, right after the device is plugged in. The report descriptor usages identify the devices capabilities and report structures. For example, a device could be identified as a keyboard by analyzing the device's report descriptor. Usages are four byte integers. The most significant two bytes are called the usage page and the least significant two bytes are called usage IDs. Usages that are related can share a common usage page. Usages can be standardized or they can be vendor defined. Standardized usages such as usages for mice and keyboards can be found in the HID Usage Tables document and can be downloaded free at www.usb.org. Vendor defined usages must have a usage page in the range 0xff00 – 0xffff. All usages for this device use 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
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**. The requests are sent over the default control pipe. When a command request is sent, the device will Nak the Status stage of the **Set Report** request until the command is completed. This insures that as soon as the **Set Report** request is completed, the **Get Report** request can be sent to get the command response. The usage ID for the command message was shown previously in the Usage Table.

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

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

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

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

COMMAND NUMBER

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

Value	Command Number	Description
0	**Get Property**	Gets a property from the device
1	**Set Property**	Sets a property in the device
2	**Reset Device**	Resets the device
3	**Get Keymap Item**	Gets a key map item
4	**Set Keymap Item**	Sets a key map item
5	**Save Custom Keymap**	Saves the custom key map
7	**Send Legacy Command**	Sends a legacy command

DATA LENGTH

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

DATA

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

RESULT CODE

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

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

GET AND SET PROPERTY COMMANDS

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

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

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

Get Property Request Data:

Data Offset	Value
0	Property ID

Get Property Response Data:

Data Offset	Value
0 – n	Property Value

Set Property Request Data:

Data Offset	Value
0	Property ID
1 – n	Property Value

Set Property Response Data:
None
The result codes for the Get and Set Property commands can be any of the codes list in the generic result code table.

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

Value	Property ID	Description
0	**Software ID**	The device's software identifier
1	**Serial Num**	The device's serial number
2	**Polling Interval**	The interrupt pipe's polling interval
4	**Track Data Send Flags**	Track data send flags
15	**ASCII To Keypress Conversion Type**	Type of conversion performed when converting ASCII data to key strokes
17	**Active Keymap**	Selects which key map to use
26	**Convert From Char A**	Selects character to use when converting to string A
27	**Convert To String A**	Selects string to use when converting from char A
28	**Convert From Char B**	Selects character to use when converting to string B
29	**Convert To String B**	Selects string to use when converting from char B

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

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

SOFTWARE ID PROPERTY

Property ID:	0
Property Type:	String
Length:	Fixed at 11 bytes
Get Property:	Yes
Set Property:	No
Description:	This is an 11 byte read only property that identifies the software part number and version for the devices USB CPU. The first 8 bytes represent the part number and the last 3 bytes represent the version. For example this string might be "22826820A01". 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 32 38 32 36 38 32 30 41 30 31

SERIAL NUM PROPERTY

Property ID:	1
Property Type:	String
Length:	0 – 15 bytes
Get Property:	Yes
Set Property:	Yes
Default Value:	The default value is no string with a length of zero.
Description:	The value is an ASCII string that represents the device's serial number. This string can be 0 – 15 bytes long. The value of this property, if any, will be sent to the host when the host requests the USB string descriptor. This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect.

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

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

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

Result Code	Data Len	Data
00	00	

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

Cmd Num	Data Len	Prp ID
00	01	01

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

Result Code	Data Len	Prp Value
00	03	31 32 33

POLLING INTERVAL PROPERTY

Property ID:	2
Property Type:	Byte
Length:	1 byte
Get Property:	Yes
Set Property:	Yes
Default Value:	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 keystroke data packets. For example, if the polling interval is set to 10, the host will poll the device for keystroke data packets every 10ms. This property can be used to speed up or slow down the time it takes to send keystroke 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 DATA SEND FLAGS PROPERTY

Property ID:	4
Property Type:	Byte
Length:	1 byte
Get Property:	Yes
Set Property:	Yes
Default Value:	0x00
Description:	This property is defined as follows:

ICL	0	0	0	0	0	0	0

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

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:	15 (0x0F)
Property Type:	Byte
Length:	1 byte
Get Property:	Yes
Set Property:	Yes
Default Value:	0 (keymap)
Description:	The value is a byte that represents the devices ASCII to keypress conversion type. The value can be set to 0 for keymap (The active keymap is set with the **Active Keymap** property) or to 1 for ALT ASCII code (international keyboard emulation). When the value is set to 0 (keymap), data will be transmitted to the host according to the active keymap which defaults to the United States keyboard keymap. For example, to transmit the ASCII character '?' (063 decimal), the character is looked up in a keymap. For a United States keyboard keymap, the '/' (forward slash) key combined with the left shift key modifier are stored in the keymap to represent the key press combination that is used to represent the ASCII character '?' (063 decimal). When the value is set to 1 (ALT ASCII code), instead of using the key map, 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

ACTIVE KEYMAP PROPERTY

Property ID:	17 (0x11)
Property Type:	Byte
Length:	1 byte
Get Property:	Yes
Set Property:	Yes
Default Value:	0 (United States)
Description:	The value is a byte that represents the device’s active key map. The value can be set to 0 for the United States key map or to 1 for the custom key map. The active key map will be used by the device to convert ASCII data into key strokes. The United States key map should be used with 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

CONVERT FROM CHAR A PROPERTY

Property ID:	26 (0x1a)
Property Type:	Byte
Length:	1 byte
Get Property:	Yes
Set Property:	Yes
Default Value:	255 (0xff) (None)

Description: The value is a byte that represents the ASCII value of a character transmitted by the device as keystroke data that is to be changed into a string of ASCII values prior to being transmitted by the device as keystroke data. The string of ASCII values that this value will be changed into is contained in the **Convert To String A** property. If the value of this property is set to 0xff, no characters will be changed into the string. For example, if you would like a carriage return to be sent as an end of text character you could set the **Convert From Char A** property to 0x0d (carriage return) and set the **Convert To String A** property to 0x03 (end of text). If you would like a carriage return to be sent as two carriage returns you could set the **Convert From Char A** property to 0x0d (carriage return) and set the **Convert To String A** property to 0x0d 0x0d (carriage return, carriage return). If you would like a carriage return to not be sent you could set the **Convert From Char A** property to 0x0d (carriage return) and set the **Convert To String A** property to no string.

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 **Convert From Char A** property Request (Hex):

Cmd Num	Data Len	Prp ID	Prp Value
01	02	1a	ff

Example Set **Convert From Char A** property Response (Hex):

Result Code	Data Len	Data
00	00	

Example Get **Convert From Char A** property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	1a

Example **Convert From Char A** property Response (Hex):

Result Code	Data Len	Prp Value
00	01	ff

CONVERT TO STRING A PROPERTY

Property ID:	27 (0x1b)
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 **Convert To String A** property. This string can be 0 – 7 bytes long. This string is sent in place of the character specified in the **Convert From Char A** property. See the **Convert From Char A** property for more information and examples.

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 **Convert To String A** property Request (Hex):

Cmd Num	Data Len	Prp ID	Prp Value
01	03	1b	0d 0d

Example Set **Convert To String A** property Response (Hex):

Result Code	Data Len	Data
00	00	

Example Get **Convert To String A** property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	1b

Example Get **Convert To String A** property Response (Hex):

Result Code	Data Len	Prp Value
00	02	0d 0d

CONVERT FROM CHAR B PROPERTY

Property ID:	28 (0x1c)
Property Type:	Byte
Length:	1 byte
Get Property:	Yes

Set Property:	Yes
Default Value:	255 (0xff) (None)
Description:	The value is a byte that represents the ASCII value of a character transmitted by the device as keystroke data that is to be changed into a string of ASCII values prior to being transmitted by the device as keystroke data. The string of ASCII values that this value will be changed into is contained in the **Convert To String B** property. If the value of this property is set to 0xff, no characters will be changed into the string. For example, if you would like a carriage return to be sent as an end of text character you could set the **Convert From Char B** property to 0x0d (carriage return) and set the **Convert To String B** property to 0x03 (end of text). If you would like a carriage return to be sent as two carriage returns you could set the **Convert From Char B** property to 0x0d (carriage return) and set the **Convert To String B** property to 0x0d 0x0d (carriage return, carriage return). If you would like a carriage return to not be sent you could set the **Convert From Char B** property to 0x0d (carriage return) and set the **Convert To String B** property to no string.

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 **Convert From Char B** property Request (Hex):

Cmd Num	Data Len	Prp ID	Prp Value
01	02	1c	ff

Example Set **Convert From Char B** property Response (Hex):

Result Code	Data Len	Data
00	00	

Example Get **Convert From Char B** property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	1c

Example **Convert From Char B** property Response (Hex):

Result Code	Data Len	Prp Value
00	01	ff

CONVERT TO STRING B PROPERTY

Property ID:	29 (0x1d)
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 **Convert To String B** property. This string can be 0 – 7 bytes long. This string is sent in place of the character specified in the **Convert From Char B** property. See the **Convert From Char B** property for more information and examples.

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 **Convert To String B** property Request (Hex):

Cmd Num	Data Len	Prp ID	Prp Value
01	03	1d	0d 0d

Example Set **Convert To String B** property Response (Hex):

Result Code	Data Len	Data
00	00	

Example Get **Convert To String B** property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	1d

Example Get **Convert To String B** property Response (Hex):

Result Code	Data Len	Prp Value
00	02	0d 0d

RESET DEVICE COMMAND

Command number: 2

Description: This command is used to reset the devices USB CPU. 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: 3

Description: This command is used to get a key map item from the active key map. The active key map is determined by the active key map property. Data from a card or check 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 E. 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 F.

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: 4

Description: This command is used to set a key map item of the active key map. The active key map is determined by the active key map property. Data from a card or check 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 E. 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 F. 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.

