USER MANUAL

SecureHead™ Encrypted Magnetic Read Head With TriMagIV ASIC

SPI Interface

80101502-002 Rev N
23 August 2022

ID TECH
10721 Walker Street, Cypress, CA 90630-4720
Tel: (714) 761-6368 Fax (714) 761-8880
www.idtechproducts.com

This document, as well as the software and hardware described in it, is furnished under license and may be used or copied online in accordance with the terms of such license. The content of this document is furnished for information use only, is subject to change without notice, and should not be construed as a commitment by ID TECH. While every effort has been made to ensure the accuracy of the information provided, ID TECH assumes no responsibility or liability for any unintentional errors or inaccuracies that may appear in this document. Except as permitted by such license, no part of this publication may be reproduced or transmitted by electronic, mechanical, recording, or otherwise, or translated into any language form without the express written consent of ID TECH.

Agency Approved
Specifications for subpart B of part 15 of FCC rule for a Class A computing device.

Limited Warranty
ID TECH warrants to the original purchaser for a period of 12 months from the date of invoice that this product is in good working order and free from defects in material and workmanship under normal use and service. ID TECH's obligation under this warranty is limited to, at its option, replacing, repairing, or giving credit for any product which has, within the warranty period, been returned to the factory of origin, transportation charges and insurance prepaid, and which is, after examination, disclosed to ID TECH's satisfaction to be thus defective. The expense of removal and reinstallation of any item or items of equipment is not included in this warranty. No person, firm, or corporation is authorized to assume for ID TECH any other liabilities in connection with the sales of any product. In no event shall ID TECH be liable for any special, incidental, or consequential damages to Purchaser or any third party caused by any defective item of equipment, whether that defect is warranted against or not. Purchaser's sole and exclusive remedy for defective equipment, which does not conform to the requirements of sales, is to have such equipment replaced or repaired by ID TECH. For limited warranty service during the warranty period, please contact ID TECH to obtain a Return Material Authorization (RMA) number & instructions for returning the product.

THIS WARRANTY IS IN LIEU OF ALL OTHER WARRANTIES OF MERCHANTABILITY OR FITNESS FOR PARTICULAR PURPOSE. THERE ARE NO OTHER WARRANTIES OR GUARANTEES, EXPRESS OR IMPLIED, OTHER THAN THOSE HEREIN STATED. THIS PRODUCT IS SOLD AS IS. IN NO EVENT SHALL ID TECH BE LIABLE FOR CLAIMS BASED UPON BREACH OF EXPRESS OR IMPLIED WARRANTY OF NEGLIGENCE OF ANY OTHER DAMAGES WHETHER DIRECT, IMMEDIATE, FORESEEABLE, CONSEQUENTIAL OR SPECIAL OR FOR ANY EXPENSE INCURRED BY REASON OF THE USE OR MISUSE, SALE OR FABRICATIONS OF PRODUCTS WHICH DO NOT CONFORM TO THE TERMS AND CONDITIONS OF THE CONTRACT.

 The information contained herein is provided to the user as a convenience. While every effort has been made to ensure accuracy, ID TECH is not responsible for damages that might occur because of errors or omissions, including any loss of profit or other commercial damage. The specifications described herein were current at the time of publication but are subject to change at any time without prior notice.

Revision History

Rev	Date	Description of Changes	By
A	10/15/2015	• Initial Release	JH
B	09/21/2016	• Added discussion of Samsung Pay decoding in 4.4.3. • Added firmware upgradability to Introduction.	JH KT
C	11/01/2016	• Added Firmware Upgrade appendix.	KT
D	02/17/2017	• Added bits 3-7 definition in notes3 – "Clear/Mask • Data Sent Status" in 4.14.8	JW
E	05/15/2017 05/22/2017	• Changed Appendix J (on firmware updating:. Flow diagrams removed. • Add note about power management and firmware update. • Add expanded discussion of firmware update to Appendix J.	KT
F	12/05/2019	• Updated section 4.2: Communication Timing • Corrected header and footer, removed section breaks and replaced them with page breaks, repaginated layout.	CB
G	07/15/2020	• Updated font/style • Updated pinout diagrams	CB
H	09/17/2020	• Corrected ACK response; correct ACK is 0x06.	CB
J	09/30/2020	• Updated sleep current to .01 mA • Added wakeup time information as a note under Specifications. • Encrypted Output for Decoded Data: Security-Related Function IDs o Updated EncryptionID default value to **1**. o Removed SessionID parameter.	CB
K	10/08/2020	• Corrected wake up times in Specifications notes.	CB
L	11/02/2020	• Updated Appendix I: Installation and Use Guide for Magnetic Heads • Updated power specifications; removed standby specs, added sleep specs • Removed content referring to Original Data Output Format • Removed content referring to Raw Data Output Format	CB
M	12/28/2020	• Removed commands related to Fixed Keys.	CB
N	08/23/2022	• Added note to Encryption Output Format Setting command regarded default settings and recommended settings.	CB

Table of Contents

1. INTRODUCTION 7
2. SPECIFICATIONS 7
2.1. Dimensions 8
2.2. Mounting Options 9
2.2.1. Wing spring mounting 9
2.3. Head Assembly Only 9
3. SPI OPERATION 10
3.1. SPI Data Transmission 10
3.2. Clock Polarity and Phase 10
3.3. Master Input, Slave Output (MISO) 11
3.4. Master Output, Slave Input (MOSI) 12
3.5. Data Available Output (DAV) 12
3.6. Chip Select 13
3.7. Voltage Input and Ground 14
3.8. Communication 14
4. CONFIGURATION 15
4.1. Command Structure 15
4.1.1. Commands sent to SecureHead 15
4.1.2. Response from SecureHead 15
4.2. Communication Timing 16
4.3. Default Settings 16
4.4. General Selections 17
4.5. Change to Default Settings 17
4.6. MSR Reading Settings 17
4.7. Decoding Method Settings 17
4.7.1. Samsung Pay Encoding/Decoding 18
4.8. Review Settings 18
4.9. Review Firmware Version 18
4.10. Review Serial Number 19
4.11. Message Formatting Selections (Only for Security Level 1 & 2) 19
4.11.1. Terminator Setting 19
4.11.2. Preamble Setting 19
4.11.3. Postamble Setting 19
4.11.4. Track n Prefix Setting 20
4.11.5. Track n Suffix Setting 20
4.12. Magnetic Track Selections (Only for Security Level 1 & 2) 20
4.12.1. Track_ Selection Settings: 20
4.12.2. Track Separator Selection 21
4.12.3. Start/End Sentinel and Track2 Account Number Only 21
4.13. Encryption Settings 21
4.13.1. Review KSN (DUKPT Key management only) 22
4.13.2. Review Security Level 22
4.13.3. Encrypt External Data Command 22
4.14. Encrypted Output for Decoded Data 23
4.14.1. Encrypt Functions 23
4.14.2. Security Related Function ID 23

4.15. Security Management 25
4.15.1. Level 0 ..... *25*
4.15.2. Level 1 ..... *25*
4.15.3. Level 2 ..... *25*
4.15.4. Level 3 ..... *25*
5. ENCRYPTION MANAGEMENT 26
5.1. Check Card Format 26
5.1.1. ISO/ABA (American Banking Association) Card Encoding method *26*
5.1.2. AAMVA (American Association of Motor Vehicle Administration) Card Encoding method *26*
6. OTHERS (CUSTOMER CARD) 26
6.1. MSR Data Masking 26
6.1.1. Masked Area *26*
6.2. Level 1 and 2 Data Output Format 27
6.2.1. Magnetic Track Basic Decoded Data Format *27*
6.2.2. Magnetic Track Basic Raw Data Format *27*
6.2.3. Definitions *28*
6.3. DUKPT Level 3 Data Output Enhanced Format 28
6.3.1. Data length byte *29*
6.3.2. Card Encode Type *30*
6.4. Track Status 30
6.5. Track Data Length 30
6.6. Clear/Masked Data sent status 31
6.6.1. Encrypted Hash Data sent status *31*
6.7. Track Masked Data 31
6.8. Track Encrypted Data 31
6.9. Track Hashed Data 32
6.9.1. Encryption Output Format Setting *32*
6.9.2. Encryption Option Setting (for enhanced encryption format only) *32*
Note: *32*
6.9.3. Hash Option Setting: *32*
6.10. Mask Option Setting 33
6.11. Note 1 : Card Encode Type 33
6.12. Note 2: Track1-3 status byte 33
6.13. Note 3: Clear/mask data sent status 34
6.14. Note 4: Encrypted/Hash data sent status 34
6.14.1. Fixed Key Management Data Output Enhanced Format *34*
7. APPENDIX A: DEFAULT SETTING TABLE 35
8. APPENDIX B: MAGNETIC STRIPE STANDARD FORMATS 36
8.1. ISO Credit Card Format 36
8.1.1. Track1 ..... *36*
8.1.2. Track2 ..... *36*
8.2. AAMVA Driver's License Format 36
8.2.1. Track1 ..... *36*
8.2.2. Track2 ..... *37*
8.2.3. Track3 ..... *37*
9. APPENDIX C: OTHER MODE CARD DATA OUTPUT 38
10. APPENDIX D: GUIDE TO ENCRYPTING AND DECRYPTING DATA 38

11. APPENDIX E: KEY MANAGEMENT FLOW CHART 39
12. APPENDIX F: EXAMPLE OF DECODED DATA DECRYPTION 40
Security Level 3 Decryption: Enhanced Encryption Format 40
13. APPENDIX G: EXAMPLE OF SPI MASTER CHIP CONTROLLING 43
14. APPENDIX H: INSTALLATION AND USE GUIDE FOR MAGNETIC HEADS 48
14.1. Track Locations 48
14.2. Reference Surface/Wear-Plate 49
14.3. Card Reader Rails/Slot and Magnetic Head Protrusion 50
15. APPENDIX I: FIRMWARE UPGRADE 52
15.1. Procedure 52
15.2. Basic steps: 52
15.3. Load new firmware 52
15.4. Example 53

1. Introduction

The SPI SecureHead™ magnetic stripe reader can read 1, 2, or 3 tracks of magnetic stripe information. When connected to the host, the SecureHead is completely compatible with SPI (Serial Peripheral Interface). The raw data or decoded data go to the host via SPI. Firmware can also be upgraded via SPI.

The SecureHead supports both unencrypted and encrypted data output. When encryption is not turned on, the decoded data can be formatted with preamble/postamble and terminator characters to match the format expected by the host.

