IDTECH®
Value through Innovation

USER MANUAL

SecureHead™
Encrypted Magnetic Read Head
With TriMagIV ASIC

SPI Interface

CE FC

80101502-002-E
5/22/2017

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.

Revision History

Revision	Date	Description of Changes	By
A	10/15/2015	Initial Release	JH
B	9/21/2016	Added discussion of Samsung Pay decoding in 4.4.3. Added firmware upgradability to Introduction.	JH KT
C	11/1/2016	Added Firmware Upgrade appendix.	KT
D	2/17/2017	Added bits 3-7 definition in notes3 – "Clear/Mask Data Sent Status" in 4.14.8	JW
E	5/15/2017 5/22/2017	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 KT

Table of Contents

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. Also, firmware can 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
Standby Power supply Current	0.03 mA

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 period (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 at https://atlassian.idtechproducts.com/confluence/display/KB/Downloads+-+Home.)

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

ESD	+4kV discharge to head can

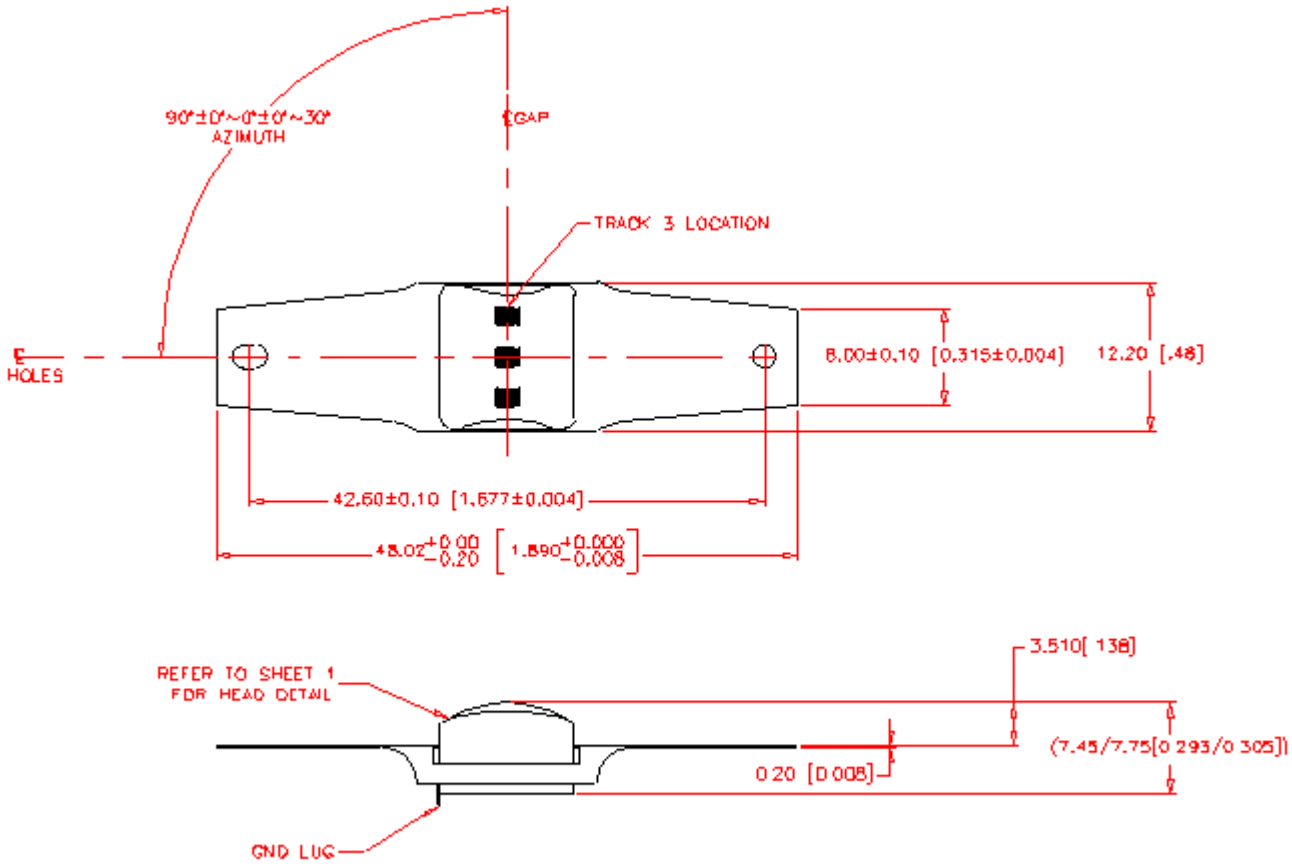
Environmental

Operating Temperature	0 °C to 55 °C
Storage Temperature	-40 °C to 70 °C
Humidity	-10% to 90% non-condensing