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: 5

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

Data structure:

Request Data: None
Response Data: None

Result codes: 0 (success)

Example Request (Hex):

Cmd Num	Data Len	Data
05	00	

Example Response (Hex):

Result Code	Data Len	Data
00	00	

SEND LEGACY COMMAND COMMAND

Command number: 7

Description: This command is used to send legacy commands to the device. Legacy commands are sent to the device over USB by using this command. The device will send up to two responses to this command. The first response is the standard response to a USB command. This response indicates to the host that the USB command was received successfully. If the legacy command is expected to send a response, then this second response will be returned to the host as keystrokes. See the legacy command section for all of the legacy commands.

Data structure:

Request Data: Legacy command to be sent including carriage return if any. The legacy command must be converted from ASCII to binary data prior to sending it. For example, the command SWA<CR> must be sent as 53 57 41 0d (hex). The legacy command can be sent one character at a time or all at once up to the maximum size allowed by a set feature report (22 bytes). For example, the command SWA<CR> can also be sent by issuing the **Send Legacy Command command** 4 times, once for each of the four characters. In this case the response to the legacy command will not be sent by the device until after the <CR> is sent.

Response Data: None

Result codes: 0 (success)

Example Request (Hex):

Cmd Num	**Data Len**	**Data** (SWA<CR>)
07	04	53 57 41 0d

Example Response (Hex):

Result Code	**Data Len**	**Data**
00	00	

Example Legacy Command Response (send as keystrokes):
SWA=00000000<CR>

SECTION 6. USBMSR DEMO PROGRAM

The primary purpose of this demo program is not to demonstrate card reading with this Keyboard Emulation device. A text editor application such as Windows Notepad can 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 what 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 program version 1.08 or newer is required to work with this device.

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

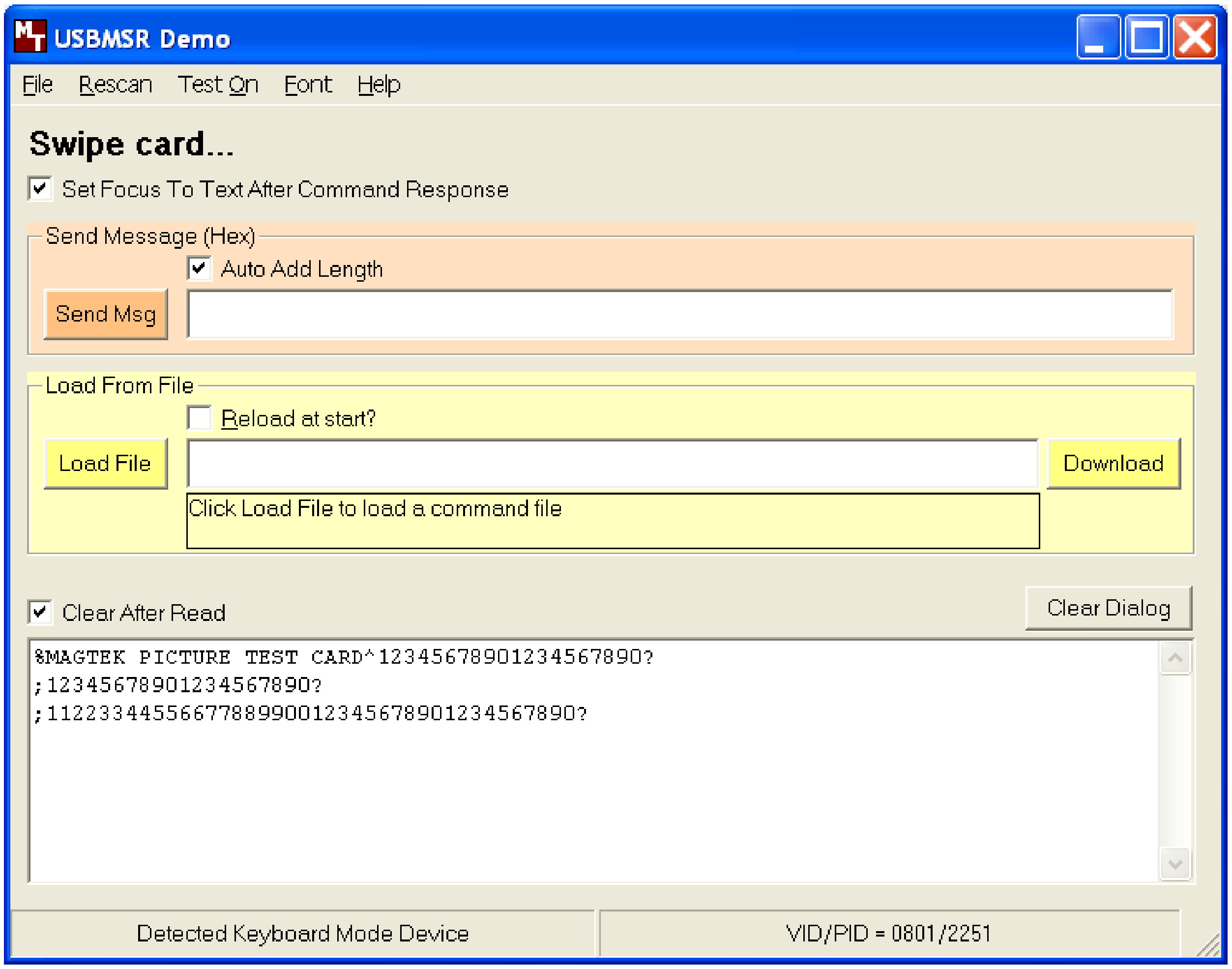
INSTALLATION

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

OPERATION

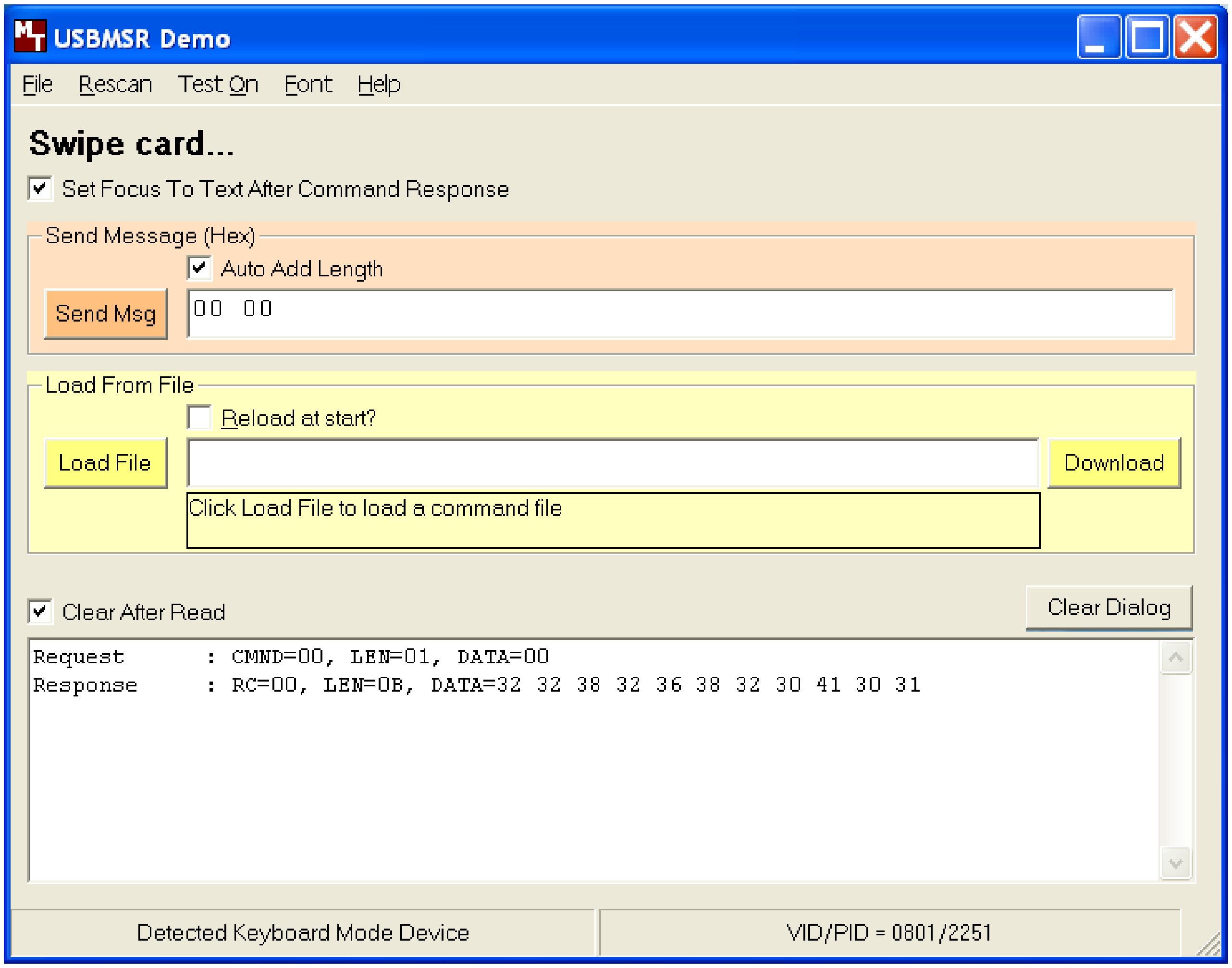
To operate the demo program perform the following steps:

- 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 a card or a check wait for the program to display "Swipe card…" and make sure the curser is in the text box at the bottom of the program then swipe a card or read a check. The data output from the reader will be displayed in the text box at the bottom of the screen. The following is an example of what a card read would look like.



```
%MAGTEK PICTURE TEST CARD^12345678901234567890?
;12345678901234567890?
;1122334455667788990012345678901234567890?
```

- To send commands to the device, do the following.
- Enter a command in the Send Message text box. All data entered should be in hexadecimal bytes with a space between each byte. Enter the command number followed by the command data if there is any. **The application will automatically calculate and send the command data length for you if you have the "Auto Add Length" check box checked.** For example, to send the **Get Property** command for property **Software ID** enter 00 00.
- Press Enter or click on "Send Msg" button to send the command and receive the result.
- The command request and the command result will be displayed in the text box at the bottom of the screen. The following is an example of what sending the get software ID property would look like.