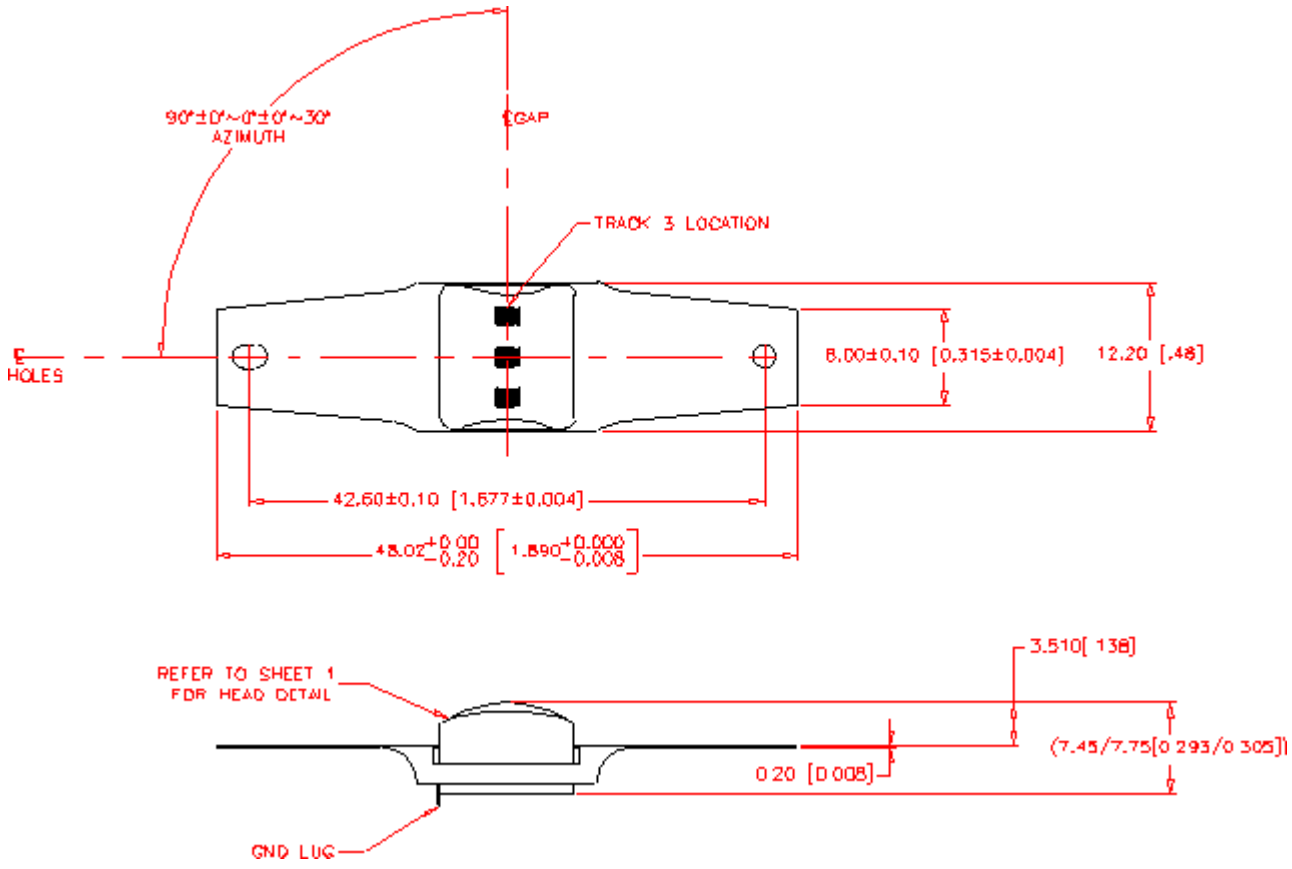
2. Specifications

General	
Card Speed	3 to 75 ips (7.6 to 190.5 cm/s)
Electrical	
Power Supply	3.0 to 3.6 VDC
I/O Voltage Range	2.7 to 3.6 VDC
Current	
Active Power Supply Current	5 mA
Sleep Power Supply Current	0.1 mA
Environmental	
ESD	+4kV discharge to head
Operating Temperature	0 °C to 55 C
Storage Temperature	-40 °C to 70 °C
Humidity	-10% to 90% non-condensing
Mechanical	
Weight	5.67 grams
Cable Length	125 +/- 6.4 mm

Note 1: During the analog components' wake up, a few capacitors are charged up, and the wake-up inrush current can go up to 40 mA for no more than 5 µsec.
Note 2: During the chip power up, the internal regulator can introduce 80 mA current for 50 µsec.
Note 3: ID TECH recommends that you incorporate the ability to separately control power to the SPI TrimagIV SecureHead. During the firmware update procedure, there is a short time (a few seconds) during which, if power is removed from the device, firmware loading can fail. The host software can cycle the power of SecureHead to get the unit back to bootload mode again, then the host needs to talk to the unit within ~500 msec and continue loading firmware. Please check Tech Note 015 for details (available for download on the ID TECH Knowledge Base.)
Note 4: The time for standard wake up is 1mS (50kHz sleep); the fast wake up time is 200uS at 4Mhz sleep.

For normal operation, we do not recommend turning off the power of the unit. Also, do not turn off the power within 2 seconds after receiving MSR data.

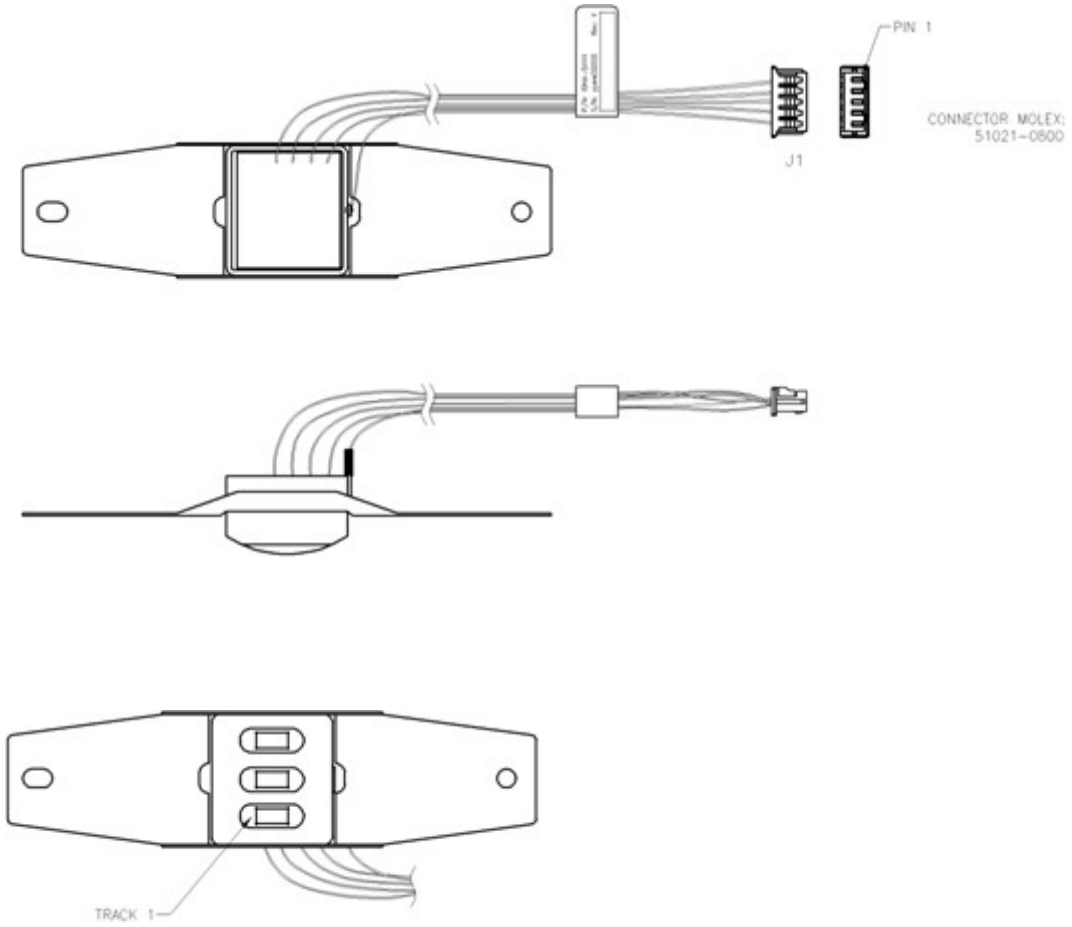
2.1. Dimensions



2.2. Mounting Options

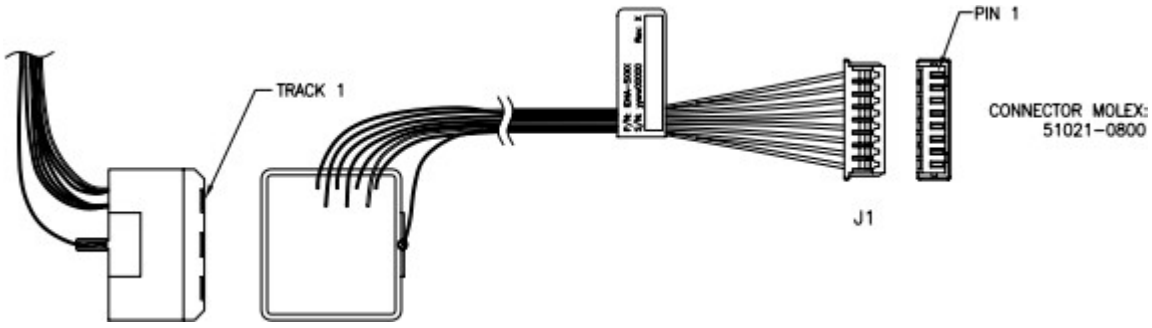
2.2.1. Wing spring mounting

This is the standard mounting option and can be used on most swipe readers. The protrusion of the head from the surface of the spring is 3.50 mm.



2.3. Head Assembly Only

This option is provided for special applications.



The mechanical interface is an eight-pin male Molex Connector 51021-0800 for option 1 and 2.

3. SPI Operation

This section describes SPI (Serial Peripheral Interface), the SPI bus interface timing, communication protocol, timeouts, and data output format. The following table shows the signals used in the SPI interface. Note that the connector is an eight-pin Molex 51021-0800.

PIN #	Signal	Description
1	SPCK	Serial Clock Input
2	MISO	Master Input, Slave Output
3	MOSI	Master Output, Slave Input
4	DAV	Data Available (output)
5	NCS	Chip Select, Active Low
6	VIN	Voltage Input
7	GND	Logic Ground
8	Head Case GND	Chassis Ground

3.1. SPI Data Transmission

A *serial peripheral interface* (SPI) is an interface that enables the serial exchange of data between two devices, one called a master and the other called a slave. The host (master) generates the clock signal (SPCK) to trigger data exchange on the SPI bus.

During each SPI clock cycle, data are transmitted in both directions at the same time (full duplex transmission):

- On the MOSI line, the master sends a bit and the slave reads it
- On the MISO line, the slave sends a bit and the master reads it

The SPI bus transmits data in 8-bit data groups, sending data one bit at a time, from MSB to LSB. An example of bit transmission for byte A and byte B (of two-byte quantity AB) would be
A(bit 7) A(bit 6) … A(bit 0) B(bit 7) B(bit 6) … B(bit 0).

3.2. Clock Polarity and Phase

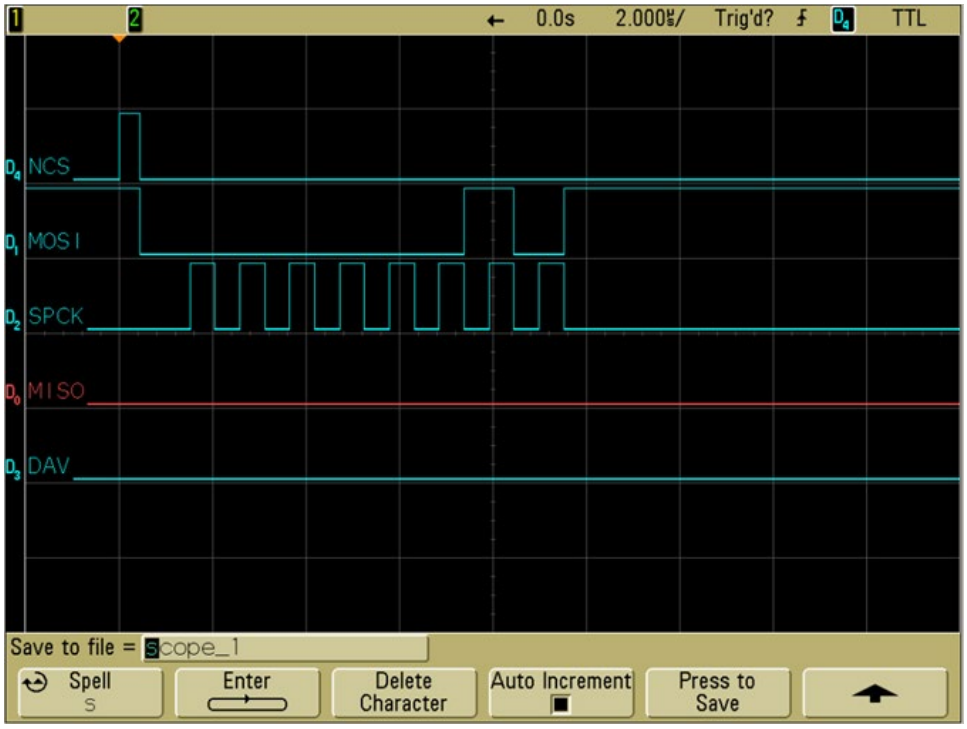
The clock polarity and phase have four different options with respect to the data. The serial clock input frequency can go up to 1M bps.

- When clock polarity = 0, the base value of the clock is 0
    - For clock phase = 0, data are read on the clock's rising edge (low->high transition) and data are changed on a falling edge (high->low transition).
    - For clock phase = 1, data are read on the clock's falling edge and data are changed on a rising edge.
    - When clock polarity = 1, the base value of the clock is 1

- For clock phase = 0, data are read on clock's falling edge and data are changed on a rising edge.
- For clock phase = 1, data are read on clock's rising edge and data are changed on a falling edge.

The signal is required to read card data from the device. The device default uses clock phase = 0 and clock polarity = 0. Custom defaults for device clock phase and polarity are available upon request.

The following picture shows an example of regular TM4 SPI firmware with clock polarity = 0 and clock phase = 0. The data are read on the rising edge of the clock and changed on the falling edge. On MOSI line, the host sends out data of 00000010, or 02h (0x02).



3.3. Master Input, Slave Output (MISO)

The MISO signal is the serial data output sent from for the device. It is also the data line that is received by the host. When the device is not active (Chip Select is high), the MISO becomes high impedance (disconnected). The MISO signal would be in an indeterminate state after the device is power-cycled or reset for a maximum of 1 second. This signal should be ignored during this time.