Mechanical

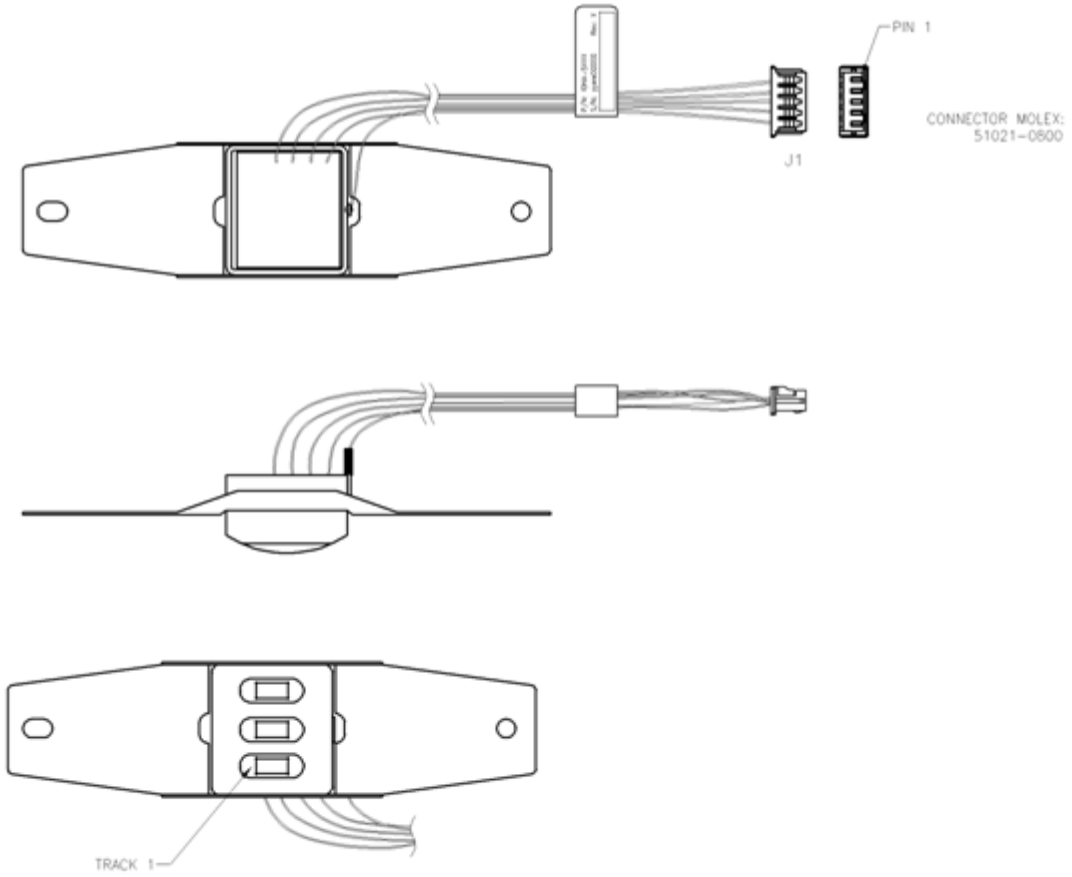
Weight	5.67 grams
Cable Length	125 +/- 6.4 mm

Dimension:

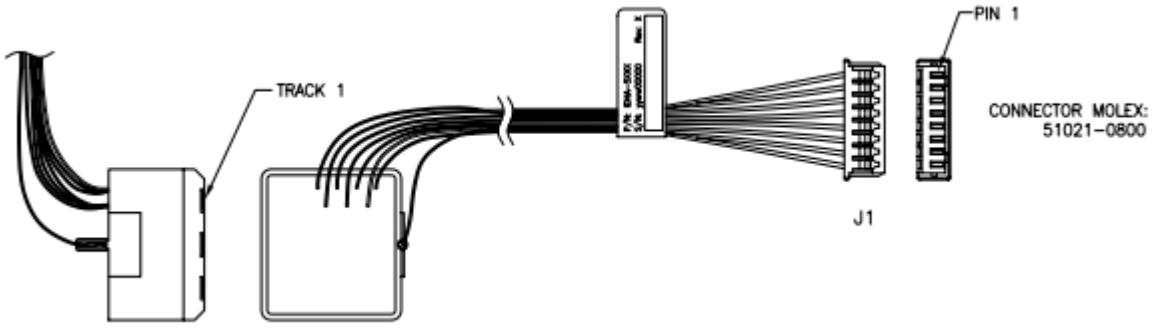


Mounting Options:

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. 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

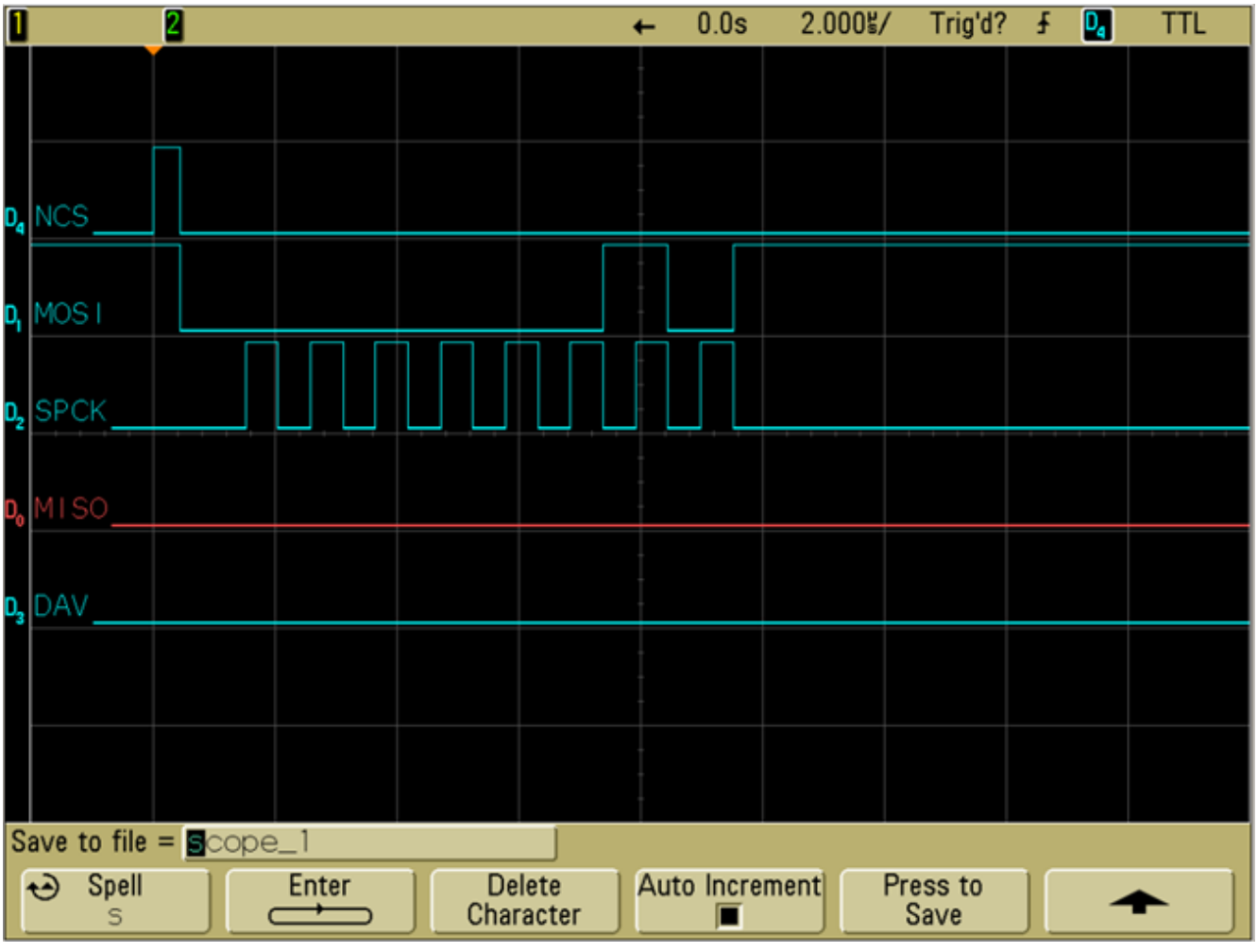
The clock polarity and phase have four different options with respect to the data. The serial clock input frequency can go up to 1M bps.