☑ Set Focus To Text After Command Response

☑ Auto Add Length

00 00

☐ Reload at start?

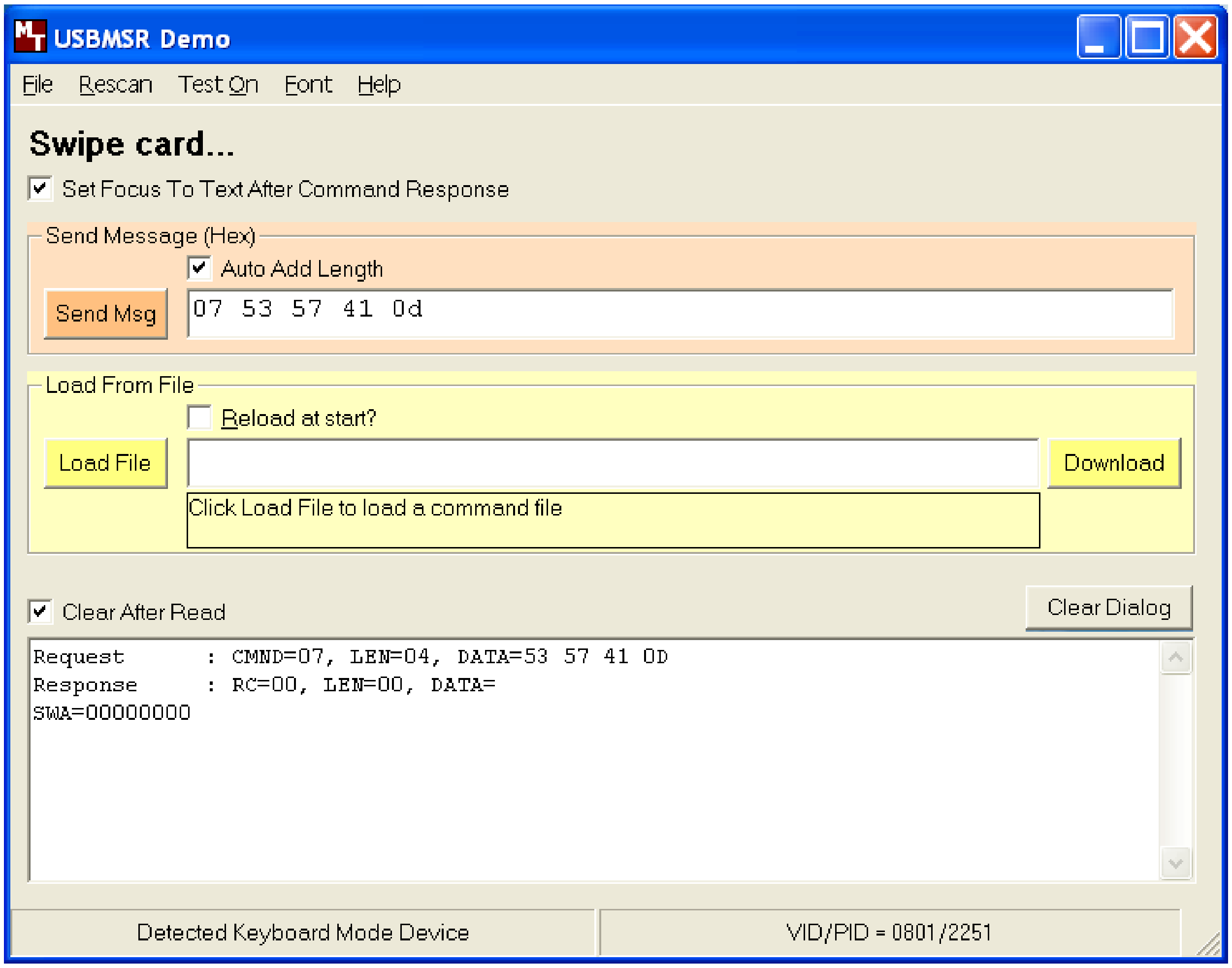
Click Load File to load a command file

☑ Clear After Read

```
Request      : CMND=00, LEN=01, DATA=00
Response     : RC=00, LEN=0B, DATA=32 32 38 32 36 38 32 30 41 30 31
```

Detected Keyboard Mode Device | VID/PID = 0801/2251

- Legacy MICR commands can be sent in the same way as regular commands except the “Set Focus To Text After Command Response” check box should be checked to make sure the response to the Legacy MICR command that is sent as keystroke data is displayed correctly in the text box at the bottom of the screen. See the “Send Legacy Command Command” area of the “USB Communications” section of this document for details about the format of this command. See the “Legacy Commands” section of this document for details of all of the legacy MICR commands. The following example demonstrates how to send the “SWA<CR>” command to the device. Notice that the “SWA<CR>” command has to be converted from ASCII to hex before sending it.



Swipe card...

✓ Set Focus To Text After Command Response

Send Message (Hex)

✓ Auto Add Length

Send Msg `07 53 57 41 0d`

Load From File

☐ Reload at start?

Load File Download

Click Load File to load a command file

✓ Clear After Read Clear Dialog

```
Request      : CMND=07, LEN=04, DATA=53 57 41 0D
Response     : RC=00, LEN=00, DATA=
SWA=00000000
```

Detected Keyboard Mode Device VID/PID = 0801/2251

- 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. FORMAT LIST

For check reading, the MICR Reader provides the flexibility to format the MICR fields and build a specific output string that will be transmitted to the Host. These output strings are referred to as formats. The Reader has a built-in list of formats (described below) from which the user may select one to become the active format every time a check is read. The formats may be selected using the FC command (Section 4, Commands) or Insta-Change checks provided by MagTek.

Each format is assigned a 4-digit number. The first two digits indicate the format number, and the last two digits are specific parameters used for various functions by each format. For example, in format “0415”, the “04” refers to format number 4 and the 15 refers the maximum number of characters allowed for the account field.

Note
The formats listed in this section apply only to U.S. and Canadian checks. The MICR line on checks from other countries will not be broken or parsed as described in these formats.

A complete description for each format follows.

```
Fmt  00xx: Raw Data Format - sends the entire MICR line - where:
           Xx          - specify what symbol set to use.  Choose from the
table
           Add xx + 16 - change multiple spaces to one space
           Add xx + 32 - Remove all spaces

           Examples:

              MICR LINE:   T122000218T  1234 5678 9U   1321
                  FC0001 - t122000218t  1234 5678 9o   1321
            (+16) FC0017 - t122000218t 1234 5678 9o 1321
            (+32) FC0033 - t122000218t123456789o1321.
```

xx	Transit	On-Us	Amount	Dash	Error
00	T	U	$	-	?
01	t	o	a	d	?
02	T	O	A	D	?
03	T	U	$	-	*
04	T	U	$	0	?
05	T	U	$	0	*
06	t	o	a	0	?
07	T	U	$	none	?

Fmt 01xx:...Parsed Text Format

```
FC0100      - Parsed text with dashes
FC0101      - Parsed text, replace dashes with “d”
Field Labels - TR-transit, AC-account #, CK-check #, AM-amount,
TP-tpc, EP-epc
Example:    - PTTR444455556;AC 999-222-3;CK11045
```

Fmt 02xx: Parsed Text Format with Error Labels

```
FC0200      - Parsed text with dashes
FC0201      - Parsed text, replace dashes with “d”
Error Labels - PE-parsed error, NE-no error, TR-transit error,
                 CK-chk # error, TC-transit check digit error,
                 AM-amount error, OU-on us/account# error, TP-tpc error
Examples:   - PTTR444455556;AC999-222-3;CK11045/PENE
            - PTTR111?11111;AC123456/PETR ("?" = unreadable
            character)
```

Fmt 03xx: **[acct #]**

• **[acct #]**: - maximum of xx characters; when xx=00 all characters are sent
- keep spaces and dashes

Fmt 04xx: **[acct #]**

• **[acct #]**: - maximum of xx characters; when xx=00 all characters are sent
- remove spaces and dashes

Fmt 05xx: **[acct #]**

• **[acct #]**: - maximum of xx characters; when xx=00 all characters are sent
- replace spaces and dashes with zeros

Fmt 06xx: **[acct #]**

• **[acct #]**: - always xx characters, zero filled;
when xx=00 all characters are sent
- replace spaces and dashes with zeros

Fmt 07xx: **[acct #]**

• **[acct #]**: - always xx characters, zero filled;
when xx=00 all characters are sent
- remove spaces and dashes

Fmt 08xx: **[transit] [acct #]**

• **[transit]**: - all characters in the field
- keep dashes

• **[acct #]**: - maximum of xx characters; when xx=00 all characters are sent
- remove spaces and dashes

Fmt 09xx: **[transit] [acct #]**

• **[transit]**: - all characters in the field
- keep dashes

• **[acct #]**: - maximum of xx characters; when xx=00 all characters are sent
- replace spaces and dashes with zeros

Fmt 10xx: **[transit] [acct #]**

- **[transit]**:
  - all characters in the field
  - keep dashes
- **[acct #]**:
  - always xx characters, zero filled; when xx=00 all characters are sent
  - replace spaces and dashes with zeros

Fmt 11xx: **[transit] 'T' [acct #] 'A' [check #]**

- **[transit]**:
  - all characters in the field
  - keep dashes
- **[acct #]**:
  - maximum of xx characters; when xx=00 all characters are sent
  - remove spaces and dashes
- **[check #]**:
  - all characters in the field

Fmt 12xx: **[transit] 'T' [acct #] 'A' [check #]**

- **[transit]**:
  - all characters in the field
  - keep dashes
- **[acct #]**:
  - maximum of xx characters; when xx=00 all characters are sent
  - remove spaces and dashes
- **[check #]**:
  - always 6 characters, zero filled