3.4. Master Output, Slave Input (MOSI)

The MOSI signal is the serial data input for the device and serial data output for the host. This signal is sent from the host (master) to the device (slave). The signal might not be required after some device parameters such as the device key has been set and saved. Set the signal to be high if it is not being used.

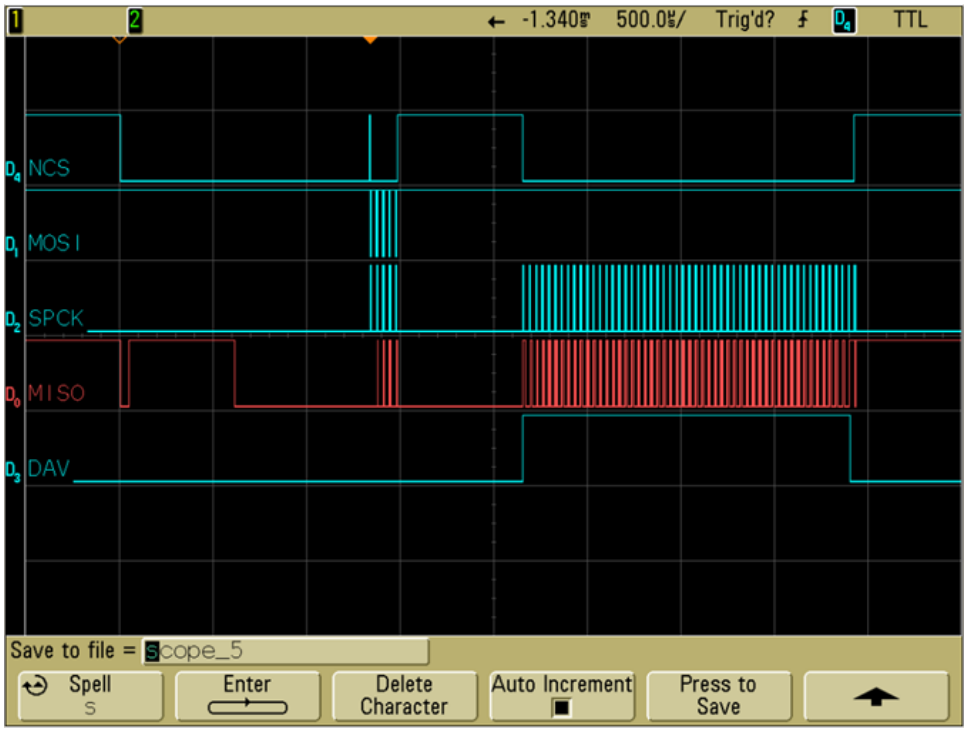
3.5. Data Available Output (DAV)

The DAV signal is low where there is no data to be transmitted. When the DAV signal is high, it indicates that there is data available for output. The host and then sends out the clock signal to read the data. After all the data is transmitted, the device sets the DAV signal low again.

The signal can be used for the host to determine if the device has data ready to transmit. However, the signal should be ignored right after (1 second maximum) the power cycle or a reset, as it would be in an indeterminate state.

In the case when the DAV signal is not used, the host will need to poll the device periodically to determine if it has data to transmit. The host needs to toggle SCL to get card data from MISO. The first non-IDLE byte indicates the start of valid card data. IDLE is FF. For more details, please refer to the communication protocol section of this document in the chapter on Configuration.

The following graph shows the command and response for Review Version command. The last signal shown in the graph is the DAV signal:



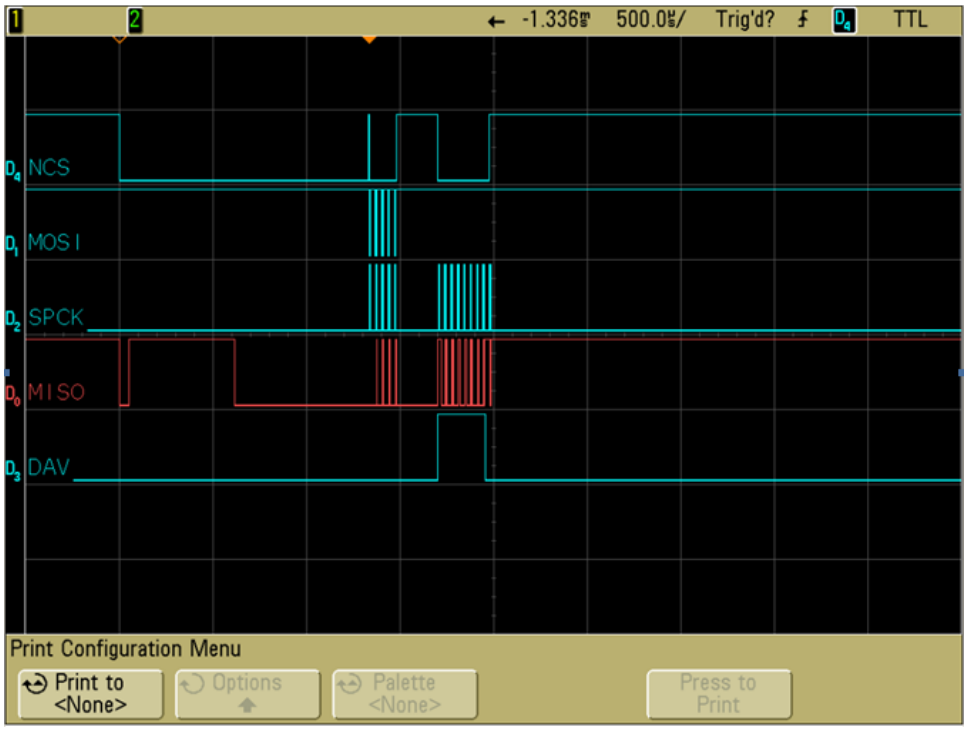
After the command is received and the response is ready, the DAV would be set too *high* for the host to receive response. After the response is received, the DAV would be *low*, indicating there is no more data to be transmitted.

After receiving a command, typically within less than 20ms, response is ready and DAV set to *high*. For some specific commands, the delay may be longer.

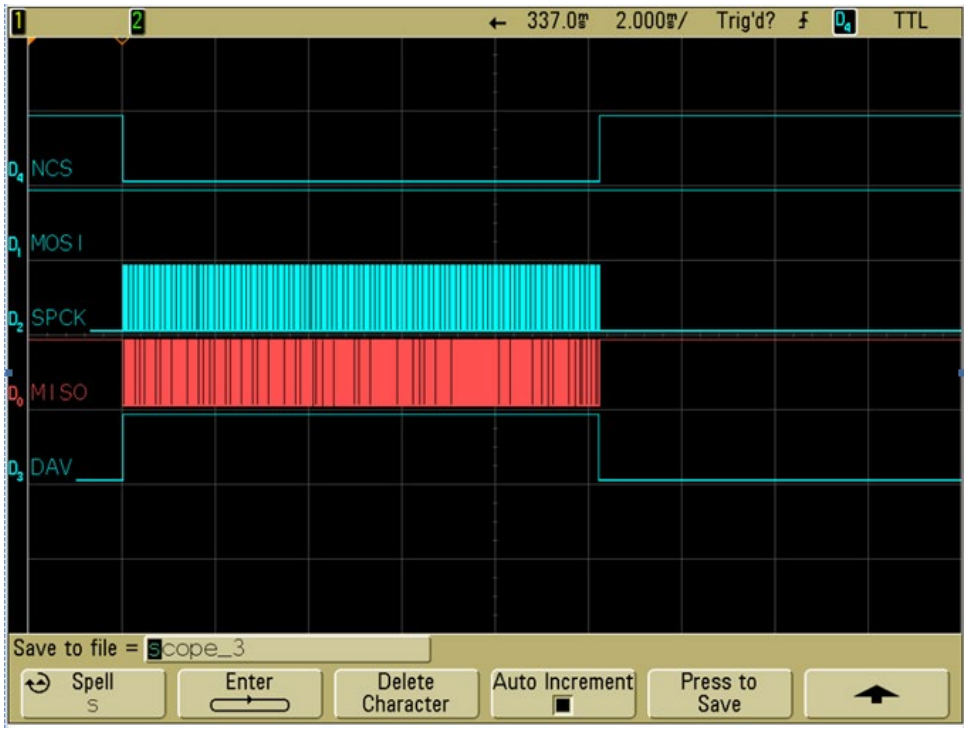
After the last byte of response is sent, the DAV is pulled low. If user polls DAV status to check whether there are data available, we suggest using 100μs polling interval and throw away any data when DAV is low.

3.6. Chip Select

SPI interface allows connecting several SPI devices while master selects each of them with NCS (Chip Select, Active Low). The device will only respond to SPCK and MOSI signals after an NCS is pulled low. For the first byte of each command sent to SecureHead, NCS needs to be pulled low for 1 millisecond before the clock line. Because SecureHead is always in deep sleep mode when in idle status, this 1 millisecond delay is required to allow SecureHead wake up from sleep mode.



When the user swipes a card, no delay is required. Following is the waveform for MSR output:



3.7. Voltage Input and Ground

The VIN signal is the power input for the device and has an operating range of 3.0 to 3.6 volts DC. The GND signal is logic ground. The head case GND signal is chassis ground, which is connected to the head case. For optimum ESD protection, this signal should be connected to earth ground.

3.8. Communication

When the host has a frame to send, it pulls the NCS line low, waits 1 millisecond, then clocks it out. When the device has a frame to send, it raises its data available (DAV) signal and waits for the host to pull the NCS line low, then clock in the frame. The host normally clocks out IDLE characters to clock in a frame from the device. Because the device typically loads its one transmit buffer with IDLE byte when it has nothing to transmit, the first byte clocked out from the device after the DAV signal is asserted could be IDLE instead of a valid byte. If this is the case, simply discard this byte.

To detect whether the device has a frame to send, the host can either monitor the DAV signal or, optionally, periodically clock in up to two bytes from the device to see if the device has sent a valid data. Up to two bytes should be clocked in instead of just one because the first byte could be an IDLE byte that was loaded into the device's transmit buffers before the device had anything to send. The host should look at each byte it clocks in to see if it is a valid byte. If a valid byte is found, then the subsequent bytes will contain the frame.

4. Configuration

The SecureHead reader must be appropriately configured to your application. Configuration settings enable the reader to work with the host system. Once programmed, these configuration settings are stored in the reader's non-volatile memory (so they are not affected by the cycling of power).
In TriMag IV, ACK is 0x06.

4.1. Command Structure

4.1.1. Commands sent to SecureHead

a. Setting Command:
<STX><S>[<FuncID><Len><FuncData>…]<ETX><CheckSum>

b. Read Status Command:
<STX><R><FuncID><ETX><CheckSum>

c. Special Function Command:
<STX>[<FuncID><Len><FuncData>…]<ETX><CheckSum>

4.1.2. Response from SecureHead

a. Setting Command

Host		SecureHead
	Setting Command	→
←	<ACK> if OK	
	or	
←	<NAK> if Error	

b. Read Status Command

Host		SecureHead
	Read Status Command	→
←	<ACK> and <Response> if OK	
	or	
←	<NAK> if Error	

c. Special Function Command

Host		SecureHead
	Special Function Command	→
←	<ACK> and <Response> if OK or	
←	<NAK> if Error	

Where:

<STX>	02h
<S>	Indicates setting commands. 53h
<R>	Indicates read status commands. 52h
<FuncID>	One-byte Function ID identifies the Specific function or settings affected.
<Len>	One-byte length count for the following data block<FuncData>
<FuncData>	Data block for the function
<ETX>	03h
<CheckSum>	Check Sum: The overall Modulo 2 (Exclusive OR) sum (from <STX> to <CheckSum>) should be zero.
<ACK>	06h
<NAK>	15h

4.2. Communication Timing

The power up time for TMIV SecureHead is 600ms. The typical delay for the reader to respond to a command is 20ms; the maximum delay for the reader to respond can be as much as 40ms. Caution must therefore be taken to maintain an appropriate delay between two commands.

4.3. Default Settings

The SecureHead reader is shipped from the factory with the default settings already programmed. In the following sections, the default settings are shown in **bold**.
For a table of default settings, see Appendix A.

4.4. General Selections

This group of configuration settings defines the basic operating parameters of SecureHead.

4.5. Change to Default Settings

Command: <STX><S><18h><ETX><CheckSum>

This command does not have any <FuncData>. It returns all settings for all groups to their default values.

4.6. MSR Reading Settings

Enable or Disable the SecureHead. If the reader is disabled, no data will be sent out to the host.