- 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's 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 once 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.

.

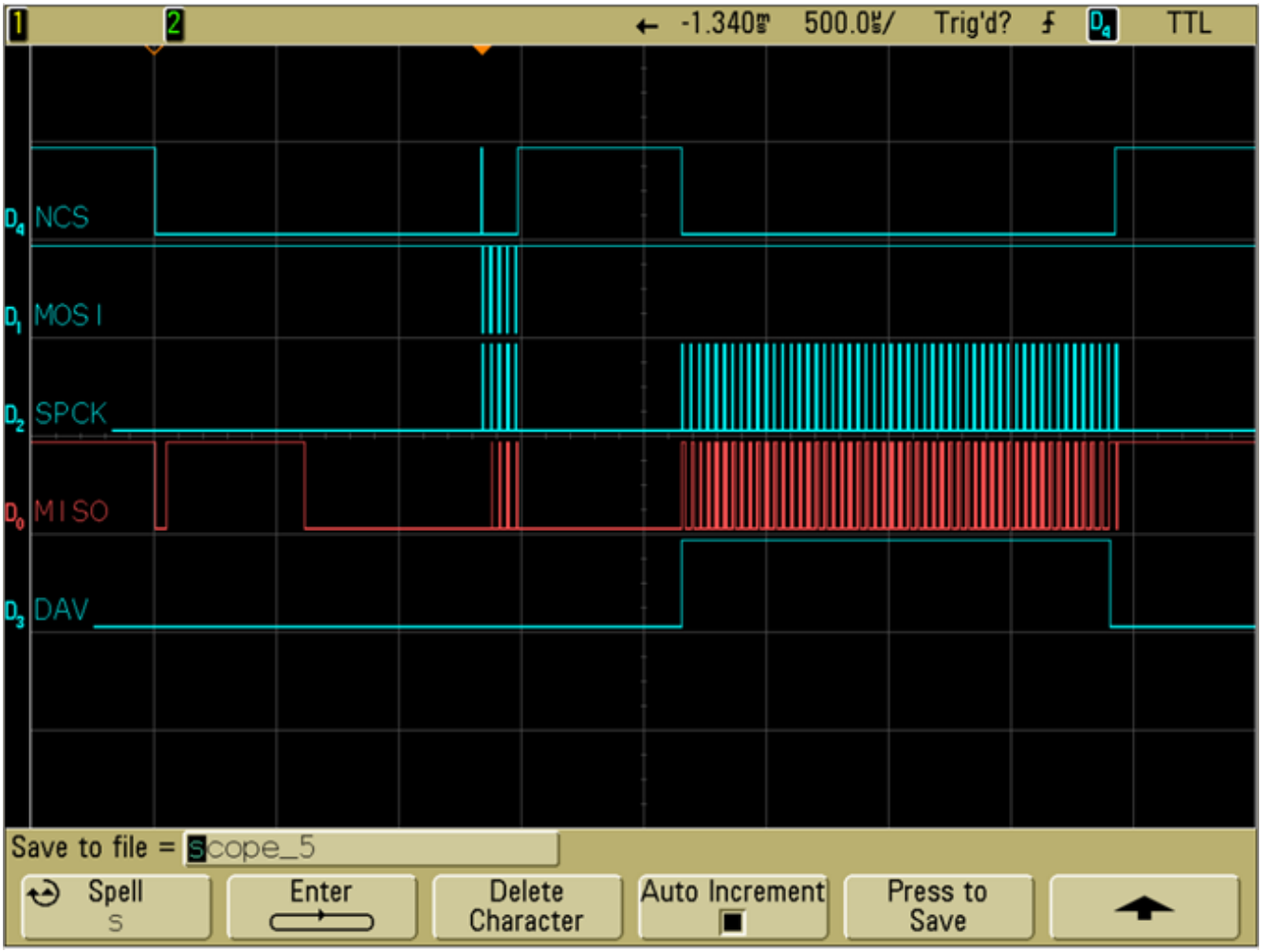
3.5. Data Available Output (DAV)

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 would 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:



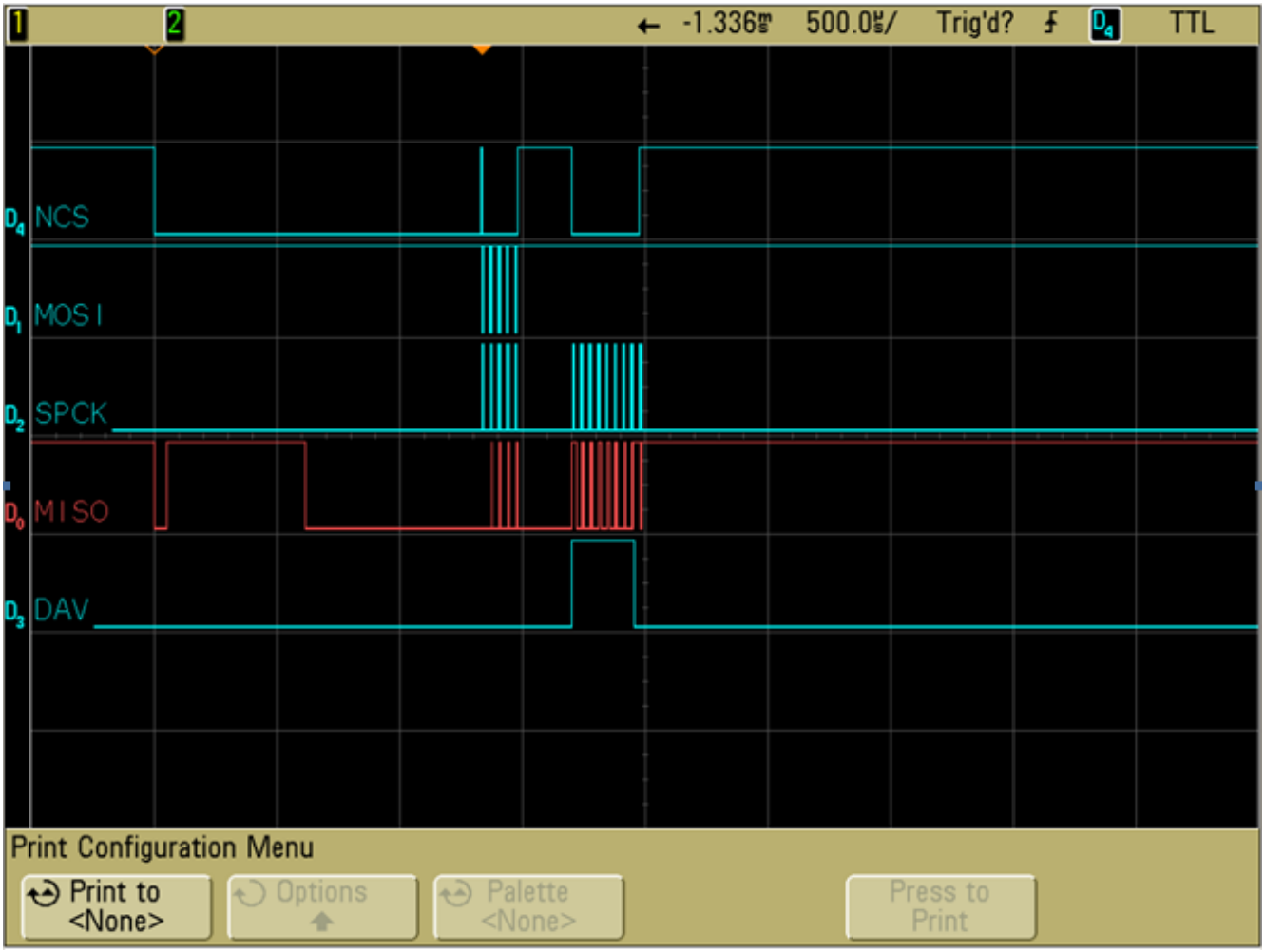
After the command is received and the response is ready, the DAV would be set to *high* for the host to receive response. After the response is received, the DAV would be *low*, indicating there is no more data to be transmitted.

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.

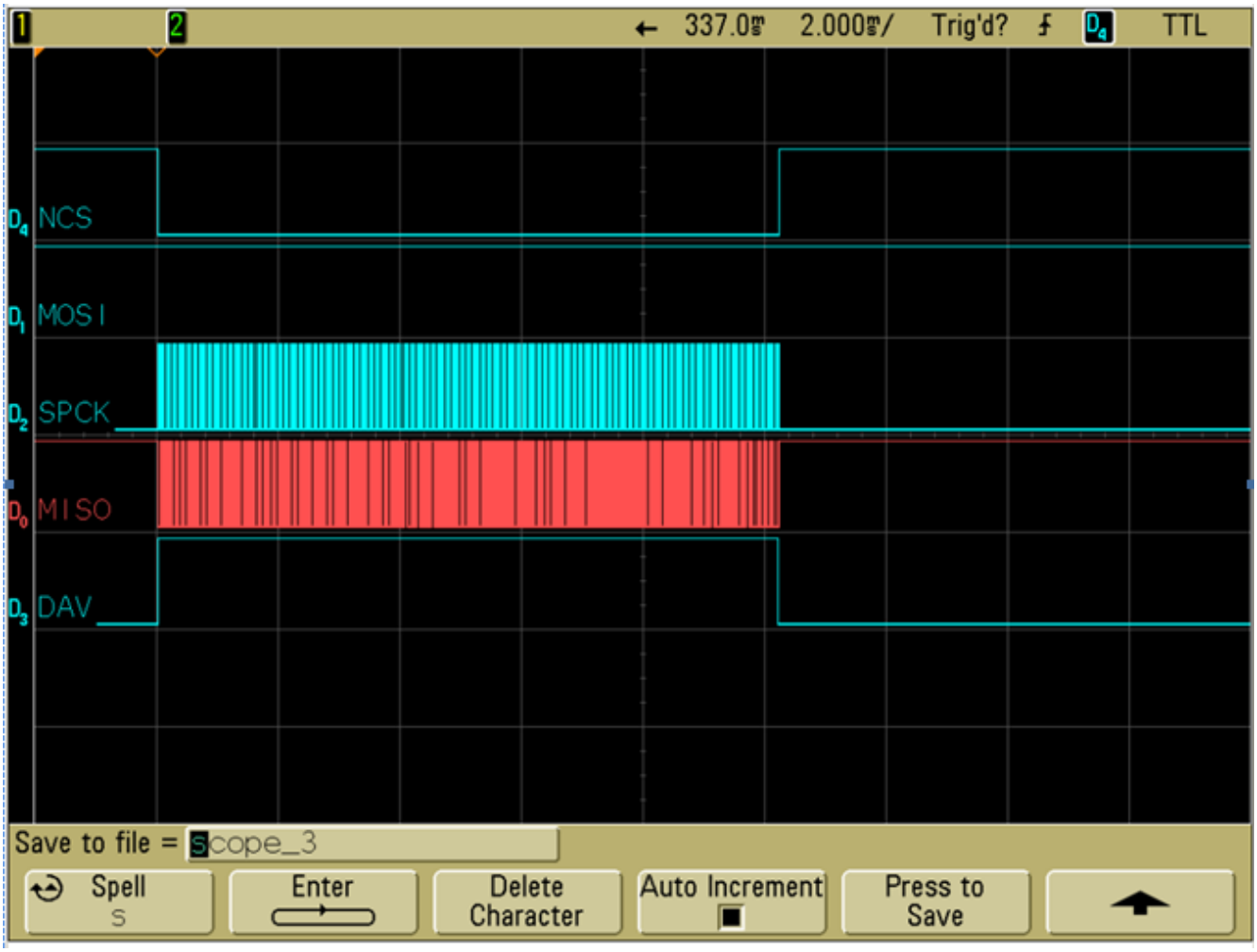
After the last byte of response is sent, the DAV is pulled low. If user polls DAV status to check whether there are data available, we suggest to use 100μs polling interval and throw away any data when DAV is low.