Fmt 13xx: **[transit] 'T' [acct #] 'A' [check #] '000'**

- **[transit]**:
  - all characters in the field
  - keep dashes
- **[acct #]**:
  - maximum of xx characters; when xx=00 all characters are sent
  - remove spaces and dashes
- **[check #]**:
  - always 6 characters, zero filled

Fmt 14xx: **[transit] [acct #] [check #]**

- **[transit]**:
  - all characters in the field
  - keep dashes
- **[acct #]**:
  - maximum of xx characters; when xx=00 all characters are sent
  - remove spaces and dashes
- **[check #]**:
  - always 6 characters, zero filled

Fmt 15xx: **[bank #] [acct #]**

- **[bank #]**:
  - all characters in the field
  - keep spaces and dashes
- **[acct #]**:
  - maximum of xx characters; when xx=00 all characters are sent
  - remove spaces and dashes

Fmt 16xx: **[bank #] [chk dgt] [acct #]**

- **[bank #]**: - all characters in the field
  - keep spaces and dashes
- **[chk dgt]**: - all characters (one character long)
- **[acct #]**: - maximum of xx characters; when xx=00 all characters are sent
  - remove spaces and dashes

Fmt 17xx: **[transit] [acct #]**

- **[transit]**: - all characters in the field
  - keep dashes
- **[acct #]**: - maximum of xx characters; when xx=00 all characters are sent
  - keep spaces and dashes

Fmt 18xx: **[acct #] "/" [check #]**

- **[acct #]**: - maximum of xx characters; when xx=00 all characters are sent
  - keep spaces and dashes
- **[check #]**: - all characters in the field

Fmt 19xx: **[transit] [acct #] [check #]**

- **[transit]**: - all characters in the field
  - keep dashes
- **[acct #]**: - maximum of xx characters; when xx=00 all characters are sent
  - replace spaces and dashes with zeros
- **[check #]**: - all characters in the field

Fmt 20xx: **[transit] [acct #] <CR> [check #]**

- **[transit]**: - all characters in the field
  - keep dashes
- **[acct #]**: - maximum of xx characters; when xx=00 all characters are sent
  - replace spaces and dashes with zeros
- **[check #]**: - all characters in the field

Fmt 21xx: **[transit] [acct #] [check #]**

- **[transit]**: - all characters in the field
  - keep dashes
- **[acct #]**: - always xx characters, zero filled;
  when xx=00 all characters are sent
  - replace spaces and dashes with zeros
- **[check #]**: - all characters in the field

Fmt 22xx: **[bank #] [acct #] [check #]**

```
• [bank #]:    - all characters in the field
               - keep dashes

• [acct #]:    - always xx characters, zero filled;
                 when xx=00 all characters are sent
               - replace spaces and dashes with zeros

• [check #]:   - all characters in the field
```

Fmt 23xx: **[error #] [transit] [acct #] [check #] 'S'**

```
• [error #]:   - one digit, always present
               - '0' read OK
               -  '1'  read  error:  bad  char,  empty  field,  invalid  length,
validation

• [transit]:   - always 9 characters, zero filled
               - keep dashes

• [acct #]:    - always xx characters, trailing spaces;
                 when xx=00 all characters are sent
               - remove spaces and dashes

• [check #]:   - always 6 characters, zero filled
               - remove spaces and dashes
```

Fmt 24xx: **[transit] 'T' [acct #] 'A' [check #] 'C' [amount] '$'**

```
• [transit]:   - all characters in the field
               - keep dashes

• [acct #]:    - maximum of xx characters; when xx=00 all characters are sent
               - remove spaces and dashes

• [check #]:   - always 6 characters, zero filled

• [amount]:    - all characters in the field
```

Fmt 25xx: **'M' 'C' [transit] 'D' [acct #] 'E' [check #]**

```
• [transit]:   - all characters in the field
               - remove dashes and keep spaces (contig spcs = 1 spc)
               - if the field is empty, remove 'C'

• [acct #]:    - include leading characters
               - maximum of xx characters; when xx=00 all characters are sent
               - remove dashes and keep all spaces
               - if the field is empty, remove 'D'

• [check #]:   - all characters in the field
               - if the field is empty, remove 'E'
```

Fmt 26xx: **[acct #]**

```
• [acct #]:    - work with characters in acct and transit fields
               - a window of xx characters; xx must be greater than 00
               - remove spaces and dashes
```

Fmt 27xx: **[acct #]**

- **[acct #]**: - work with characters in the acct field only
  - a window of xx characters; xx must be greater than 00
  - remove spaces and dashes

Fmt 28xx: **[acct #]**

- **[acct #]**: - work with characters in the acct field only
  - a window of xx characters; xx must be greater than 00
  - minimum of 6 digits, fill with zeros if necessary
  - remove spaces and dashes

Fmt 29xx: **'C' '/' [transit] '/' [acct #] '/' [check #] '/' [status]**

- **[transit]**: - all characters in the field
  - keep dashes

- **[acct #]**: - maximum of xx characters; when xx=00 all characters are sent
  - remove spaces and dashes

- **[check #]**: - maximum of 6 digits

- **[status]**: - this is a programmable option that must be enabled (See Table 4-4).

Fmt 30xx: **[zero fill] [transit] [acct #]**

- **[zero fill]**: - if length of (transit+account) is less than xx; xx must be greater than 00
- **[transit]**: - all characters in the field
  - remove dashes

- **[acct #]**: - all characters in the field
  - remove spaces and dashes

Fmt 31xx: **[transit] '/' [acct #] '/' [check #]**

- **[transit]**: - all characters in the field
  - remove dashes

- **[acct #]**: - maximum of xx characters; when xx=00 all characters are sent
  - remove spaces and dashes

- **[check #]**: - maximum of 10 digits
  - remove spaces and dashes
  - if no check number, remove preceding slash ('/')

Fmt 3200: **'^' [transit] '^' [acct #] '^' [check #] '^' [status]**

- **[transit]**: - all characters in the field
  - remove dashes

- **[acct #]**: - all characters in the field
  - remove spaces and dashes

- **[check #]**: - all characters in the field
  - remove spaces and dashes

- **[status]** : - this is a programmable option that must be enabled (See Table 4-4).

Fmt 3300:   **'=' [transit] '=' [acct #] '=' [check #] '=' [status]**

- **[transit]**: - all characters in the field
  - remove dashes

- **[acct #]** : - maximum of 14 digits
  - remove spaces and dashes

- **[check #]**: - maximum of 8 digits
  - remove spaces and dashes

- **[status]**: - this is a programmable option that must be enabled (See Table 4-4).

Fmt 34xx: **[transit] [acct #] [zero fill]**

- **[transit]**: - all characters in the field
  - remove dashes

- **[acct #]**: - all characters in the field
  - remove spaces and dashes

- **[zero fill]**: - zero filled up to xx; xx must be greater than 00

Fmt 3500:   **MA [aux] B [epc] C [tran] D [acct] E [chk] F [tpc] G [amt]**

This format is defined specifically for Target Test Checks. A description of the Target Test Check must be loaded in the exception table.

- **[aux], [epc], [tran], [chk], [tpc], [amt]**:
  - all characters in the field
  - keep spaces and dashes

- **[acct]**: - all characters in the field
  - keep spaces and remove dashes

Fmt 36xx: Read OK   : **[transit] [acct #] [check #] '/'**
Read error: **'0' '/'**

- **[transit]**: - all characters in the field
  - remove spaces and dashes

- **[acct #]**: - maximum of xx characters; when xx=00 all characters are sent
  - remove spaces and dashes

- **[check #]**: - always 6 characters, zero filled
  - remove spaces and dashes

Fmt 37xx: **[ABA] [chk dgt] [acct #]**

- **[ABA], [chk dgt]**:
  - all characters in the field
  - keep spaces and dashes

- **[acct #]**: - work with characters in the acct field only
  - window of xx characters; xx must be greater than 00
  - remove spaces and dashes

Fmt 38xx: **'T' [transit] 'A' [acct #] 'C' [check #]**

- **[transit]**: - all characters in the field
  - keep dashes
- **[acct #]**: - maximum of xx characters; when xx=00 all characters are sent
  - include leading characters
  - keep spaces and dashes

- **[check #]**: -all characters in the field

Fmt 39xx: **[transit] <CR> [acct #]**

- **[transit]**: - all characters in the field
  - remove dashes

- **[acct #]**: - maximum of xx characters; when xx=00 all characters are sent
  - remove spaces and keep dashes

Fmt 40xx: **[country code] [transit] [acct #]**

- **[country code]**: - '1' for US checks
  - '2' for Canadian checks

- **[transit]**: - all characters in the field
  - remove dashes

- **[acct #]**: - maximum of xx characters; when xx=00 all characters are sent
  - remove spaces and dashes

Fmt 4100: **'S' 'T' [transit] 'A' [acct #] 'C' [check #]**

- **[transit]**: - all characters in the field
  - remove dashes

- **[acct #]**: - all characters in the field
  - place a slash ('/') after 10th character
  - if 10 characters or less, precede with a slash ('/')
  - remove spaces and dashes
- **[check #]**: - always 6 characters, zero filled
  remove spaces and dashes

Fmt 42xx: US check : **[transit] [acct #]**

Can check: **'9' [transit] [acct #]**

- **[transit]**: - all characters in the field
  - remove dashes

- **[acct #]**: - always xx characters; zero filled;
  when xx=00 all characters are sent.
  - remove spaces and dashes

Fmt 43xx: **[check #] <CR> <CR> [transit] <CR> [acct #]**

- **[check #]**: - maximum of 6 digits
  - remove spaces and dashes

- **[transit]**: - all characters in the field
  - remove dashes

- **[acct #]**: - maximum of xx characters; when xx=00 all characters are sent
  - remove spaces and dashes