Command: <STX><S><1Ah><01h><MSR Reading Settings><ETX><CheckSum>

MSR Reading Settings:

- "0" MSR Reading Disabled
- "1" MSR Reading Enabled

4.7. Decoding Method Settings

The SecureHead can support four kinds of decoded directions.

Command: <STX><S><1Dh><01h><Decoding Method Settings><ETX><CheckSum>

Decoding Method Settings:

- "0": Raw Data Decoding in Both Directions, sent out in ID TECH mode.
- "1": Decoding in Both Directions. If the encryption feature is enabled, the key management method used is DUKPT.
- "2": Moving stripe along head in direction of encoding. If the encryption feature is enabled, the key management method used is DUKPT.
- "3": Moving stripe along head against direction of encoding. If the encryption feature is enabled, the key management method used is DUKPT.
- "4": Raw Data Decoding in Both Directions, send out in other mode. If the encryption feature is enabled, the key management method used is fixed key.

With the bi-directional method, the user can swipe the card in either direction and still read the data encoded on the magnetic stripe. Otherwise, the card can only be swiped in one specified direction to read the card. Raw Decoding just sends the card's magnetic data in groups of 4 bits per character. The head reads from the first byte of each track, starting from the most significant bit. The data start to be collected when the first 1 bit is detected. No checking is done except to verify that the track has, or does not have, magnetic data.

4.7.1. Samsung Pay Encoding/Decoding

Special track decoding considerations apply to Samsung Pay interactions. Samsung Pay/MST (LoopPay) sends out a magnetic signal to a magnetic head. So MCUs may receive identical magnetic signals on all tracks. However, Samsung Pay devices send out Track1 and Track2 data consecutively, making it possible to disambiguate the tracks.

If the reading device receives identical MSR data for multiple tracks, MSR processing will ignore Track2 and Track3 data if the card data is ISO 7-bit encoded, treating it as Track1 data . If the data are 5-bit encoded, it is received as Track2 data only.

If MSR receives single track data corresponding to ABA, IATA, or ISO 4909, but not in the expected track, the data will be ignored to avoid capturing track data as an incorrect type. The processor will not move data from one track to another.

4.8. Review Settings

Command: <STX><R><1Fh><ETX><CheckSum>
This command does not have any <FuncData>. It activates the review settings command. SecureHead sends back an <ACK> and <Response>.

<Response> format:
The current setting data block is a collection of many function-setting blocks <FuncSETBLOCK> as follows:

<STX><FuncSETBLOCK1>…<FuncSETBLOCKn><ETX><CheckSum>

Each function-setting block <FuncSETBLOCK> has the following format:
<FuncID><Len><FuncData>

Where:

- <FuncID> is one byte identifying the setting(s) for the function.
- <Len> is a one-byte length count for the following function-setting block <FuncData>
- <FuncData> is the current setting for this function. It has the same format as in the sending command for this function.
- <FuncSETBLOCK> are in the order of their Function ID<FuncID>

4.9. Review Firmware Version

Command: <STX><R><22h><ETX><CheckSum>

This command gets the device firmware version.

4.10. Review Serial Number

Command: <STX><R><4Eh><ETX><CheckSum>

This command gets the device serial number.

4.11. Message Formatting Selections (Only for Security Level 1 & 2)

4.11.1. Terminator Setting

Terminator characters are used to end a string of data in some applications.

Command: <STX><S><21h><01h><Terminator Settings><ETX><CheckSum>

<Terminator Settings>: Any one character, 00h is none; default is **CR** (0Dh).

4.11.2. Preamble Setting

Characters can be added to the beginning of a string of data. These can be special characters for identifying a specific reading station, to format a message header expected by the receiving host, or any other character string. Up to fifteen ASCII characters can be defined.

Command: <STX><S><D2h><Len><Preamble><ETX><CheckSum>

Where:

- <Len>= the number of bytes of preamble string
- <Preamble> = {string length}{string}

NOTE: String length is one byte, maximum fifteen <0Fh>.

4.11.3. Postamble Setting

The postamble serves the same purpose as the preamble, except it is added to the *end* of the data string, after any terminator characters.

Command: <STX><S><D3h><Len><Postamble><ETX><CheckSum>

Where:

- <Len> = the number of bytes of postamble string
- <Postamble> = {string length}{string}

NOTE: String length is one byte, maximum fifteen <0Fh>.

4.11.4. Track n Prefix Setting

Characters can be added to the beginning of a track data. These can be special characters to identify the specific track to the receiving host, or any other character string. Up to six ASCII characters can be defined.

Command: <STX><S><n><Len><Prefix><ETX><CheckSum>

Where:

- <n> = 34h for Track1; 35h for Track2 and 36h for Track3
- <Len> = the number of bytes of prefix string
- <Prefix> = {string length}{string}

NOTE: String length is one byte, maximum six.

4.11.5. Track n Suffix Setting

Characters can be added to the end of track data. These can be special characters to identify the specific track to the receiving host, or any other character string. Up to six ASCII characters can be defined.

Command: <STX><S><n><Len><Suffix><ETX><CheckSum>

Where:
<n> = 37h for Track1; 38h for Track2 and 39h for Track3
<Len> = the number of bytes of suffix string
<Suffix> = {string length}{string}

NOTE: String length is one byte, maximum six.

4.12. Magnetic Track Selections (Only for Security Level 1 & 2)

There are up to three tracks of encoded data on a magnetic stripe. This option selects the tracks that will be read and decoded.

Command: <STX><S><13h><01h><Track_Selection Settings><ETX><CheckSum>

4.12.1. Track_Selection Settings:

- "0" Any Track
- "1" Require Track1 Only
- "2" Require Track2 Only
- "3" Require Track1 & Track2
- "4" Require Track3 Only
- "5" Require Track1 & Track3
- "6" Require Track2 & Track3

- "7" Require All Three Tracks
- "8" Any Track1 & 2
- "9" Any Track2 & 3

Note: If any of the required multiple tracks fail to read for any reason, no data for any track will be sent.

4.12.2. Track Separator Selection

This option allows the user to select the character to be used to separate data decoded by a multiple-track reader.

Command: <STX><S><17h><01h><Track_Separator><ETX><CheckSum>

<Track_Separator> is one ASCII Character. The default value is **CR,** 0h means no track separator.

4.12.3. Start/End Sentinel and Track2 Account Number Only

The SecureHead can be set to either send, or not send, the Start/End sentinel, and to send either the Track2 account number only, or all the encoded data on Track2. (The Track2 account number setting does not affect the output of Track1 and Track3.)

Command: <STX><S><19h><01h><SendOption><ETX><CheckSum>

SendOption:

- "0" Do not send start/end sentinel and send all data on Track2
- "1" Send start/end sentinel and send all data on Track2
- "2" Don't send start/end sentinel and send account # on Track2
- "3" Send start/end sentinel and send account number on Track2

4.13. Encryption Settings

Enable or disable the SecureHead Encryption output in ID TECH protocol. If encryption is disabled, original data will be sent out to the host. If it enabled, encrypted data will be sent out to the host.

Command: <STX><S><4Ch><01h><Encryption Settings><ETX><CheckSum>

Encryption Settings:

- "0" Encryption Disabled
- "1" Enable TDES Encryption
- "2" Enable AES Encryption

4.13.1. Review KSN (DUKPT Key management only)

Command: <STX><R><51h><ETX><CheckSum>

This command gets the DUKPT key serial number and counter.

4.13.2. Review Security Level

Command: <STX><R><7Eh><ETX><CheckSum>

This command gets the current security level.

4.13.3. Encrypt External Data Command

This command encrypts the data passed to the SecureHead and sends back the encrypted data to the host. The command is valid when the security level is set to 3 or 4.

Command:
Host->Device:
Command: <STX><41h><Length<Data to Be Encrypted><ETX><CheckSum>

Where:
<Length> is the 2-byte length of <Data to Be Encrypted> in hex, represented as <Length_L> and <Length_H>

Device->Host:
Command: <ACK><STX><Length><Encrypted Data>[SessionID]<KSN><ETX><LRC> (success)
or <NAK> (fail)

Where:
<Length> is the 2-byte length of <Encrypted Data>[SessionID]<KSN> in hex, represented as <Length_L> and <Length_H>
[SessionID] is only used at security level 4; it is part of the encrypted data. No data in this field at security level 3.
<KSN> is a 10 bytes string, in the case of fix key management, use serial number plus two bytes null characters instead of KSN.

After each successful response, KSN will increment automatically.

4.14. Encrypted Output for Decoded Data

4.14.1. Encrypt Functions

When a card is swiped through the Reader, the track data will be encrypted via TDES (Triple Data Encryption Algorithm, aka, Triple DES) or AES (Advanced Encryption Standard) using Fixed key management or DUKPT (Derived Unique Key Per Transaction) key management. DUKPT key management uses a base derivation key to encrypt a key serial number that produces an initial encryption key (IPEK), which is injected into the Reader prior to deployment. After each transaction, the encryption key is modified per the DUKPT algorithm so that each transaction uses a unique key. Thus, the data will be encrypted with a different encryption key for each transaction, as a safeguard against replay attacks. DUKPT is described by ANSI X9.24-1:2009; for details, refer to that spec.

4.14.2. Security Related Function ID

Security Related Function IDs are listed below. Their functions are described in other sections.

Characters	Hex Value	Description
PrePANID	49	First N Digits in PAN which can be clear data
PostPANID	4A	Last M Digits in PAN which can be clear data
MaskCharID	4B	Character used to mask PAN
EncryptionID	4C	Security Algorithm
SecurityLevelID	7E	Security Level (Read Only)
Device Serial Number ID	4E	Device Serial Number (Can be write once. After that, can only be read)
DisplayExpirationDataID	50	Display expiration data as mask data or clear data
KSN and Counter ID	51	Review the Key Serial Number and Encryption Counter
Session ID	54	Set current Session ID
Key Management Type ID	58	Select Key Management Type

The example below lists possible settings of these new functions.

Characters	Default Setting	Description
PrePANID	04h	00h ~ 06h Allowed clear text from start of PAN Command format: 02 53 49 01 04 03 LRC
PostPANID	04h	00h ~ 04h Allowed clear text from end of PAN Command format: 02 53 4A 01 04 03 LRC
MaskCharID	'*'	20h ~ 7Eh Command format: 02 53 4B 01 3A 03 LRC
DisplayExpirationDataID	'0'	'0' Display expiration data as mask data '1' Display expiration data as clear data
EncryptionID	'1'	'0' Clear Text '1' Triple DES '2' AES Command format: 02 53 4C 01 31 03 LRC
SecurityLevelID	'1'	'0' ~ '3' Command format: 02 52 7E 03 LRC
Device Serial Number ID	00, 00, 00, 00, 00, 00, 00, 00, 00, 00	10 bytes number: Command format: Set Serial Number: 02 53 01 4E 09 08 37 36 35 34 33 32 31 30 03 LRC Get Serial Number: 02 52 4E 03 LRC
KSN and Counter ID	00, 00, 00, 00, 00, 00, 00, 00, 00, 00	This field includes the Initial Key Serial Number in the leftmost 59 bits and a value for the Encryption Counter in the right most 21 bits. Get DUKPT KSN and Counter: 02 52 51 03 LRC
Session ID	00, 00, 00, 00, 00, 00, 00, 00	This Session ID is an eight-byte string which contains hex data. This field is used by the host to uniquely identify the present transaction. Its primary purpose is to prevent replays. It is only used at Security Level 4 (not supported). After a card is read, the Session ID will be encrypted, along with the card data, supplied as part of the transaction message. The cleartext version of this will never be transmitted. New Session ID stays in effect until one of the following occurs: 1. Another Set Session ID command is received. 2. The reader is powered down. 3. The reader is put into Suspend mode.
Key Management Type ID	'1'	Fixed key management by default. '0': Fixed Key '1': DUKPT Key

4.15. Security Management

This reader is intended to be a secure reader. Security features include:

- Can include Device Serial Number
- Can encrypt Track1 and Track2 data for all bank cards
- Provides clear text confirmation data including card holder's name and a portion of the PAN as part of the Masked Track Data
- Optional display of expiration data
- Security Level is settable

The reader features configurable security settings. Before encryption can be enabled, Key Serial Number (KSN) and Base Derivation Key (BDK) must be loaded; then encrypted transactions can take place. The keys must be injected by certified key injection facility (such as ID TECH). Contact ID TECH for more information about key injection services.

Four security levels are available when using DUKPT key management:

4.15.1. Level 0

Security Level 0 is a special case where all DUKPT keys have been used and is set automatically when it runs out of DUKPT keys. The supply of DUKPT keys is effectively 1 million, meaning that a new key can be generated, per swipe, for up to a million card swipes. After this limit has been reached, key injection will need to occur again before any more transactions can be done.