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 a 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. Since 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. Since 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 0x5A.

4.1. Command Structure

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>

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 particular 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 SecureHead has a 50ms start up period. During this period, it doesn't support communication nor card reading. If the terminal tries to talk to SecureHead during this period, SecureHead may not be functional until restart.

SecureHead also takes time to process a command. During that processing time, it will not respond to a new command.

Card swipe actions can interrupt command processing; instead of sending command responses, SecureHead will only send card data.

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.

A minimum delay of 50μs is required between each character sent to SecureHead through SPI interface.

If the radio-frequency noise of the working environment interferes with the SPI communication, to the level that SecureHead can't receive commands correctly, SecureHead will *restart* in an attempt to reduce the impact. Restart takes 600ms. The terminal won't get any command response in this situation.

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 **boldface**.

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.

4.4.1. Change to Default Settings

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

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

4.4.2. MSR Reading Settings

Enable or Disable the SecureHead. If the reader is disabled, no data will be sent out to the host.
<STX><S><1Ah><01h><MSR Reading Settings><ETX><CheckSum>
MSR Reading Settings:

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

4.4.3. Decoding Method Settings

The SecureHead can support four kinds of decoded directions.
<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.

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 Track 1 and Track 2 data consecutively, making it possible to disambiguate the tracks.

If the reading device receives identical MSR data for multiple tracks, MSR processing will ignore Track 2 and Track 3 data if the card data is ISO 7-bit encoded, treating it as Track 1 data . If the data are 5-bit encoded, it is received as Track 2 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.

4.5. Review Settings

<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>

4.6. Review Firmware Version

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

This command gets the device firmware version.

4.7. Review Serial Number

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

This command gets the device serial number.

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

4.8.1. Terminator Setting

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

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

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

4.8.2. 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.

<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>.

4.8.3. 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.

<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>.

4.8.4. 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.

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

Where:
<n> = 34h for track 1; 35h for track 2 and 36h for track 3
<Len> = the number of bytes of prefix string
<Prefix> = {string length}{string}
NOTE: String length is one byte, maximum six.

4.8.5. 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.

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

Where:

<n> = 37h for track 1; 38h for track 2 and 39h for track 3
<Len> = the number of bytes of suffix string
<Suffix> = {string length}{string}
NOTE: String length is one byte, maximum six.

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

4.9.1. Track Selection

There are up to three tracks of encoded data on a magnetic stripe.

This option selects the tracks that will be read and decoded.

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

Track_Selection Settings:

"0" Any Track
"1" Require Track 1 Only
"2" Require Track 2 Only
"3" Require Track 1 & Track 2
"4" Require Track 3 Only
"5" Require Track 1 & Track 3
"6" Require Track 2 & Track 3
"7" Require All Three Tracks
"8" Any Track 1 & 2
"9" Any Track 2 & 3

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

4.9.2. Track Separator Selection

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

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

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

4.9.3. Start/End Sentinel and Track 2 Account Number Only

The SecureHead can be set to either send, or not send, the Start/End sentinel, and to send either the Track 2 account number only, or all the encoded data on Track 2. (The Track 2 account number setting doesn't affect the output of Track 1 and Track 3.)

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

SendOption:
"0" Don't send start/end sentinel and send all data on Track 2
"1" Send start/end sentinel and send all data on Track 2
"2" Don't send start/end sentinel and send account # on Track 2
"3" Send start/end sentinel and send account number on Track 2

4.10. Security Settings

4.10.1.Select Key Management Type

<STX><S><58h><01h><Key Management Type><ETX><CheckSum>

Key Management Type:
"0" Fix key management
"1" DUKPT Key management

4.10.2.External Authenticate Command (Fixed Key Only)

Before a security related command is executed, an authentication process is required to make sure the device key used is correct. For example, authentication is generally required whenever encryption is enabled/ disabled or the device key is changed. Once the authentication process has finished successfully, the same process would not be needed again until the device is restarted.

- First, the host would get a data block which is generated by encrypting a random 8-byte data using TDES algorithm.
- The host then decrypts the data block using TDES algorithm using the current device key.
- The host initiates an External Authenticate Command to verify the decrypted 8 bytes of random data
- The device checks to see if the data matches the random data generated. If the data are the same, authentication process is successful. If it fails, the host must start the authentication process again until it's succeed, before any security related featured can be changed.

Commands:

1) Retrieve Encrypted Challenge Command

Host -> Device:
<STX><R><74h><ETX><CheckSum>

Device -> Host:
<ACK><STX><8 bytes of TDES-encrypted random data><ETX><CheckSum> (success)
<NAK> (fail)

2) Send External Authenticate Command

Host -> Device:
<STX><S><74h><08h><8 bytes of original random data><ETX><CheckSum>

Device -> Host:

<ACK> (success)
<NAK> (fail)

4.10.3.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.

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

Encryption Settings:
"0" Encryption Disabled
"1" Enable TDES Encryption
"2" Enable AES Encryption

4.11. Review KSN (DUKPT Key management only)

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

This command gets the DUKPT key serial number and counter.

4.12. Review Security Level

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

This command gets the current security level.

4.13. 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:
<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 <Lengh_H>

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

<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

Feasible settings of these new functions are listed below.

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	'0'	'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 ocurs: 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

4.14.3.Security Management

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

- Can include Device Serial Number
- Can encrypt track 1 and track 2 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:

- 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. Once this limit has been reached, key injection will need to occur again before any more transactions can be done.

- 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.

- 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.

- 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.

4.14.4. 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.

4.14.5.Check Card Format

- 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 2nd byte is 'B'.
There is only one '=' in track 2 and the position of '=' is between 13th ~ 20th character.
Total length of track 2 should above 21 characters.

- 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.