Fmt 44xx: **[transit] [acct #]**

- **[transit]**: - all characters in the field
  - if Canadian check, replace dash with a space
- **[acct #]**: - always xx characters, trailing spaces, when xx=00 all characters are sent
  - remove spaces and dashes

Fmt 45xx: **[transit] <CR> [acct #] <CR> [check #]**

- **[transit]**: - all characters in the field
  - remove dashes
- **[acct #]**: - maximum of xx characters; when xx=00 all characters are sent
  - remove spaces, dashes and leading zeros
- **[check #]**: - all characters in the field

Fmt 46xx: **[transit] [acct #] [check #]**

- **[transit]**: - all characters in the field
  - remove dashes
- **[acct #]**: - always xx characters, zero filled; when xx=00 all characters are sent
  - remove spaces and dashes
- **[check #]**: - always 6 characters, zero filled
  - remove spaces and dashes

Fmt 47xx: **[transit] 'T' [acct #] 'A' [check #]**

- **[transit]**: - all characters in the field
  - remove dashes
- **[acct #]**: - maximum of xx characters; when xx=00 all characters are sent
  - remove spaces and dashes
- **[check #]**: - all characters in the field

Fmt 48xx: **[transit] 'T' [acct #] 'A'**

- **[transit]**: - all characters in the field
  - remove dashes
- **[acct #]**: - maximum of xx characters; when xx=00 all characters are sent
  - remove spaces and dashes

Fmt 49xx: **[transit] '/' [acct #] '/' [check #] '/' [check type]**

- **[transit]**: - always 9 characters, zero filled
  - remove dashes
- **[acct #]**: - maximum of xx characters; when xx=00 all characters are sent
  - remove spaces and dashes
- **[check #]**: - maximum of 9 digits
- **[check type]**:- personal checks ('1'); commercial checks ('2')

Fmt 50xx: **'T' [transit] 'T' 'O' [acct #] 'O' [check #]**

- **[transit]**: - all characters in the field
  - remove dashes
- **[acct #]**: - maximum of xx characters; when xx=00 all characters are sent
  - remove spaces and dashes
- **[check #]**: - all characters in the field

Fmt 51xx: **'=' [transit] '=' [acct #] '='**

- **[transit]**: - all characters in the field
  - remove dashes
- **[acct #]**: - maximum of xx characters; when xx=00 all characters are sent
  - remove spaces and dashes

Fmt 52xx: **'T' [transit] 'T' [acct #] 'A' [check #]**

- **[transit]**: - all characters in the field
  - remove dashes
- **[acct #]**: - maximum of xx characters; when xx=00 all characters are sent
  - remove spaces and dashes
- **[check #]**: - all characters in the field
  - remove dashes and spaces

Fmt 53xx: **'/' [transit] '/' [acct #] '/' [check #] '/' [tpc] '/' [status] '/'**

- **[transit]**: - all characters in the field
  - remove dashes
- **[acct #]**: - maximum of xx characters; when xx=00 all characters are sent
  - remove spaces and dashes
- **[check #]**: - all characters in the field
- **[tpc]**: - all characters in the field
- **[status]**: - this is a programmable option that must be enabled (See Table 4-4)

Fmt 54xx: **[transit] [acct #] [check #] [status]**

- **[transit]**: - always 12 characters, zero filled
  - remove dashes
- **[acct #]**: - always xx characters, zero filled; when xx=00 all characters are sent
  - remove spaces and dashes
- **[check #]**: - always 12 characters, zero filled
  - remove dashes and spaces
- **[status]**: - this is a programmable option that must be enabled (See Table 4-4)

Fmt 55xx: **'C' '/' [acct #] '/' [transit] '/' [check #] '/' 0000000000**

- **[acct #]**: - always xx characters, zero filled; when xx=00 all characters are sent
  - remove spaces and dashes
- **[transit]**: - all characters in the field
  - remove dashes
- **[check #]**: - always 6 characters, zero filled
  - remove dashes and spaces

Fmt 56xx: **[transit] <CR> [acct #] <CR> [check #] <CR> [amount]**

- **[transit]**: - all characters in the field
  - remove dashes
- **[acct #]**: - maximum of xx characters; when xx=00 all characters are sent
  - remove spaces and dashes
- **[check #]**: - all characters in the field
  - remove dashes and spaces
- **[amount]**: - all characters in the field
  - remove dashes and spaces

Fmt 57xx: **[acct #] <CR> [amount]**

- **[acct #]**: - maximum of xx characters; when xx=00 all characters are sent
  - remove spaces and dashes
- **[amount]**: - all characters in the field
  - remove dashes and spaces

Fmt 58xx: **[short transit] [acct #] ':'**

- **[transit]**: - 3 rightmost characters
  - remove dashes
- **[acct #]**: - maximum of xx characters; when xx=00 all characters are sent
  - remove spaces and dashes

Fmt 59xx: **[transit] [acct #] <TAB> [check #] [amount]**

- **[transit]**: - all characters in the field
  - remove dashes
- **[acct #]**: - maximum of xx characters; when xx=00 all characters are sent
  - remove spaces and dashes
- **[check #]**: - always 9 characters, zero filled
  - remove dashes and spaces
- **[amount]**: - all characters in the field
  - remove dashes and spaces
  - insert decimal point ('.') before 2nd rightmost digit

Fmt 60xx: **[transit] '/' [acct #] '/' [check #] '/' [check type]**

- **[transit]**: - all characters in the field
  - remove dashes
- **[acct #]**: - maximum of xx characters; when xx=00 all characters are sent
  - remove spaces and dashes
- **[check #]**: - maximum of 10 characters
  - remove spaces and dashes
  - if no check #, remove preceding slash ('/')
- **[check type]**:- personal checks ('1'); commercial checks ('2')

Fmt 61xx: **[transit] <TAB> [acct #] <TAB> [check #] <TAB>**

- **[transit]**: - all characters in the field
  - remove dashes
- **[acct #]**: - maximum of xx characters; when xx=00 all characters are sent
  - remove spaces, dashes and leading zeros
- **[check #]**: - all characters in the field

Fmt 62xx: **'T' [transit] 'T' [acct #] 'A' [check #] 'S' [status]**

- **[transit]**: - all characters in the field
  - remove dashes
- **[acct #]**: - maximum of xx characters; when xx=00 all characters are sent
  - remove spaces and dashes
- **[check #]**: - all characters in the field
  - remove dashes and spaces
- **[status]**: - this is a programmable option that must be enabled (See Table 4-4).

Fmt 63xx: **[transit] [acct #] [check #]**

- **[transit]**: - all characters in the field
  - remove dashes
- **[acct #]**: - maximum of xx characters; when xx=00 all characters are sent
  - remove spaces and dashes
- **[check #]**: - always 4 characters, zero filled
  - remove spaces and dashes

Fmt 64xx: **[transit] [acct #] [check #] [amount]**

- **[transit]**: - all characters in the field
  - keep dashes
- **[acct #]**: - always xx characters, trailing spaces; when xx=00 all characters are sent
  - keep spaces and dashes
- **[check #]**: - always 6 characters (N is on quick-init check), trailing spaces
  - remove spaces and dashes
- **[amount]**: - all characters in the field
  - remove spaces and dashes
  - insert decimal point ('.') before 2nd rightmost digit

Fmt 65xx: **'!' [transit] '/' [acct #] '/' [check #] '/' [amount]**

- **[transit]**: - all characters in the field
  - remove dashes
- **[acct #]**: - maximum of xx characters; when xx=00 all characters are sent
  - remove spaces and dashes
- **[check #]**: - all characters in the field
  - remove dashes and spaces
- **[amount]**: - all characters in the field
  - remove dashes and spaces

Fmt 66xx: **[transit] [acct #] <CR> '7' '1' <CR>**

- **[transit]**: - all characters in the field
  - keep dashes
- **[acct #]**: - maximum of xx characters; when xx=00 all characters are sent
  - remove spaces and dashes

Fmt 67xx: **<CR> <CR> [check #]**

- **[check #]** : - maximum of xx characters; when x=00 all characters are sent
  - remove spaces and dashes

Fmt 68xx: **[transit] <TAB> [acct #] <TAB> [check #] <TAB> [amount] <TAB>**

- **[transit]**: - all characters in the field
  - remove dashes
- **[acct #]**: - maximum of xx characters; when xx=00 all characters are sent
  - remove spaces and dashes
- **[check #]**: - all characters in the field
  - remove dashes and spaces
- **[amount]**: - all characters in the field
  - remove dashes, spaces and leading zeros
  - insert decimal point ('.') before 2nd rightmost digit

Fmt 69xx: Read OK : **[transit] [acct #] [check #]**

Read error: **'0' '/'**

- **[transit]**:
  - all characters in the field
  - remove dashes
- **[acct #]**:
  - always xx characters, trailing spaces; when xx=00 all characters are sent
  - remove spaces and dashes
- **[check #]**:
  - always 6 characters, zero filled
  - remove dashes and spaces

Fmt 70: **[transit] ',' [acct #] ',' [check #] ',' [amount]**

- **[transit]**:
  - all characters in the field
  - keep dashes
- **[acct #]**:
  - always N characters (N is on quick-init check), space filled
  - remove spaces and dashes from the account
- **[check #]**:
  - always 8 characters, zero filled
  - remove dashes and spaces
- **[amount]**:
  - all characters in the field
  - remove dashes and spaces
  - if amount is not present, remove last ','

Fmt 71: **[acct #] '?' [check #]**

- **[acct #]**:
  - work with a window of N characters in the acct field
  - always N characters (N is on quick-init check), zero filled
  - remove spaces and dashes
- **[check #]**:
  - maximum of 4 characters
  - remove spaces and dashes

Fmt 72: **[transit] <TAB> [acct #]**

- **[transit]**:
  - all characters in the field
  - remove dashes
- **[acct #]**:
  - maximum of N characters (N is on quick-init check)
  - remove spaces and dashes