4.15.2. Level 1

By default, readers from the factory are configured to have this security level. There is no encryption process, no key serial number transmitted with decoded data. The reader functions as a non-encrypting reader and the decoded track data is sent out in default mode.

4.15.3. Level 2

Key Serial Number and Base Derivation Key have been injected but the encryption process is not yet activated. The reader will send out decoded track data in default format. Setting the encryption type to TDES and AES will change the reader to security level 3.

4.15.4. Level 3

Both Key Serial Number and Base Derivation Keys are injected and encryption mode is turned on. For payment cards, both encrypted data and masked cleartext data are sent out. (Users can select the data masking of the PAN area; the encrypted data format cannot be modified.) You can choose whether to send hashed data and whether to reveal the card expiration date. When encryption is turned on, Level 3 is the default security level.

5. Encryption Management

The Encrypted swipe read supports TDES and AES encryption standards for data encryption. Encryption can be turned on via a command. TDES is the default.

If the reader is at or above security Level 3, for the encrypted fields, the original data is encrypted using the TDES/AES CBC mode with an Initialization Vector of all binary zeroes and the Encryption Key associated with the current DUKPT KSN.

5.1. Check Card Format

5.1.1. ISO/ABA (American Banking Association) Card Encoding method

Track1 is 7 bits encoding.
Track1 is 7 bits encoding.
Track2 is 5 bits encoding.
Track3 is 5 bits encoding.
Track1 is 7 bits encoding.
Track2 is 5 bits encoding.
Track2 is 5 bits encoding.

Additional check
Track1 second byte is 'B'.
There is only one '=' in Track2 and the position of '=' is between 13th ~ 20th character.
Total length of Track2 should above 21 characters.

5.1.2. AAMVA (American Association of Motor Vehicle Administration) Card Encoding method

Track1 is 7 bits encoding.
Track2 is 5 bits encoding.
Track3 is 7 bits encoding.

6. Others (Customer card)

6.1. MSR Data Masking

For cards that need to be encrypted, a combination of encrypted data and masked clear text data are sent.

6.1.1. Masked Area

The data format of each masked track is ASCII.
The clear data include start and end sentinels, separators, first N, last M digits of the PAN, card holder name (for Track1).
The rest of the characters should be masked using mask character.

Set PrePANClrData (N), PostPANClrData (M), MaskChar (Mask Character)

N and M are configurable and default to 4 first and 4 last digits. They follow the current PCI constraints requirements (N 6, M 4 maximum).
Mask character default value is '*'.

Set PrePANClrDataID (N), parameter range 00h ~ 06h, default value 04h

Set PostPANClrDataID (M), parameter range 00h ~ 04h, default value 04h

MaskCharID (Mask Character), parameter range 20h ~ 7Eh, default value 2Ah

DisplayExpirationDataID, parameter range '0'~'1', default value '0'

6.2. Level 1 and 2 Data Output Format

6.2.1. Magnetic Track Basic Decoded Data Format

Track1: <SS1><T1 Data><ES><Track Separator>
Track2: <SS2><T2 Data><ES><Track Separator>
Track3: <SS3><T3 Data><ES><Terminator>

Where:
SS1 (start sentinel Track1) = %
SS2 (start sentinel Track2) = ;
SS3 (start sentinel Track3) = ; for ISO, % for AAMVA
ES (end sentinel all tracks) = ?
Track Separator = Carriage Return
Terminator = Carriage Return
Language: US English

6.2.2. Magnetic Track Basic Raw Data Format

Track1: <01><T1 Raw Data><CR>
Track2: <02><T2 Raw Data><CR>
Track3: <03><T3 Raw Data><CR>

Where: The length of T1 Raw Data, T2 Raw Data, T3 Raw Data is 0x60 for each field. Pad with 0 if the original data length does not reach 0x60.

Language: US English

6.2.3. Definitions

Start or End Sentinel: Characters in encoding format which come before the first data character (start) and after the last data character (end), indicating the beginning and end, respectively, of data.

Track Separator: A designated character that separates data tracks.

Terminator: A designated character that comes at the end of the last track of data, to separate card reads.

6.3. DUKPT Level 3 Data Output Enhanced Format

For ISO cards, both masked clear and encrypted data are sent; no unmasked clear data will be sent. For other cards, only clear data is sent.

This mode is used when all tracks must be encrypted, or encrypted OPOS support is required, or when the tracks must be encrypted separately or when cards other than type 0 (ABA bank cards) must be encrypted or when Track3 must be encrypted. This format is the standard encryption format, but not yet the default encryption format.

Card data is sent out in the following format
<STX><LenL><LenH><Card Data><CheckLRC><CheckSum><ETX>

Value	Description
0	STX
1	Data Length low byte
2	Data Length high byte
3	Card Encode Type[1]
4	Track1-3 Status[2]
5	Track1 data length
6	Track2 data length
7	Track3 data length
8	Clear/masked data sent status [3]
9	Encrypted/Hash data sent status [4]
10	Track1 clear/mask data Track2 clear/mask data Track3 clear/mask data Track1 encrypted data Track2 encrypted data Track3 encrypted data Session ID info for Level 4 (Level 4 not available) Track1 hashed (20 bytes each) (if encrypted and hash Track1 allowed) Track2 hashed (20 bytes each) (if encrypted and hash Track2 allowed)

Track3 hashed (20 bytes each) (if encrypted and hash Track3 allowed)
KSN (10 bytes)

CheckLRC CheckSum ETX

Where:

- <STX> = 02h
- <ETX> = 03h

Description (see also Appendix F for a real-world example):

6.3.1. Data length byte

LenL – Overall length of data, low bits LenH – Overall length of data, high bits

6.3.2. Card Encode Type

Value	Encode Type Description
80	ISO 7813/ISO 4909/ABA format
81	AAMVA format
83	Other
84	Raw; un-decoded format All tracks are encrypted and no mask data is sent. No track indicator '01', '02' or '03' in front of each track.
85	JIS II; Only supported in some products
86	JIS I; Only supported in some products
87	JIS II; SecureKey and Secure MIR
91	Contactless Visa (Kernel 1)
92	Contactless MasterCard
93	Contactless Visa (Kernel 3)
94	Contactless American Express
95	Contactless JCB
96	Contactless Discover
97	Contactless UnionPay
90	Contactless Others
C0	Manual data entry enhanced mode (similar to ABA Track2)

6.4. Track Status

MSR sampling and decode status

MB							LB
B7	B6	B5	B4	B3	B2	B1	B0

B0 1: Track1 decode success (0: Track1 decode fail)
B1 1: Track2 decode success (0: Track2 decode fail)
B2 1: Track3 decode success (0: Track3 decode fail)
B3 1: Track1 sampling data exists (0: Track1 sampling data does not exist)
B4 1: Track2 sampling data exists (0: Track2 sampling data does not exist)
B5 1: Track3 sampling data exists (0: Track3 sampling data does not exist)
B6 0: reserved for future use
B7 0: reserved for future use

6.5. Track Data Length

This one-byte value indicates the number of bytes in the respective track masked data field. For ISO 7813 and ISO 4909 compliant Financial Transaction Cards:
Track1 maximum length is 79 alphanumeric characters. Track2 maximum length is 40 numeric digits. Track3 maximum length is 107 numeric digits.

6.6. Clear/Masked Data sent status

Bit 0 1: Track1 clear/mask data present
Bit 1 1: Track2 clear/mask data present
Bit 2 1: Track3 clear/mask data present
Bit 3 1: fixed key
0: DUKPT Key Management
Bit 4 0: TDES
1: AES
Bit 5 0: No requirement to use IC
1: means chip present on card (2 or 6 in Service Code)
Bit 6 1: Pin Encryption Key
0: Data Encryption Key Refer ANSI X9.24 2009 Page 56 for details.
Bit7 1: Serial # present
0: not present

6.6.1. Encrypted Hash Data sent status

Bit 0 1: Track1 encrypted data present
Bit 1 1: Track2 encrypted data present
Bit 2 1: Track3 encrypted data present
Bit 3 1: Track1 hash data present
Bit 4 1: Track2 hash data present
Bit 5 1: Track3 hash data present
Bit 6 1: Session ID present
Bit 7 1: KSN present

6.7. Track Masked Data

Track data masked with the MaskCharID (default is '*'). The first PrePANID (up to 6 for BIN, default is 4) and last PostPANID (up to 4, default is 4) characters can be in the clear (unencrypted).

6.8. Track Encrypted Data

This field is the encrypted Track data, using either TDES-CBC or AES-CBC with initial vector of 0. If the original data is not a multiple of 8 bytes for TDES or a multiple of 16 bytes for AES, the reader right pads the data with 0.

The key management scheme is DUKPT or Fixed key. For DUKPT, the key used for encrypting data is called the Data Key. Data Key is generated by first taking the DUKPT Derived Key exclusive or'ed with 0000000000FF0000 to get the resulting intermediate variant key. The left side of the intermediate variant key is then TDES encrypted with the entire 16-byte variant as the key.

After the same steps are performed for the right side of the key, combine the two key parts to create the Data Key.

6.9. Track Hashed Data

SecureHead reader uses SHA-1 to generate hashed data for both Track1, Track2 and Track3 unencrypted data. It is 20 bytes long for each track. This is provided with two purposes in mind: One is for the host to ensure data integrity by comparing this field with a SHA-1 hash of the decrypted Track data, prevent unexpected noise in data transmission. The other purpose is to enable the host to store a token of card data for future use without keeping the sensitive card holder data. This token may be used for comparison with the stored hash data to determine if they are from the same card.

6.9.1. Encryption Output Format Setting

Command: 53 85 01 <Encryption Format>
Encryption Format:
'0': No longer supported
'1': Enhanced Encryption Format

Note: Most devices use Enhanced Encryption Format, but some legacy devices require option **0.** Use the **R** command to check a device's configuration; ID TECH advises device administrators to set their device to use Enhanced Encryption Format.

6.9.2. Encryption Option Setting (for enhanced encryption format only)

Command: 53 84 01 <Encryption Option>

Encryption Option: (default 08h)

- **bit 0:** 1: Track1 force encrypt
- **bit 1** 1: Track2 force encrypt
- **bit 2** 1: Track3 force encrypt
- **bit 3** 1: Track3 force encrypt when card type is 0

Note:

1. When force encrypt is set, this track will always be encrypted, regardless of card type. No clear/mask text will be sent.
2. If and only if in enhanced encryption format, each track is encrypted separately. Encrypted data length will round up to 8 or 16 bytes.

6.9.3. Hash Option Setting:

Command: 53 5C 01 <Hash Option>

Hash Option: ('0' – **'7'**)

- **bit0** 1: track1 hash will be sent if data is encrypted
- **bit1** 1: track2 hash will be sent if data is encrypted
- **bit2** 1: track3 hash will be sent if data is encrypted

6.10. Mask Option Setting

(for enhanced encryption format only) Command: 53 86 01 <Mask Option>
Mask Option: (**Default: 0x07**)
bit0: 1: Track1 mask data allow to send when encrypted
bit1: 1: Track2 mask data allow to send when encrypted
bit2: 1: Track3 mask data allow to send when encrypted

When mask option bit is set, if data is encrypted (but not forced encrypted), the mask data will be sent.

If mask option is not set, the mask data will not be sent under the same condition.

6.11. Note 1 : Card Encode Type

Card Type will be 8x for enhanced encryption format.

Value	Encode Type Description
00h / 80h	ISO/ABA format
01h / 81h	AAMVA format
03h / 83h	Other
04h / 84h	Raw; un-decoded format

Track indicator '01','02' and '03' will still exist for non-encrypted mode.

6.12. Note 2: Track1-3 status byte

Field 4:

Bit 0	1: Track1 decoded data present
Bit 1	1: Track2 decoded data present
Bit 2	1: Track3 decoded data present
Bit 3	1: Track1 sampling data present
Bit 4	1: Track2 sampling data present
Bit 5	1: Track3 sampling data present
Bit 6	1: Field 10 "optional bytes length" exists (0: No Field 10)

6.13. Note 3: Clear/mask data sent status

Field 8 (Clear/mask data sent status) and field 9 (Encrypted/Hash data sent status) will only be sent out in enhanced encryption format.