- Others (Customer card)

4.14.6.MSR Data Masking

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

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'

4.14.7.Level 1 and 2 Data Output Format

Magnetic Track Basic Decoded Data Format

Track 1: <SS1><T1 Data><ES><Track Separator>
Track 2: <SS2><T2 Data><ES><Track Separator>
Track 3: <SS3><T3 Data><ES><Terminator>

Where: SS1 (start sentinel track 1) = %
SS2 (start sentinel track 2) = ;
SS3 (start sentinel track 3) = ; for ISO, % for AAMVA
ES (end sentinel all tracks) = ?
Track Separator = Carriage Return
Terminator = Carriage Return

Language: US English

Magnetic Track Basic Raw Data Format

Track 1: <01><T1 Raw Data><CR>
Track 2: <02><T2 Raw Data><CR>
Track 3: <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 doesn't reach 0x60.

Language: US English

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.

4.14.8. 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 track 3 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>

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

Where <STX> = 02h, <ETX> = 03h

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

Data length byte:
LenL – Overall length of data, low bits
LenH – Overall length of data, high bits

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 track 2)	

Track Status:

MSR sampling and decode status

MB							LB
B7	B6	B5	B4	B3	B2	B1	B0

B0 1: Track 1 decode success (0: Track 1 decode fail)
B1 1: Track 2 decode success (0: Track 2 decode fail)
B2 1: Track 3 decode success (0: Track 3 decode fail)
B3 1: Track 1 sampling data exists (0: Track 1 sampling data does not exist)
B4 1: Track 2 sampling data exists (0: Track 2 sampling data does not exist)
B5 1: Track 3 sampling data exists (0: Track 3 sampling data does not exist)
B6 0—reserved for future
B7 0—reserved for future

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:
Track 1 maximum length is 79 alphanumeric characters.
Track 2 maximum length is 40 numeric digits.
Track 3 maximum length is 107 numeric digits.

Clear/Masked Data sent status

Bit 0: 1--- if Track1 clear/mask data present
Bit 1: 1--- if Track2 clear/mask data present
Bit 2: 1--- if 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, 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

Encrypted Hash Data sent status

Bit 0: if 1—track1 encrypted data present
Bit 1: if 1—track2 encrypted data present
Bit 2: if 1—track3 encrypted data present
Bit 3: if 1—track1 hash data present
Bit 4: if 1—track2 hash data present
Bit 5: if 1—track3 hash data present
Bit 6: if 1—session ID present
Bit 7: if 1—KSN present

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).

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 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 preformed for the right side of the key, combine the two key parts to create the Data Key.

Track Hashed Data

SecureHead reader uses SHA-1 to generate hashed data for both track 1, track 2 and track 3 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.

1. Encryption Output Format Setting:

   Command: 53 85 01 <Encryption Format>

   Encryption Format:
   '0': No longer supported
   '1': Enhanced Encryption Format

2. Encryption Option Setting: (for enhanced encryption format only)

   Command: 53 84 01 <Encryption Option>

   Encryption Option: (**default 08h**)
   bit0: 1 – track 1 force encrypt
   bit1: 1 – track 2 force encrypt
   bit2: 1 – track 3 force encrypt
   bit3: 1 – track 3 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.
   3) When force encrypt is not set, the data will be encrypted in original encryption format, that is, only track 1 and track 2 of type 0 cards (ABA bank cards) will be encrypted.

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

4. Mask Option Setting: (for enhanced encryption format only)

   Command: 53 86 01 <Mask Option>

   Mask Option: (**Default: 0x07**)
   bit0: 1 – tk1 mask data allow to send when encrypted
   bit1: 1 – tk2 mask data allow to send when encrypted
   bit2: 1 – tk3 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.

Note 1 : Card Encode Type

Card Type will be 8x for enhanced encryption format and 0x for original encryption format

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

For Type 04 or 84 Raw data format, all tracks are encrypted and no mask data is sent. No track indicator ‘01’, ‘02’ or ‘03’ in front of each track. Track indicator ‘01’,’02’ and ‘03’ will still exist for non-encrypted mode.

Note 2: Track 1-3 status byte

Field 4:
Bit 0: 1— track 1 decoded data present

Bit 1: 1— track 2 decoded data present
Bit 2: 1— track 3 decoded data present
Bit 3: 1— track 1 sampling data present
Bit 4: 1— track 2 sampling data present
Bit 5: 1— track 3 sampling data present
Bit 6: 1--- Field 10 “optional bytes length” exists (0: No Field 10)

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 —track 1 clear/mask data present
Bit 1: 1— track 2 clear/mask data present
Bit 2: 1— track 3 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.
Bit7: 1 – Serial # present; 0- not present

Note 4: Encrypted/Hash data sent status

Field 9: Encrypted data sent status
Bit 0: 1— track 1 encrypted data present
Bit 1: 1— track 2 encrypted data present
Bit 2: 1— track 3 encrypted data present
Bit 3: 1— track 1 hash data present
Bit 4: 1— track 2 hash data present
Bit 5: 1— track 3 hash data present
Bit 6: 1—session ID present
Bit 7: 1—KSN present

4.14.9.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.

APPENDIX A. DEFAULT SETTING TABLE

Default Setting Table

MSR Reading	Enable
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

APPENDIX B: MAGNETIC STRIPE STANDARD FORMATS

ISO Credit Card Format

ISO stands for International Standards Organization

Track 1

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

Track 2

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

AAMVA Driver's License Format

Track 1

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

Track 2

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	
	ID/DL	5
h	End Sentinel	1
I	Linear Redundancy Check (LRC) Character	1

Track 3

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

APPENDIX C: OTHER MODE CARD DATA OUTPUT

There is an optional data output format supported by SecureHead in order 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>.