Fmt 73: **[transit] <CR> [acct #] <CR> [check #]**

- **[transit]**:
  - all characters in the field
  - remove dashes
- **[acct #]**:
  - maximum of N characters (N is on quick-init check)
  - remove spaces and dashes
- **[check #]**:
  - all characters in the field
  - remove dashes and spaces

Fmt 74: **[transit] [acct #] [check #]**

- **[transit]**: - all characters in the field
  - remove dashes
- **[acct #]**: - always N characters (N is on quick-init check), zero filled
  - remove spaces and dashes
- **[check #]**: - always 8 characters, zero filled
  - remove spaces and dashes

Fmt 75xx: **[transit] <CR> [acct #] <CR> [check #] <CR> [status]**

- **[transit]**: - always 9 characters, zero filled
  - keep dashes; remove spaces
- **[acct #]**: - maximum of xx characters; when xx=00 all characters are sent
  - remove dashes and spaces
- **[check #]**: - maximum of 12 characters
  - remove dashes and spaces

Fmt 76xx: **'T' [transit] 'A' [acct #] 'C' [check #] 'M' [raw data]**

- **[transit]**: - all characters in the field
  - remove dashes and spaces
- **[acct #]**: - maximum of xx characters; when xx=00 all characters are sent
  - remove dashes and spaces
- **[check #]**: - all characters in the field- remove dashes and spaces
- **[raw data]**: - translate MICR symbols to t,o,a,d

Fmt 7700: The Flexible Format

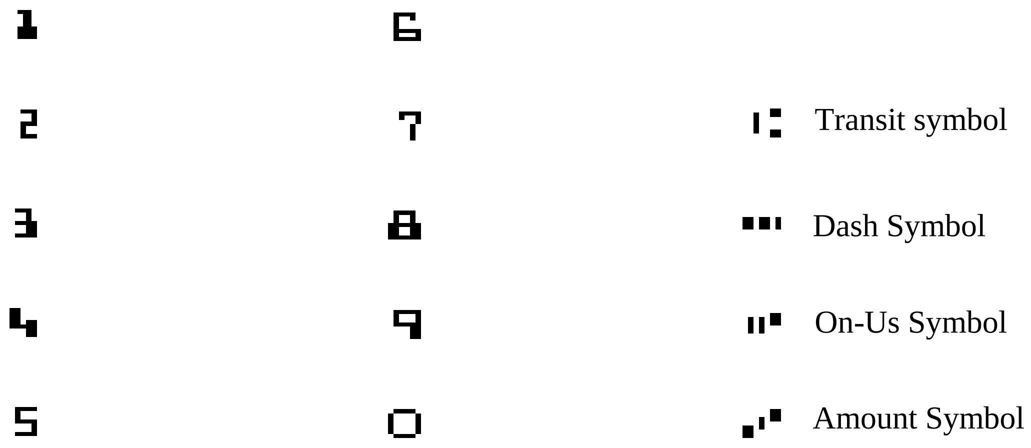
Select this format to activate a preloaded Flexible Format. The Flexible Format is a feature that allows the user to create custom MICR formats.

APPENDIX B. CHECK READING

The characters printed on the bottom line of commercial and personal checks are special. They are printed with magnetic ink to meet specific standards. These characters can be read by a MICR Reader at higher speeds and with more accuracy than manual data entry. Two MICR character sets are used world wide; they are: E13-B and CMC-7. The E13-B set is used in the US, Canada, Australia, United Kingdom, Japan, India, Mexico, Venezuela, Colombia, and the Far East. The CMC-7 set is used in France, Spain, other Mediterranean countries, and most South American countries.

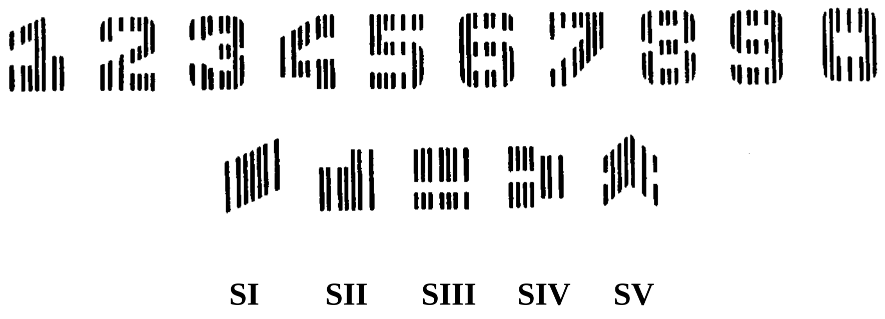
E13-B CHARACTER SET

The MICR font character set E13-B includes digits 0 through 9 and four symbols. The numbers found on U.S. checks are of the E13-B character set. The numbers and symbols of E13-B are as follows:



CMC-7 CHARACTER SET

The numbers and symbols of the CMC-7 character set are as follows:



The nonnumeric CMC-7 characters are translated by the MICR Reader as shown in Table B-1.

Table B-1. CMC-7 Nonnumeric Characters

CMC-7 Character	MICR Reader Output
SI	A
SII	B
SIII	C
SIV	D
SV	E

CHECK LAYOUTS

Personal checks with MICR fields are shown in Figure B-1. Business checks are shown in Figure B-2. The digits 1 through 4 in the illustrations are described below under MICR Fields.

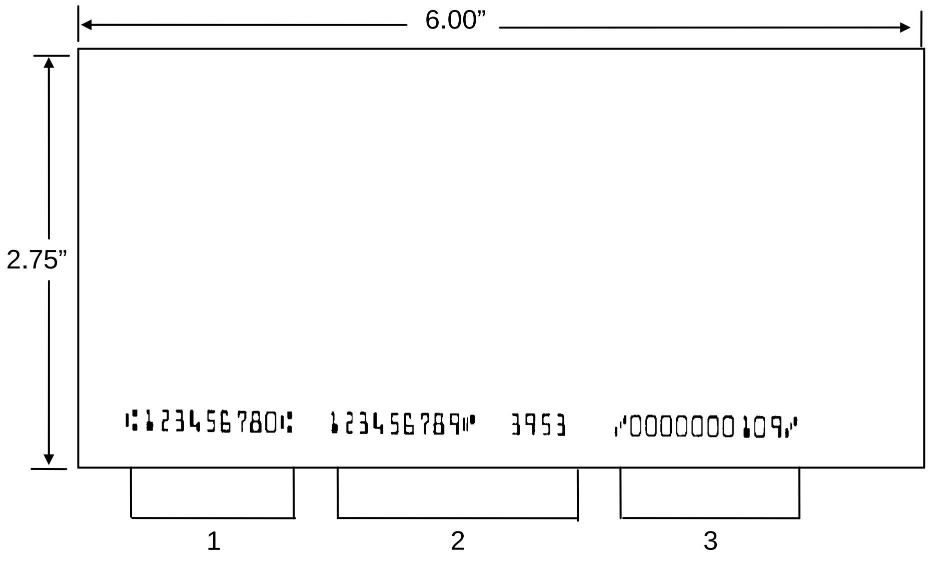


Figure B-1. Personal Checks

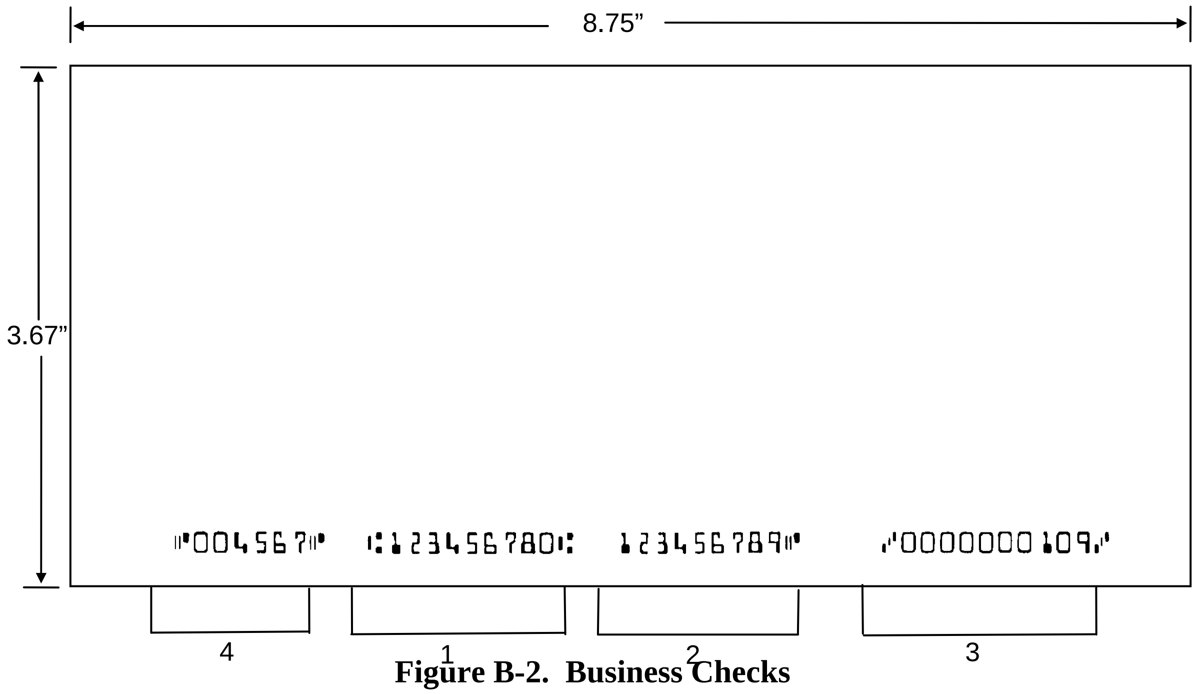


Figure B-2. Business Checks

MICR FIELDS

The numbers 1 through 4 refer to the numbers below the checks on the illustration and represent the 4 MICR fields.