Field 8: Clear/masked data sent status byte:

Bit 0 1: Track1 clear/mask data present
Bit 1 1: Track2 clear/mask data present
Bit 2 1: Track3 clear/mask data present
Bit 3 1 if fixed key
0 DUKPT Key Management
Bit 4 0: TDES
1: AES
Bit 5 0: No requirement to use IC (1st digit in Service Code is different from 2 or 6)
1: Use IC where feasible (1st digit in Service Code is 2 or 6)
Bit 6 1: Pin Encryption Key
0: Data Encryption Key Refer ANSI X9.24 2009 Page 56 for details.
Bit 7 1: Serial # present
0: not present

6.14. Note 4: Encrypted/Hash data sent status

Field 9: Encrypted data sent status

Bit 0 1: Track1 encrypted data present
Bit 1 1: Track2 encrypted data present
Bit 2 1: Track3 encrypted data present
Bit 3 1: Track1 hash data present
Bit 4 1: Track2 hash data present
Bit 5 1: Track3 hash data present
Bit 6 1: session ID present
Bit 7 1: KSN present

6.14.1. Fixed Key Management Data Output Enhanced Format

Same as 4.14.10 DUKPT Level 3 Data Output Enhanced Format, only change <KSN> to <device serial number> plus two NULL bytes.

7. Appendix A: Default Setting Table

MSR Reading	Enabled
Decoding Method	Both Swiping Direction Decode mode
Track Separator Settings	CR
Terminator Settings	CR
Preamble Settings	None
Postamble Settings	None
Track Selected Settings	Any Track
Sentinel and T2 Account No	Send Sentinels and all T2 data
Data Edit Setting	Disabled
Track Prefix	None
Track Suffix	None

8. Appendix B: Magnetic Stripe Standard Formats

8.1. ISO Credit Card Format

ISO stands for International Standards Organization

8.1.1. Track1

Field ID Character	Contents	Length
a	Start Sentinel	1
b	Format Code "B"	1
c	Account Number	12 or 19
d	Separator "^"	1
e	Cardholder Name	Variable
f	Separator "^"	1
g	Expiration date 4	
h	Optional Discretionary data	Variable
i	End Sentinel	1
j	Linear Redundancy Check (LRC) Character	1

8.1.2. Track2

Field ID Character	Contents	Length
a	Start Sentinel	1
b	Account Number	12 or 19
c	Separator "="	1
d	Expiration date "YYMM"	4
e	Optional discretionary data	Variable
f	End Sentinel	1
g	Linear Redundancy Check (LRC) Character	1

8.2. AAMVA Driver's License Format

8.2.1. Track1

Field ID Character	Contents	Length
a	Start Sentinel	1
b	State or Province	2
c	City	13
d	Name	35
e	Address	29
f	End Sentinel	1
g	Linear Redundancy Check (LRC) Character	1

8.2.2. Track2

Field ID Character	Contents	Length
a	Start Sentinel	1
b	ANSI User Code	1
c	ANSI User ID	5
d	Jurisdiction ID/DL	14
e	Expiration date	4
f	Birth Date	8
g	Remainder of Jurisdiction	
i	ID/DL	5
h	End Sentinel	1
i	Linear Redundancy Check (LRC) Character	1

8.2.3. Track3

Field ID Character	Contents	Length
a	Start Sentinel	1
b	Template Version #	1
c	Security Version #	1
d	Postal Code	11
e	Class	2
f	Restrictions	10
g	Endorsements	4
h	Sex	1
i	Height	3
j	Weight	3
k	Hair Color	3
l	Eye Color	3
m	ID #	10
n	Reserved Space	16
o	Error Correction	6
p	Security	5
q	End Sentinel	1
r	Linear Redundancy Check (LRC) Character	1

9. APPENDIX C: Other Mode Card Data Output

There is an optional data output format supported by SecureHead to be compatible with specific software requirements.

<01h> <01h> <1Ah> <02h> <00h> <Left 8 bytes Device Serial Number> <6 Byte Random data> <30h> <31h> <264 bytes of Sampling data>.

10. APPENDIX D: Guide To Encrypting And Decrypting Data

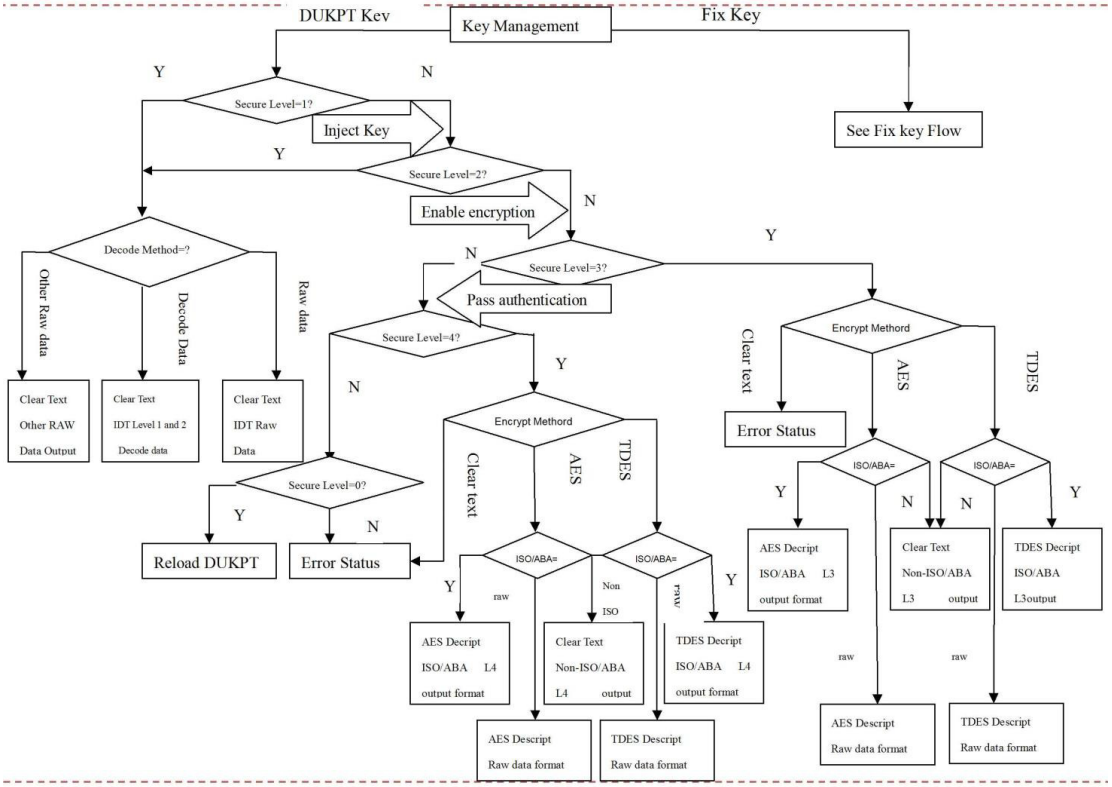
The encryption mode used by SecureHead is called Cipher-block Chaining (CBC). With this method, each block of data is XOR'ed with the previous data block before being encrypted. The encryption of each block depends on all the previous blocks. As a result, each encrypted data block would need to be decrypted sequentially.

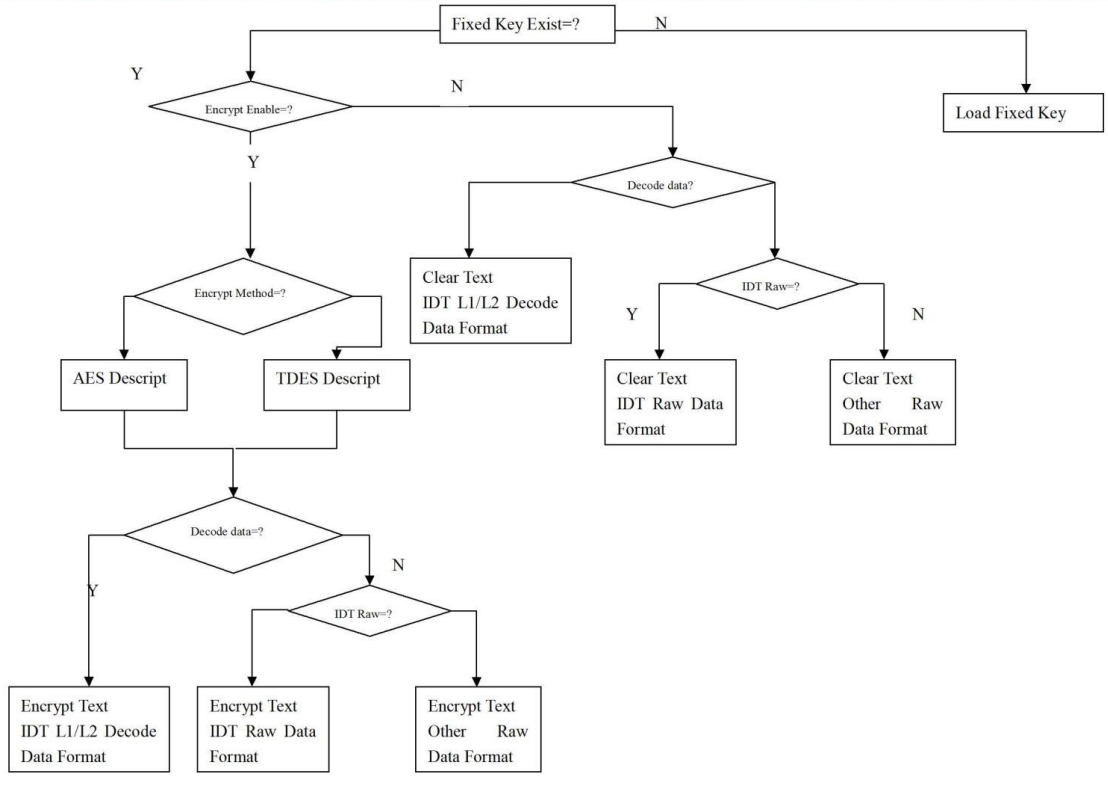
To encrypt the data, first generate 8 bytes of 0x00 to use as an initialization vector which is XOR'ed with the first data block before it is encrypted. Then the data is encrypted with the device key using TDES algorithm. The result is again XOR'ed with the next 8-byte data block before it is encrypted. The process repeats until all the data blocks have been encrypted.

The host can decrypt the cipher text from the beginning of the block when the data is received. However, it must keep track of both the encrypted and clear text data. Or alternatively, the data can be decrypted backward form that last data block to the first, so that the decrypted data can replace the original data as the decryption is in process.

To decrypt the data using reverse method, first decrypt the last 8-byte of data using TDES decryption. Then perform an XOR operation with result and the preceding data block to get the last data block in clear text. Continue to decrypt the next previous block with the same method till it reaches the first block. For the first data block, the XOR operation can be skipped because it is XOR'ing with 00h bytes.

11. APPENDIX E: Key Management Flow Chart





12. APPENDIX F: Example of Decoded Data Decryption

Key for all examples is 0123456789ABCDEFFEDCBA9876543210

Security Level 3 Decryption: Enhanced Encryption Format

Example of decryption of a three track ABA card with the enhanced encryption format. SecureHead Reader with default settings except enhanced encryption structure format.

Enhanced encryption Format (this can be recognized because the high bit of the fourth byte underlined (80) is 1.

```
02980180 3F48236B03BF252A343236362A2A2A2A2A2A2A393939395E42555348204A
522F47454F52474520572E4D525E2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A
2A2A2A2A2A2A2A2A2A2A3F2A3B343236362A2A2A2A2A2A2A393939393D2A2A2A2A2A
2A2A2A2A2A2A2A2A2A2A3F2ADA7F2A52BD3F6DD8B96C50FC39C7E6AF22F06ED1F033BE0
FB23D6BD33DC5A1F808512F7AE18D47A60CC3F4559B1B093563BE7E07459072ABF8FAAB
5338C6CC8815FF87797AE3A7BEAB3B10A3FBC230FBFB941FAC9E82649981AE79F263215
6E775A06AEDAFAF6F0A184318C5209E55AD44A9CCF6A78AC240F791B63284E15B40191
02BA6C505814B585816CA3C2D2F42A99B1B9773EF1B116E005B7CD8681860D174E6AD3
16A0ECDBC687115FC89360AEE7E430140A7B791589CCAADB6D6872B78433C3A25DA9DD
AE83F12FEFAB530CE405B701131D2FBAAD970248A456000933418AC88F65E1DB7ED4D10
973F99DFC8463FF6DF113B6226C4898A9D355057ECAF11A5598F02CA31688861C157C1C
E2E0F72CE0F3BB598A614EAABB16299490119000000000206E203
```

STX, Length(LSB, MSB), card type, track status, length Track1, length Track2, length Track3 02 9801 80 3F 48-23-6B 03BF

The above broken down and interpreted
02: STX character
98: low byte of total length
01: high byte of total length
80: card type byte (interpretation new format ABA card)
3F: 3 tracks of data all good
48: length of Track1
23: length of Track2
6B: length of Track3
03: tracks 1 and 2 have masked/clear data
BF: bit 7=1: KSN included
Bit 6=0: no Session ID included so not level 4 encryption
Bit 5=1: Track3 hash data present
Bit 4=1: Track2 hash data present
Bit 3-1: Track1 hash data present
Bit 2=1: Track3 encrypted data present