APPENDIX D: GUIDE TO ENCRYPTING AND DECRYPTING DATA

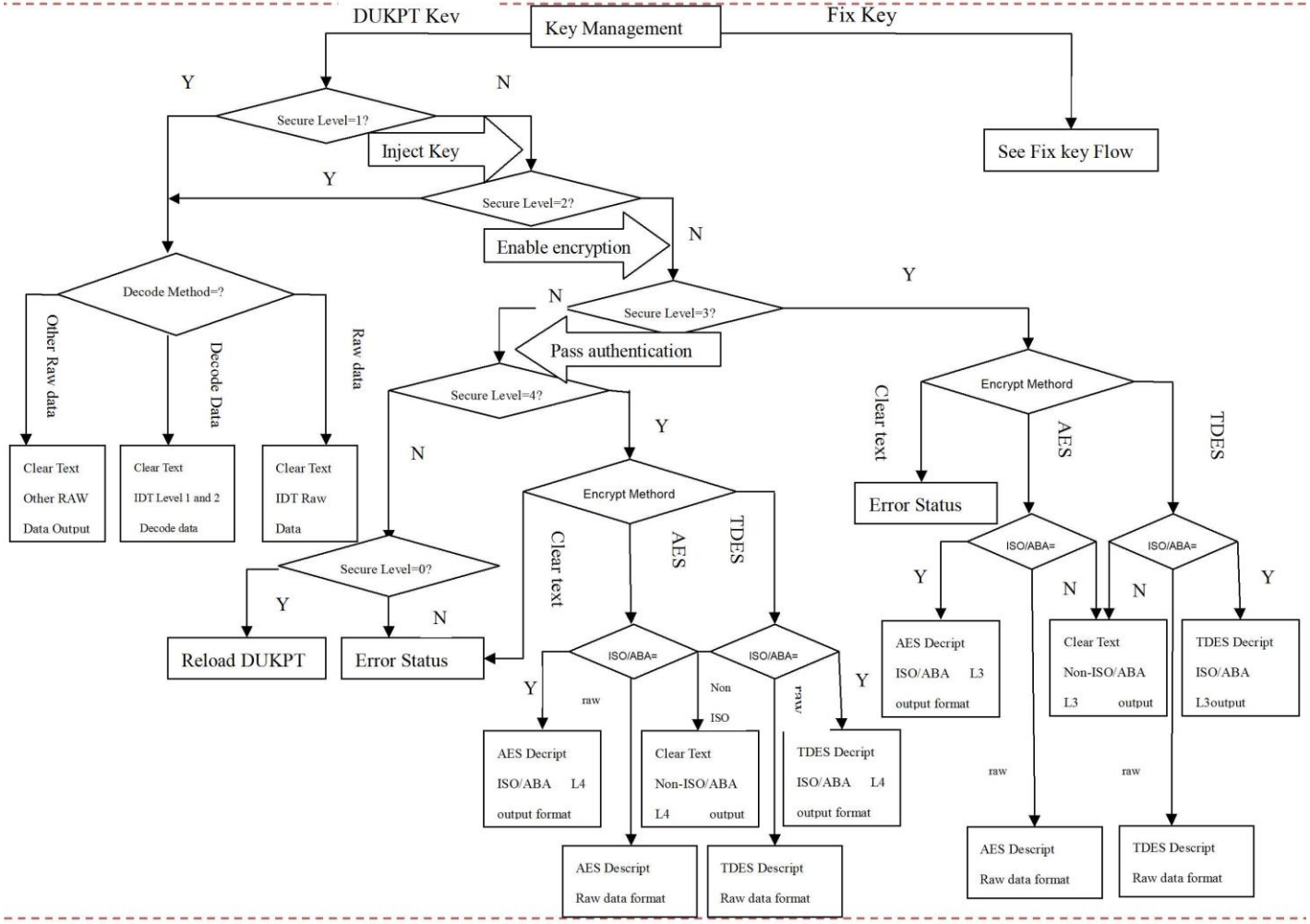
The encryption mode used by SecureHead is called Cipher-block Chaining (CBC). With this method, each block of data is XOR'ed with the previous data block before being encrypted. The encryption of each block depends on all the previous blocks. As a result, each encrypted data block would need to be decrypted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

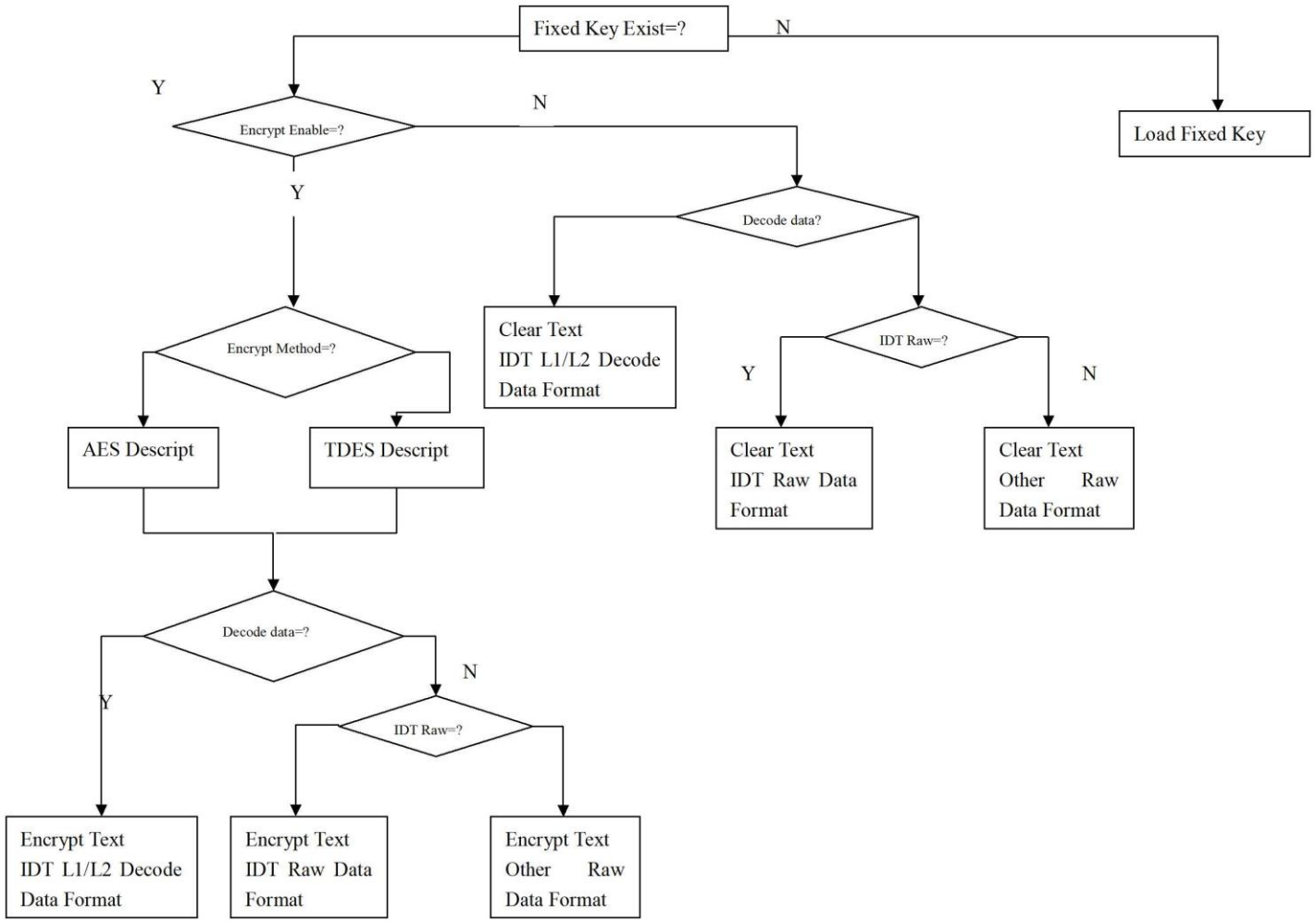
To encrypt the data, first generate 8 bytes of 0x00 to use as an initialization vector which is XOR'ed with the first data block before it is encrypted. Then the data is encrypted with the device key using TDES algorithm. The result is again XOR'ed with the next 8-byte data block before it is encrypted. 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, since it is XOR'ing with 00h bytes.