1-Transit Field

The Transit field is a 9-digit field bracketed by two Transit symbols. The field is subdivided as follows:

• Digits 1-4	Federal Reserve Routing Number
• Digits 5-8	Bank ID Number (American Banking Association)
• Digit 9	Check Digit

2-On-Us Field

The On-Us field is variable, up to 19 characters (including symbols). Valid characters are digits, spaces, dashes, and On-Us symbols. The On-Us field contains the account number and may also contain a serial number (Check number) and/or a transaction code. Note that an On-Us symbol must always appear to the right of the account number.

3-Amount Field

The Amount field is a 10-digit field bracketed by Amount symbols. The field is always zero-filled to the left.

4-Auxiliary On-Us Field

The Auxiliary On-Us field is variable, 4-10 digits, bracketed by two On-Us symbols. This field is not present on personal checks. On business checks, this field contains the check serial number.

APPENDIX C. TROUBLESHOOTING GUIDE

REQUIREMENTS

- Personal Computer.
- USB Cable, P/N 22517582, or 22517583
- AC adapter, P/N 64300050.
- USBMSR program, P/N 21042806.
- Sample checks, P/N 96530005.
- A small bottle of compressed air.
- A cleaning card, P/N 96700006.

SET-UP

1. Plug the USB Interface Cable into the MICR and the host, 9-pin Mini Din, Male, USB A Plug, 6’, Part Number 22517582, or Part Number 22517583.
2. Power on the MICR Reader.
3. Run the USBMSR program on the PC.
4. Start trouble-shooting procedure at Step 00.

00	**CHECK LED**

Check the status of the LED indicator:

◊ off, continue to step 01.
◊ green, continue to step 02.
◊ blinking red, continue to step 11.
◊ blinking green, continue to step 16.
◊ blinking red/green, continue to 12.
◊ red or orange, continue to step 17.

01 CHECK THE POWER TO THE MICR READER

Possible causes for this problem are:

- AC adapter connection to outlet - make sure the AC adapter is securely connected to outlet on the wall or power strip.
- AC adapter connection to MICR Reader - make sure the AC adapter is securely connected to the power jack on the MICR Reader.
- Power strip - if using a power strip, make sure the strip is connected to outlet on the wall and the switch on the strip is turned on.
- AC adapter is defective - replace the AC adapter.

Determine if any of the conditions described above are true:

◊ If yes, rectify and continue to step 00.
◊ If no, continue to step 17.

02 READ A CHECK

Read a check through the MICR Reader:

◊ If the check is transported all the way around the check path, continue to step 03.
◊ If the check gets "stuck" in the check path, continue to step 10.
◊ If the motor does not turn on, continue to step 17.

03 DID PC RECEIVE DATA?

After the check is read, did the PC receive any data?

◊ If yes, continue to step 04.
◊ If no, continue to step 05

04 ANALYZE DATA

Analyze the data received by the PC:

◊ If the data is good, continue to step 15.
◊ If the data contains one or more '?', continue to step 06.
◊ If the data is missing characters, continue to step 07.
◊ If the data is garbled, continue to step 08.
◊ If the data is good but not what is expected, continue to step 09.

05 VERIFY PARAMETERS

Use USBMSR to verify the following parameters:
- "Send Data After Error" - if this option is set to NO, the MICR Reader will not send any data after a read error. Use MICRbase to change this option to YES.

Determine if any of the conditions described above are true:
◊ If yes, rectify and continue to step 02.
◊ If no, continue to step 13.

06 READ ERROR

Possible causes for this problem are:
- Interference - the MICR Reader may be too close to a monitor, AC adapter or magnetic device. Move the MICR Reader away from the source of interference.
- Printing problem - the check being read may not meet the requirements of the ANSI Standards. Use one the sample checks provided by MagTek.
- Feeding the check - do not hold on to the check as it goes around the path. Release the check immediately after the MICR Reader "grabs" it. Also, make sure that the front end is not tilted up while the check is being read.
- Foreign debris – power off the MICR Reader and try to push out any loose debris on the check path. Grab the cleaning card and force it through the check path (this is a manual process, the motor will not turn on). Try this procedure several times until the debris comes out. Power on the MICR Reader again.

Determine if any of the conditions described above are true:
◊ If yes, rectify and continue to step 02.
◊ If no, continue to step 14.

07 MISSING CHARACTERS

Possible causes for this problem are:
- Feeding the check - When feeding the check, make sure that the MICR line is at the bottom and the printed side of the check is facing the MagTek logo on the MICR Reader.

Determine if any of the conditions described above are true:
◊ If yes, rectify and continue to step 02.
◊ If no, continue to step 08.

08 COMMUNICATION PARAMETERS DO NOT MATCH

Verify that the communication parameters of the MICR Reader match the parameters of the PC. Use USBMSR to verify/change the communication parameters.

Determine if the condition described above is true:

◊ If yes, rectify and continue to step 02.
◊ If no, continue to step 14.

09 INCORRECT FORMAT

Possible causes for this problem are:

- Incorrect Format Number - the current Check data format in the MICR Reader is not the desired format. Use USBMSR to verify/change the format.
- Incorrect Message Format - the current Message format in the MICR Reader is not the desired format. Use USBMSR to verify/change the Message format.

Determine if any of the conditions described above are true:

◊ If yes, rectify and continue to step 02.
◊ If no, continue to step 17.

10 PATH IS OBSTRUCTED

Foreign debris is obstructing the check path:

- Loose debris - power off the MICR Reader and try to push out any loose debris on the check path. Grab the cleaning card and force it through the check path (this is a manual process, the motor will not turn on). Try this procedure several times until the debris comes out. Power on the MICR Reader.
- Wedged debris - the debris is wedged in and cannot be removed with the procedure described above.

Is the foreign debris removable?

◊ if yes, remove and continue to step 02.
◊ If no, continue to step 17.

11 MOTOR SENSOR IS BLOCKED

The Motor sensor may be blocked by dust build-up or foreign debris (see Figure C-1). Use forced air to clean the sensor.

Power off the MICR Reader and then power on again, observe the LED indicator:
- ◊ If the LED indicator blinks red, continue to step 17.
- ◊ Any other LED indicator status, continue to step 00.

12 EMF NOISE/INTERFERENCE

When idle, if EMF detect is set to YES (see HW Command, Section 4), the MICR Reader monitors the signal coming from the MICR head. If any signal (noise/interference) with amplitude large enough to affect check reading is detected, the LED indicator blinks red/green. Possible sources of EMF are monitors, AC adapters, or magnetic devices. Set EMF to NO, or move the MICR Reader at least 6 inches away from the source of noise/interference.

Determine if the condition described above is true:
- ◊ If yes, rectify and continue to step 00.
- ◊ If no, continue to step 13.

13 DATA SENSOR IS BLOCKED

The data sensor may be blocked (see Figure C-1). Try one or both of the following procedures:
- Forced air - use forced air to clean the sensor.
- Cleaning card - power off the MICR Reader and try to push out any loose debris on the check path. Grab the cleaning card and force it through the check path (this is a manual process, the motor will not turn on). Try this procedure several times until the debris comes out.

Power off the MICR Reader and then power on again, observe the LED indicator:
- ◊ If the LED indicator blinks red/green, continue to step 17.
- ◊ Any other LED indicator status, continue to step 00.
- ◊

14	NO MICR DATA DETECTED

Possible causes for this problem are:

- No MICR characters - the ink used to print the MICR characters does not have magnetic properties. Try one of the sample checks provided by MagTek.
- Feeding the check - When feeding the check, make sure that the MICR line is at the bottom and the printed side of the check is facing the MagTek logo on the MICR Reader.

Determine if any of the conditions described above are true:

◊ If yes, rectify and continue to step 02.
◊ If no, continue to step 14.

15	CABLE PROBLEM

Possible causes for this problem are:

- Loose connection - the cable connector on the PC or the MICR Reader may be loose. Make sure that both connectors are tightly connected.
- Damaged cable - the connectors, pins or wires in the cable may be damaged. Replace cable.

Determine if any of the conditions described above are true:

◊ If yes, rectify and continue to step 02.
◊ If no, continue to step 17.

16	NO PROBLEM FOUND

The MICR Reader is operating properly. If you have additional concerns or requirements please contact your MagTek representative.

17	READ INSTA-CHANGE CHECK

Read Insta-Change check with the appropriate settings. Return to step 00. If condition persists, continue to step 18.

18 RETURN MICR READER TO MAGTEK

The MICR Reader has a problem that needs further analysis, testing, and possibly repair. Please contact the MagTek Help Desk at (888) 624-8350, and make arrangements to send the unit back to MagTek. Include a detailed description of the problem.

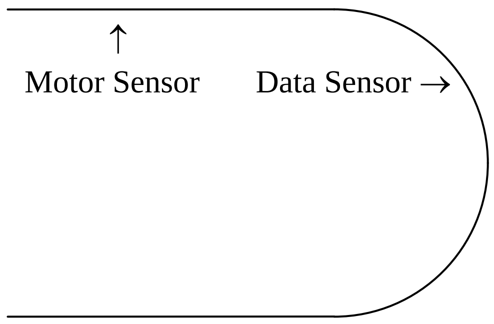


Figure C-1. Sensor Location

APPENDIX D. ASCII CODES

The following is a listing of the ASCII (American Standard Code for Information Interchange) codes. ASCII is a 7-bit code, which is represented here with a pair of hexadecimal digits.