Bit 1=1: Track2 encrypted data present
Bit 0=1: Track1 encrypted data present

Track1 data masked (length 0x48)
252A343236362A2A2A2A2A2A2A393939395E42555348204A522F47454F5247452057
2E4D525E2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A
3F2A

Track1 masked data in ASCII
%*4266********9999^BUSH JR/GEORGE
W.MR^*******************************?*

Track2 data in hex masked (length 0x23)
3B343236362A2A2A2A2A2A2A393939393D2A2A2A2A2A2A2A2A2A2A2A2A2A2A3F2A

Track2 masked data in ASCII
;4266********9999=***************?*

In this example there is no Track3 data either clear or masked (encrypted and hashed data is below)

Track1 encrypted length 0x48 rounded up to 8 bytes = 0x48 (72 decimal)
DA7F2A52BD3F6DD8B96C50FC39C7E6AF22F06ED1F033BE0FB23D6BD33DC5A1F8
08512F7AE18D47A60CC3F4559B1B093563BE7E07459072ABF8FAAB5338C6CC88
15FF87797AE3A7BE

Track2 encrypted length 0x32 rounded up to 8 bytes =0x38 (56 decimal)
AB3B10A3FBC230FBFB941FAC9E82649981AE79F2632156E775A06AEDAFAF6F0A
184318C5209E55AD

Track3 encrypted length 0x6B rounded up to 8 bytes =0x70 (64 decimal)
44A9CCF6A78AC240F791B63284E15B4019102BA6C505814B585816CA3C2D2F42
A99B1B9773EF1B116E005B7CD8681860D174E6AD316A0ECDBC687115FC89360A
EE7E430140A7B791589CCAADB6D6872B78433C3A25DA9DDAE83F12FEFAB530CE
405B701131D2FBAAD970248A45600093

Track1 data hashed length 20 bytes 3418AC88F65E1DB7ED4D10973F99DFC8463FF6DF

Track2 data hashed length 20 bytes 113B6226C4898A9D355057ECAF11A5598F02CA31
Track3 data hashed length 20 bytes 688861C157C1CE2E0F72CE0F3BB598A614EAABB1

KSN length 10 bytes 62994901190000000002

LCR, check sum and ETX 06E203

Clear/Masked Data in ASCII:

Track1: %*4266********9999^BUSH JR/GEORGE W.MR^*******************************?* Track2: ;4266********9999=***************?*

Key Value: 1A 99 4C 3E 09 D9 AC EF 3E A9 BD 43 81 EF A3 34 KSN: 62 99 49 01 19 00 00 00 00 02

Decrypted Data:

Track1 decrypted

%B4266841088889999^BUSH JR/GEORGE W.MR^0809101100001100000000046000000?!

Track2 decrypted

;4266841088889999=080910110000046?0

Track3 decrypted

;333333333376767607070776767633333333337676760707077676763333333333767676070707767676333333333376767607077?2

Track1 decrypted data in hex including padding zeros (but there are no pad bytes here)

```
2542343236363834313038383838393939395E42555348204A522F47454F5247452057
2E4D525E30383039313031313030303031313030303030303030303436303030303030
3F21
```

Track2 decrypted data in hex including padding zeros

```
3B343236363834313038383838393939393D30383039313031313030303030343633F30
0000000000
```

Track3 decrypted data in hex including padding zeros

```
3B3333333333333333333337363736373630373037303737363736373633333333333333
3333333333373637363736303730373037373637363736333333333333333333373637
3637363037303730373736373637363333333333333333333337363736373630373037
3F320000000000
```

13. APPENDIX G: Example oF SPI Master Chip Controlling

```
/*H**************************************************************************
  * NAME:       spi_drv.h
  * Copyright (c) 2003 ID TECH.
  * RELEASE:    cc03-demo-spi-0_0_1
* REVISION:     1.1.1.1
  * PURPOSE:
  * spi lib header file
*****************************************************************************/


#ifndef
_spi_DRV
_H_
#define
_spi_DRV
_H_

/*_____I N C L U D E S_______*/

/*_____D E F I N I T I O N___*/
// Pin define
 #define _DAV_IN                P3_4                                    // SPI
 chip has data ready
 #define _SPI_SS                P1_1                                    // SPI
 chip select pin

//In Master mode, the baud rate can be selected from a baud rate generator which is controlled
//by three bits in the SPCON register: SPR2, SPR1 and SPR0. The Master clock is
//chosen from one of seven clock rates resulting from the division of the internal clock by
//2, 4, 8, 16, 32, 64 or 128.


#define SPI_RATIO_2    0x00 // FCLK PERIPH/2
#define SPI_RATIO_4    0x01 // FCLK PERIPH/4
#define SPI_RATIO_8           0x02 // FCLK
PERIPH/8 #define SPI_RATIO_16  0x03 // FCLK
PERIPH/16
#define SPI_RATIO_32   0x80 // FCLK PERIPH/32
#define SPI_RATIO_64   0x81 // FCLK
PERIPH/64 #define SPI_RATIO_128       0x82
// FCLK PERIPH/128 #define
SPI_RATIO_INVALID      0x83 // No BRG

/*_____M A C R O S___*/
// SPIF: Serial Peripheral data transfer flag
// Cleared by hardware to indicate data transfer is in progress or has been
// approved by a clearing sequence.
// Set by hardware to indicate that the data transfer has been completed.
#define Spif_set()     ((SPSCR & MSK_SPSCR_SPIF) == MSK_SPSCR_SPIF) // If equal, the data
transfer has been completed.
/*_____D E C L A R A T I O N__*/ Uchar spi_set_speed(Uchar data
ratio);
void spi_master_init(Uchar data cpol, Uchar data cpha, Uchar data ssdis, Uchar
data speed); void spi_Sendout(Uchar data inchar);


#endif /* _SPI_DRV_H_ */
/*C**************************************************************************
  * Module:    main.c
/**************************************************************************
  * Copyright (c) 2004 ID TECH inc.,
/**************************************************************************
  * CREATION_DATE:    2004.1.10
/**************************************************************************
  * PURPOSE:
```

```
  * spi library low level functions (init, receive and send functions)
  * and global variables declarations to use with user software application
*****************************************************************************/
/*_____I N C L U D E S_______*/ #include "spi_drv.h"
/*_____M A C R O S___________*/ #define MAX_LEN     512
/*_____D E F I N I T I O N___*/

Uchar data SPI_IPNT; // Temp buffer to store SPI data.
Uchar data Command_OUTbuf[MAX_LEN]; // Command
output buffer Uchar data Command_INbuf[MAX_LEN];
// Command input buffer Uint16 data spilength; //
received command length
Uint16 data Command_Length; // output command length
/*_____D E C L A R A T I O N__*/

void main(void){
Uint16 data i, j; // Internal counter.


       spi_master_init(0, 0, 1, 32); //SPI master mode, initialize to CPOL=0,
CPHA=0, SSDIS=1, bitrate=Fper/32
Enable_spi_interrupt(); // Turn on SPI interrupt in system.
_SPI_SS = 0; // Disable SPI slave during power on, to prevent indeterminate state.


do{ // keep polling...
{
// .............      Other subroutine to handle other tasks
}


if(_DAV_IN){ // If DAV pin is high level, SPI slave has data ready.
                      _SPI_SS = 1; // To Generate a falling edge. Not useful for clock phase
0, but clock phase 1 needs this falling edge.
delay10us(); // Wait for high level get steady.
_SPI_SS = 0; // Pull chip select pin low, ready to start SPI communication. spilength =
0; // Initialize Command_buf pointer.


                      while(_DAV_IN){ // Keep polling DAV pin till it turns low level.
Polling interval is 40us in this demo code.


in this subroutine too.



buffer.
spi_Sendout(0xff); // Send out any data to get SPI slave input, delay 40us
Command_INbuf[spilength++] = SPI_IPNT; // Save data into Command_buf. if(spilength >= MAX_LEN){
// Quit while loop if read the end of input
break;
}


high.
```

```
}
_SPI_SS = 1; // Read out all the data from SPI slave, set chip select pin to idle

for(i = 0; i < spilength; i++){ // Send out data from UART port.
put_byte(Command_INbuf[i]);
}
}


{
// .............      Other subroutine to handle other tasks
}


if(SPIMasterCommandReady){ // If SPI master wants to send a command to SPI slave
                        _SPI_SS = 1; // To Generate a falling edge. Not useful for clock phase
0, but clock phase 1 needs this falling edge.
delay10us(); // Wait for high level get steady.
_SPI_SS = 0; // Pull chip select pin low, ready to start SPI communication.

for(j = 0; j < Command_Length; j++){ // Send out whole command string.
spi_Sendout(Command_OUTbuf[j]);



chip select pin to idle high.

}

{

tasks
}
```

```
}

        _SPI_SS = 1; // Read out all the data from SPI slave, set BeepOn_Long(); // Send out
one beep to indicate command finished.

// .............      Other subroutine to handle other
}
while( TRUE );
}
/*C**************************************************************************
  * Module:    spi_drv.c
/**************************************************************************
  * Copyright (c) 2004 ID TECH inc.,
/**************************************************************************
  * CREATION_DATE:    2004.1.10
/**************************************************************************
  * PURPOSE:
  * spi library low level functions (init, receive and send functions)
  * and global variables declarations to use with user software application
**************************************************************************/
/*_____I N C L U D E S_______*/ #include "spi_drv.h"
/*_____M A C R O S___*/
/*_____D E F I N I T I O N___*/ Uchar transmit_completed = 0; // 0
by default
extern Uchar data SPI_IPNT;
/*_____D E C L A R A T I O N__*/

// Here are some global flags to use with spi library
// These global flags are used to communicate with higher level functions ( user application )
// Here the global variables to communicate with spi interrupt routine

/*F**************************************************************************
  * NAME: spi_isp
*
  * PARAMS: none
  * return: none
  * PURPOSE:
  * spi - interruption program for serial transmission ( Master and Slave mode )
*
  * NOTE:
*****************************************************************
**********/ Interrupt(void spi_isp(void), IRQ_SPI){
if(Spif_set()){ // Quit if data transfer has not been
completed. transmit_completed = 1;    // Set software complete
flag
SPI_IPNT = SPDAT;      // Store SPI input data in SPI_IPNT. SPDAT - Serial Peripheral Data
R
e
g
i
s
t
e
r
return
   ;
    }
}
/*F**************************************************************************
  * NAME: spi_set_speed
  * PARAMS: ratio: spi clock ratio/XTAL
  * return: Uchar: status
  * PURPOSE:
  * Configure the baud rate of the spi, set CR2, CR1, CR0
  * NOTE:
  * This function is only used in spi master mode, called by spi_master_init
*************************************************************
**********/ Uchar spi_set_speed(Uchar data ratio){
switch(ratio){ // Set SPCON register
case 2: SPCON |= SPI_RATIO_2; break; // FCLK PERIPH/2 case 4: SPCON
```

```
|= SPI_RATIO_4; break; // FCLK PERIPH/4 case 8: SPCON |=
SPI_RATIO_8; break; // FCLK PERIPH/8
case 16:SPCON |= SPI_RATIO_16;break; // FCLK PERIPH/16
case 32:SPCON |= SPI_RATIO_32;break; // FCLK PERIPH/32
case 64:SPCON |= SPI_RATIO_64;       break; // FCLK PERIPH/64 case
128:SPCON |= SPI_RATIO_128;   break; // FCLK PERIPH/128
default :      return FALSE;
}
return TRUE;
}

/*F***************************************************************************
  * NAME: spi_master_init
  * PARAMS:
  * cpol: Uchar CPOL value
  * cpha: Uchar CPHA value
  * ssdis: Uchar SSDIS value
  * speed: Uchar spi speed ratio transmission Vs Fper
  * return:    none
*
  * PURPOSE:
  * Initialize the spi module in master mode
*
  * EXAMPLE:
  * spi_master_init(0,0,1,32); // init spi in mater mode with CPOL=0, CPHA=0,
  * // SSDIS=1 and bitrate=Fper/32
  * NOTE:
*****************************************************************************/
void spi_master_init(Uchar data cpol, Uchar data cpha, Uchar data ssdis, Uchar
data speed){ SPCON = 0; // Initialize SPCON: Serial Peripheral Control Register
SPCON |= MSK_SPCON_MSTR; // Serial Peripheral Master: Set to configure the SPI as a Master.
_SPI_SS = 1; // Initialize chip select pin to idle - high
level. spi_set_speed(speed); // Set SPI master speed to
Fper/32.
if(cpol) SPCON |= MSK_SPCON_CPOL; // Cleared to have the SCK set to "0" in idle state.
if(cpha) SPCON |= MSK_SPCON_CPHA; // Cleared to have the data sampled when the SCK leaves the
idle
state
if(ssdis)      SPCON |= MSK_SPCON_SSDIS; // Set to disable chip select in both Master and Slave
modes. Select manually control CS pin.
SPCON |= MSK_SPCON_SPEN; // Set to enable the SPI interface.
}

/*F***************************************************************************
  * NAME: spi_Sendout
  * PARAMS: inchar: the desired character to send out
  * return: none
*
  * PURPOSE:
  * Send out one character
  * NOTE:
  * This function is use only in spi master mode
*****************************************************************
**********/ void spi_Sendout(Uchar data inchar){
Uchar data m;
SPDAT = inchar;// send a data, put the data into SPDAT register
while(!transmit_completed);// wait for transmission complete (interrupt
complete), flag
transmit_completed will be set in SPI interrupt
subroutine. transmit_completed = 0;   // clear software
transmit end flag
m = 4;  // Delay 40us then poll for DAV pin status or send out next
byte. do{
delay10us();
}while(m--)
```