ASCII	Hex	Dec	ASCII	Hex	Dec	ASCII	Hex	Dec	ASCII	Hex	Dec
NUL	00	0	SP	20	32	@	40	64	`	60	96
SOH	01	1	!	21	33	A	41	65	a	61	97
STX	02	2	"	22	34	B	42	66	b	62	98
ETX	03	3	#	23	35	C	43	67	c	63	99
EOT	04	4	$	24	36	D	44	68	d	64	100
ENQ	05	5	%	25	37	E	45	69	e	65	101
ACK	06	6	&	26	38	F	46	70	f	66	102
BEL	07	7	'	27	39	G	47	71	g	67	103
BS	08	8	(	28	40	H	48	72	h	68	104
HT	09	9	)	29	41	I	49	73	i	69	105
LF	0A	10	*	2A	42	J	4A	74	j	6A	106
VT	0B	11	+	2B	43	K	4B	75	k	6B	107
FF	0C	12	,	2C	44	L	4C	76	l	6C	108
CR	0D	13	-	2D	45	M	4D	77	m	6D	109
SO	0E	14	.	2E	46	N	4E	78	n	6E	110
SI	0F	15	/	2F	47	O	4F	79	o	6F	111
DLE	10	16	0	30	48	P	50	80	p	70	112
DC1	11	17	1	31	49	Q	51	81	q	71	113
DC2	12	18	2	32	50	R	52	82	r	72	114
DC3	13	19	3	33	51	S	53	83	s	73	115
DC4	14	20	4	34	52	T	54	84	t	74	116
NAK	15	21	5	35	53	U	55	85	u	75	117
SYN	16	22	6	36	54	V	56	86	v	76	118
ETB	17	23	7	37	55	W	57	87	w	77	119
CAN	18	24	8	38	56	X	58	88	x	78	120
EM	19	25	9	39	57	Y	59	89	y	79	121
SUB	1A	26	:	3A	58	Z	5A	90	z	7A	122
ESC	1B	27	;	3B	59	[	5B	91	{	7B	123
FS	1C	28	<	3C	60	\	5C	92	|	7C	124
GS	1D	29	=	3D	61	]	5D	93	}	7D	125
RS	1E	30	>	3E	62	^	5E	94	~	7E	126
US	1F	31	?	3F	63	_	5F	95	DEL	7F	127

APPENDIX E. USAGE ID DEFINITIONS

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

KEYBOARD/KEYPAD PAGE (0X07)

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

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

Note

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

Table A-1. Keyboard/Keypad

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

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

Usage ID (Dec)	Usage ID (Hex)	Usage Name	Ref: Typical AT-101 Position	PC-AT	Mac	UNIX	Boot
55	37	Keyboard. and >[4]	54	√	√	√	4/101/104
56	38	Keyboard / and ?	55	√	√	√	4/101/104
57	39	Keyboard Caps Lock[11]	30	√	√	√	4/101/104
58	3A	Keyboard F1	112	√	√	√	4/101/104
59	3B	Keyboard F2	113	√	√	√	4/101/104
60	3C	Keyboard F3	114	√	√	√	4/101/104
61	3D	Keyboard F4	115	√	√	√	4/101/104
62	3E	Keyboard F5	116	√	√	√	4/101/104
63	3F	Keyboard F6	117	√	√	√	4/101/104
64	40	Keyboard F7	118	√	√	√	4/101/104
65	41	Keyboard F8	119	√	√	√	4/101/104
66	42	Keyboard F9	120	√	√	√	4/101/104
67	43	Keyboard F10	121	√	√	√	4/101/104
68	44	Keyboard F11	122	√	√	√	101/104
69	45	Keyboard F12	123	√	√	√	101/104
70	46	Keyboard PrintScreen[1]	124	√	√	√	101/104
71	47	Keyboard Scroll Lock[11]	125	√	√	√	4/101/104
72	48	Keyboard Pause[1]	126	√	√	√	101/104
73	49	Keyboard Insert[1]	75	√	√	√	101/104
74	4A	Keyboard Home[1]	80	√	√	√	101/104
75	4B	Keyboard PageUp[1]	85	√	√	√	101/104
76	4C	Keyboard Delete Forward[1;14]	76	√	√	√	101/104
77	4D	Keyboard End[1]	81	√	√	√	101/104
78	4E	Keyboard PageDown[1]	86	√	√	√	101/104
79	4F	Keyboard RightArrow[1]	89	√	√	√	101/104
80	50	Keyboard LeftArrow[1]	79	√	√	√	101/104
81	51	Keyboard DownArrow[1]	84	√	√	√	101/104
82	52	Keyboard UpArrow[1]	83	√	√	√	101/104
83	53	Keypad Num Lock and Clear1[1]	90	√	√	√	101/104
84	54	Keypad /[1]	95	√	√	√	101/104
85	55	Keypad *	100	√	√	√	4/101/104
86	56	Keypad -	105	√	√	√	4/101/104
87	57	Keypad +	106	√	√	√	4/101/104
88	58	Keypad ENTER5	108	√	√	√	101/104
89	59	Keypad 1 and End	93	√	√	√	4/101/104
90	5A	Keypad 2 and Down Arrow	98	√	√	√	4/101/104
91	5B	Keypad 3 and PageDn	103	√	√	√	4/101/104
92	5C	Keypad 4 and Left Arrow	92	√	√	√	4/101/104
93	5D	Keypad 4 and Left Arrow	97	√	√	√	4/101/104
94	5E	Keypad 4 and Left Arrow	102	√	√	√	4/101/104

Usage ID (Dec)	Usage ID (Hex)	Usage Name	Ref: Typical AT-101 Position	PC-AT	Mac	UNIX	Boot
95	5F	Keypad 7 and Home	91	√	√	√	4/101/104
96	60	Keypad 8 and Up Arrow	96	√	√	√	4/101/104
97	61	Keypad 9 and PageUp	101	√	√	√	4/101/104
98	62	Keypad 0 and Insert	99	√	√	√	4/101/104
99	63	Keypad . and Delete	104	√	√	√	4/101/104
100	64	Keyboard Non-US \ and |[3;6]	45	√	√	√	4/101/104
101	65	Keyboard Application[10]	129	√		√	104
102	66	Keyboard Power[9] =			√	√	
103	67	Keypad =			√		
104	68	Keyboard F13	62		√		
105	69	Keyboard F14	63		√		
106	6A	Keyboard F15	64		√		
107	6B	Keyboard F16	65				
107	6C	Keyboard F17					
109	6D	Keyboard F18					
110	6E	Keyboard F19					
111	6F	Keyboard F20					
112	70	Keyboard F21					
113	71	Keyboard F22					
114	72	Keyboard F23					
115	73	Keyboard F24					
116	74	Keyboard Execute				√	
117	75	Keyboard Help				√	
118	76	Keyboard Menu				√	
119	77	Keyboard Select				√	
120	78	Keyboard Stop				√	
121	79	Keyboard Again				√	
122	7A	Keyboard Undo				√	
123	7B	Keyboard Cut				√	
124	7C	Keyboard Copy				√	
125	7D	Keyboard Paste				√	
126	7E	Keyboard Find				√	
127	7F	Keyboard Mute				√	
128	80	Keyboard Volume Up				√	
129	81	Keyboard Volume Down				√	
130	82	Keyboard Locking Caps Lock[12]				√	
131	83	Keyboard Locking Num Lock[12]				√	
132	84	Keyboard Locking Scroll Lock[12]				√	
133	85	Keypad Comma[27]	107				
134	86	Keypad Equal Sign[29]					

Usage ID (Dec)	Usage ID (Hex)	Usage Name	Ref: Typical AT-101 Position	PC-AT	Mac	UNIX	Boot
135	87	Keyboard International1[15-28]	56				
136	88	Keyboard International2[16]					
137	89	Keyboard International3[17]					
138	8A	Keyboard International4[18]					
139	8B	Keyboard International5[19]					
140	8C	Keyboard International6[20]					
141	8D	Keyboard International7[21]					
142	8E	Keyboard International8[22]					
143	8F	Keyboard International9[22]					
144	90	Keyboard Lang1[25]					
145	91	Keyboard Lang2[26]					
146	92	Keyboard Lang3[30]					
147	93	Keyboard Lang4[31]					
148	94	Keyboard Lang5[32]					
149	95	Keyboard Lang6[8]					
150	96	Keyboard Lang7[8]					
151	97	Keyboard Lang8[8]					
152	98	Keyboard Lang9[8]					
153	99	Keyboard Alternate Erase[7]					
154	9A	Keyboard Sys/Req Attention[1]					
155	9B	Keyboard Cancel					
156	9C	Keyboard Clear					
157	9D	Keyboard Prior					
158	9E	Keyboard Return					
159	9F	Keyboard Separator					
160	A0	Keyboard Out					
161	A1	Keyboard Oper					
162	A2	Keyboard Clear/Again					
163	A3	Keyboard Cr/Sel/Props					
164	A4	Keyboard Ex Sel					
165-175	A5-CF	Reserved					
176	B0	Keypad 00					
177	B1	Keypad 000					
178	B2	Thousands Separator[33]					
179	B3	Decimal Separator[33]					
180	B4	Currency Unit[34]					
181	B5	Currency Sub-unit[34]					
182	B6	Keypad (					
183	B7	Keypad)					
184	B8	Keypad {					

Usage ID (Dec)	Usage ID (Hex)	Usage Name	Ref: Typical AT-101 Position	PC-AT	Mac	UNIX	Boot
185	B9	Keypad}					
186	BA	Keypad Tab					
187	BB	Keypad Backspace					
188	BC	Keypad A					
189	BD	Keypad B					
190	BE	Keypad C					
191	BF	Keypad D					
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	√	√	√	

Usage ID (Dec)	Usage ID (Hex)	Usage Name	Ref: Typical AT-101 Position	PC-AT	Mac	UNIX	Boot
226	E2	Keyboard LeftA;t	60	√	√	√	
227	E3	Keyboard Left GUI[10;23]	127	√	√	√	
228	E4	Keyboard RightControl	64	√	√	√	
229	E5	Keyboard RightShift	57	√	√	√	
230	E6	Keyboard RightAlt	62	√	√	√	
231	E7	Keyboard Right GUI[10;24]	128	√	√	√	
232 – 65535	E8-FFFF	Reserved					

Footnotes

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

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

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

APPENDIX F. MODIFIER BYTE DEFINITIONS

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

The modifier byte is defined as follows:

Table B-1. Modifier Byte

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