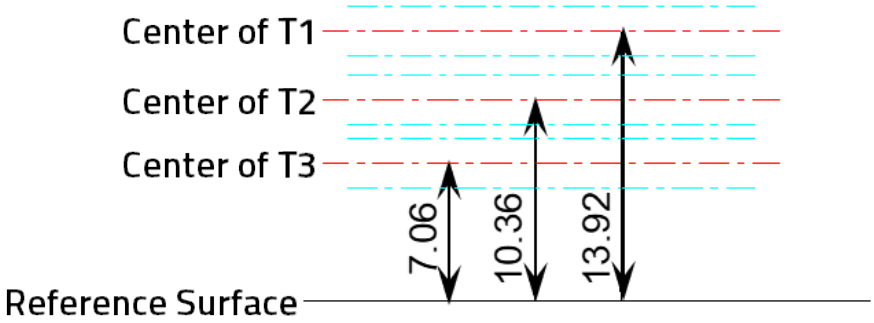
14. APPENDIX H: Installation and Use Guide for Magnetic Heads

This section defines the design specifications ID TECH customers require to install magnetic readers and heads to the correct dimensions and other specific requirements that ensure maximum life and reading reliability. ID TECH has spent years testing magnetic heads with our electronics to determine the best dimensions and characteristics. These factors, combined with the specified reference surface, provide for ID TECH's industry-leading reading reliability. It is extremely important to follow these instructions to achieve the best performance for ID TECH magnetic reading components that are designed into your product(s).

14.1. Track Locations

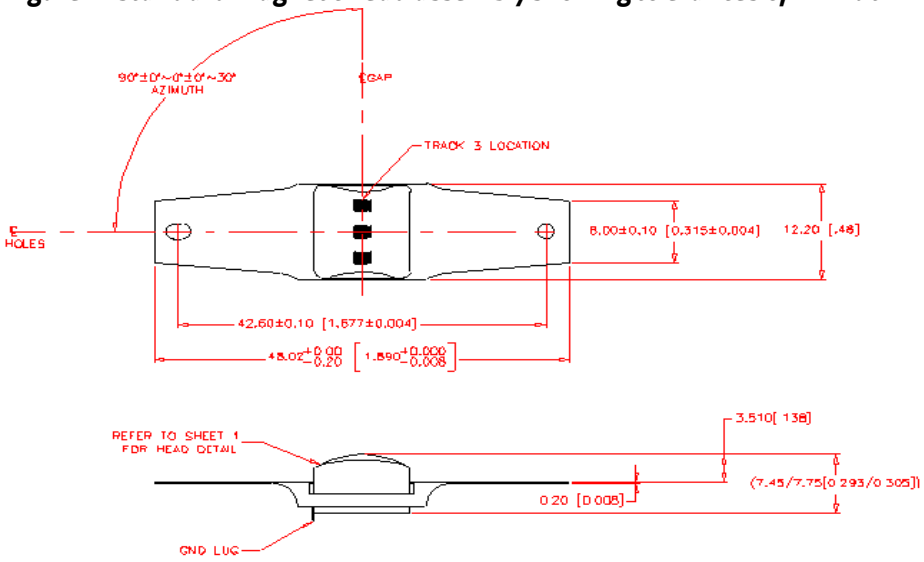
ISO 7810 and ISO 7811 standards define the specification for all "standard" magnetic stripe cards. The location of each magnetic head's track's centerline is shown below in **Figure 1**[1]. ID TECH's heads are installed in spring mounts that have mounting holes located on the centerline of Track 2; refer to **Figure 2** for the 3-track standard magnetic head and (wing) spring mount. The pivot pins must be precisely located to the dimensions shown below in **Figure 1** for the Track 2 centerline, ensuring the read head will be to the proper dimensions for all tracks.

Figure 1: Location of Magnetic Track Centerlines



[1] Note: Magnetic heads can contain one, two, or three tracks, but a three-track head is the most common.

Figure 2: Standard Magnetic head assembly showing tolerances of Azimuth



14.2. Reference Surface/Wear-Plate

The reference surface is an important element for the proper design of all credit card readers because all dimensions for installing magnetic heads are measured from that surface. There are important considerations to understand when designing the reference surface/wear-plate:

1. ID TECH uses stainless steel for the reference surface/wear-plate in most of our credit readers to prevent any measurable wear from the pressure exerted by card edges, assuring negligible wear. Integrators should remember that the magnetic head's installation dimensions are taken from the reference surface and that any variation from those dimensions could have a negative effect on reading reliability.

   Note that electrostatic discharge can be an issue for MagStripe readers. When using metal for the reference surface/wear-plate, integrators should either ground the plate or use conductive plastic to help minimize ESD.

2. ID TECH uses wear-resistant 30% glass-filled plastics in applications where stainless steel is impractical, such as insert readers. In this type of reader, the force from the card's edges is small while inserting and withdrawing cards compared to the force exerted on the wear plate in a conventional swipe reader. ID TECH's insert readers use a 30% glass-filled polycarbonate plastic in the insert reader's rails.

3. It is extremely important that the reference surface not have any bumps or abrupt changes on the surface for *one card length* (3 and 3/8 inches) from the centerline of the read head's gap; any irregularities will cause reading failures. The critical design requirement is that the reference surface/wear-plate must be at minimum flush to above any surface within a card length of the read head's center line. Any surface that is in-line with the card swipe, if

plastic, should at a minimum be of 30% glass-filled plastic because the card's edge will inevitably scrape that surface upon entering and exiting the card swipe. We recommend having stainless steel surfaces on both the entrance and exit area surfaces or have them substantially below the rail's reference surface.

14.3. Card Reader Rails/Slot and Magnetic Head Protrusion

When designing a card reader, engineers must consider the thickness of the media used. Magnetic media comes in various thicknesses, but most readers use cards that are nominally 0.030 inches thick +/- 10%[2]. Some applications do occur where the media is thinner, normally in specialized applications where the media can be as thin as 0.010 inches thick (such as the paper cards used for many parking lot paper tickets).

To accommodate media with minimum and maximum thicknesses, the slot needs to be a between 0.040 and 0.050 inches wide in an area a minimum of 0.5 inches on either side of the centerline of the magnetic head's gap (this gap is located at the apex of the head's radius). The remaining portion of the rails (slot width) can be wider, but never smaller; also remember that it is best to have the reference surface/wear-plate extend for a minimum of 1.5 inches from the magnetic head's gap.

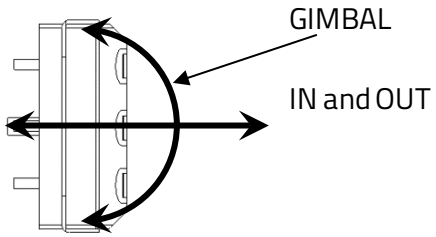
1. Magnetic heads need to be able to rotate on a gimbal (refer to **Figure 3** for an example) to compensate for tilting cards, and therefore must have a minimum deflection. For credit/debit cards used in financial transactions or other applications using standard thickness cards (0.030 inches +/- 10%), the magnetic head's gap (the apex of the protruding radius) should be spaced within 0.010 +/- 0.003 inches from the opposing rail/wall.

To ensure reliable read rates on thinner cards, the head must contact the media and be deflected by a minimum of 0.007 inches (0.18mm); this is regardless of the media thickness, or head configuration. For example, if the media is 0.010 inches (0.25mm) thick, the head face must be positioned a maximum of 0.003 inches (0.08mm) from the opposite wall. To avoid damaging the gap material the head should not contact the opposing rail/surface in the card slot.

[2] Note: 0.030-inch thickness is the dimension for all credit and debit cards, and is common for other cards as well.

2. If the rails are designed without using an ID TECH rail, the minimum slot width should be 0.040 inches wide, at a minimum of 0.5 inches on both sides of the magnetic head's gap. There must also be a smooth transition leading up to the 0.040-inch-wide area of the slot both entering and exiting the magnetic head.

Figure 3



3. When designing insert style readers, make sure the magnetic media on cards can be inserted completely, past the read head, so the reader registers the stop sentinel on the magnetic stripe.

15. APPENDIX I: Firmware Upgrade

ID TECH TM4 SPI SecureHead firmware can be updated through the SPI communication port.

ID TECH can provide Windows-based utility software, FWUpdate.exe, and an RS-232 to SPI converter board for reference. The customer can also develop their own software to upgrade the firmware. (Prerequisite: The host must already be in communication with SecureHead. It must support regular commands like "read firmware version.")

Refer also to ID TECH's Tech Note 015 on TriMag IV firmware upgrade procedures, available for download at https://atlassian.idtechproducts.com/confluence/display/KB/Downloads+-+Home.

15.1. Procedure

TriMag IV firmware can be updated using the following commands.

Except where noted, commands should be wrapped in STX (0x02) and ETX (0x03), followed by a one-byte LRC (calculated as the XOR of all preceding bytes including STX and ETX).

Also, except where noted, a successful response will begin with ACK (0x06).

15.2. Basic steps:

1. Read firmware version (52 22 88 command). This is to confirm current reader is working 2. Erase firmware (53 7E 0D 31 01 02 03 04 05 06 07 08 04 03 02 01).
The firmware will be erased in about 2 seconds, then rise DAV line to request the send of 0x5A. Host needs to read this response.

Note: The DAV line will be high for 500 mS. If software does not read a response, the SecureHead will shift to RS232 communication. In such a case, you must cycle the SecureHead power and read response within the 500 mS DAV high period to get the 5A byte.

We suggest waiting another 3 seconds after reading the response, then perform the following loading sequence.

15.3. Load new firmware

1. Send hex byte 0xBD to start loading.
2. Open firmware bin file and send the *whole file* to the SecureHead.

Note: The new firmware file is a binary file that contains 26K bytes encrypted firmware and 4 bytes CheckSum and LRC. The CheckSum and LRC will be checked by SecureHead. The SecureHead will decide to reject or accept the firmware download. (The host does not need to check these bytes. Just send the whole file.)

3. Wait for DAV line high and read one-byte response.
4. Wait for 3 seconds.

15.4. Example

Following is an example when loading firmware with ID TECH *FWUpdate* software.

Step 1: Review current firmware version:

OUT	02 52 22 88 03 f9		
IN	06 02 49 44 20 54 45	43	..ID TEC 250ms
	48 20 54 4d 34 20 53	65	H TM4 Se
	63 75 72 65 48 65 61	64	cureHead
	20 53 50 49 20 52 65	61	SPI Rea
	64 65 72 20 56 31 2e	32	der V1.2 4.049..
	34 2e 30 34 39 03 17		
	B. Erase current firmware:		
OUT	02 53 7e 0d 31 01 02	03	.S..1... 18sc
	04 05 06 07 08 04 03	02	.....
	01 03 1c	...	
IN	5a	Z	2.2sc

Note: It takes about 2 seconds for SecureHead to finish erasing firmware. The host should wait for DAV line rise and read the response 5A. The host might wait another 3 seconds to perform following loading step.

Step 2: Download firmware:

1. Send one byte for getting into download mode: BD.
2. Send encrypted bin file (new firmware file).
3. Wait for DAV line rise, get one-byte response, ignore it.
4. Wait a few seconds (about 3 seconds).

Step 3: Check new firmware version

OUT	02 52 22 88	03 f9	.R"...	5.0sc
IN	06 02 49 44	20 54 45 43	..ID TEC	251ms
	48 20 54 4d	34 20 53 65	H TM4 Se	
	63 75 72 65	48 65 61 64	cureHead	
	20 53 50 49	20 52 65 61	SPI Rea	
	64 65 72 20	56 31 2e 32	der V1.2	
	34 2e 30 35	30 03 1f	4.050..	