APPENDIX E: KEY MANAGEMENT FLOW CHART





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.

029801803F48236B03BF252A343236362A2A2A2A2A2A2A393939395E42555348204A522F47454
F52474520572E4D525E2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2
A2A2A2A2A3F2A3B343236362A2A2A2A2A2A2A393939393D2A2A2A2A2A2A2A2A2A2A2A2A2
A2A2A2A3F2ADA7F2A52BD3F6DD8B96C50FC39C7E6AF22F06ED1F033BE0FB23D6BD33DC5A1
F808512F7AE18D47A60CC3F4559B1B093563BE7E07459072ABF8FAAB5338C6CC8815FF87797AE
3A7BEAB3B10A3FBC230FBFB941FAC9E82649981AE79F2632156E775A06AEDAFAF6F0A184318
C5209E55AD44A9CCF6A78AC240F791B63284E15B4019102BA6C505814B585816CA3C2D2F42A9
9B1B9773EF1B116E005B7CD8681860D174E6AD316A0ECDBC687115FC89360AEE7E430140A7B7
91589CCAADB6D6872B78433C3A25DA9DDAE83F12FEFAB530CE405B701131D2FBAAD970248
A456000933418AC88F65E1DB7ED4D10973F99DFC8463FF6DF113B6226C4898A9D355057ECAF11
A5598F02CA31688861C157C1CE2E0F72CE0F3BB598A614EAABB16299490119000000000206E203

STX, Length(LSB, MSB), card type, track status, length track 1, length track 2, length track 3
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 track 1
23—length of track 2
6B—length of track 3
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—track 3 hash data present
Bit 4=1—track 2 hash data present
Bit 3-1—track 1 hash data present
Bit 2=1—track 3 encrypted data present
Bit 1=1—track 2 encrypted data present
Bit 0=1—track 1 encrypted data present

Track 1 data masked (length 0x48)
252A343236362A2A2A2A2A2A2A393939395E42555348204A522F47454F52474520572E4D525E2
A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A3F2A

Track 1 masked data in ASCII
%*4266********9999^BUSH JR/GEORGE W.MR^**********?*

Track 2 data in hex masked (length 0x23)
3B343236362A2A2A2A2A2A2A2A393939393D2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A3F2A

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

In this example there is no Track 3 data either clear or masked (encrypted and hashed data is below)

Track 1 encrypted length 0x48 rounded up to 8 bytes = 0x48 (72 decimal)
DA7F2A52BD3F6DD8B96C50FC39C7E6AF22F06ED1F033BE0FB23D6BD33DC5A1F8
08512F7AE18D47A60CC3F4559B1B093563BE7E07459072ABF8FAAB5338C6CC88
15FF87797AE3A7BE

Track 2 encrypted length 0x32 rounded up to 8 bytes =0x38 (56 decimal)
AB3B10A3FBC230FBFB941FAC9E82649981AE79F2632156E775A06AEDAFAF6F0A
184318C5209E55AD

Track 3 encrypted length 0x6B rounded up to 8 bytes =0x70 (64 decimal)
44A9CCF6A78AC240F791B63284E15B4019102BA6C505814B585816CA3C2D2F42
A99B1B9773EF1B116E005B7CD8681860D174E6AD316A0ECDBC687115FC89360A
EE7E430140A7B791589CCAADB6D6872B78433C3A25DA9DDAE83F12FEFAB530CE
405B701131D2FBAAD970248A45600093

Track 1 data hashed length 20 bytes
3418AC88F65E1DB7ED4D10973F99DFC8463FF6DF

Track 2 data hashed length 20 bytes
113B6226C4898A9D355057ECAF11A5598F02CA31

Track 3 data hashed length 20 bytes
688861C157C1CE2E0F72CE0F3BB598A614EAABB1

KSN length 10 bytes
62994901190000000002

LCR, check sum and ETX
06E203

Clear/Masked Data in ASCII:
Track 1: %*4266********9999^BUSH JR/GEORGE W.MR^********************************?*
Track 2: ;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:
Track 1 decrypted
%B4266841088889999^BUSH JR/GEORGE W.MR^08091011000011000000000046000000?!
Track 2 decrypted
;4266841088889999=08091011000046?0
Track 3 decrypted
;3333333337676760707077676763333333333767676070707767676333333333337676760707077676763333333333767676070707767676

Track 1 decrypted data in hex including padding zeros (but there are no pad bytes here)
254234323636383431303838383839393939 5E42555348204A522F47454F52474520572E4D525E30383
039313031313030303031313030303030303030303436303030303030303F21

Track 2 decrypted data in hex including padding zeros
3B343236363834313038383838393939393D303830393130313130303030303436 3F300000000000

Track 3 decrypted data in hex including padding zeros
3B333333333333333333333373637363736303730373037373637363736333333333333333333333373637
363736303730373037373637363736333333333333333333333373637363736303730373037373637363736373
633333333333333333333373637363736303730373F320000000000

APPENDIX G: EXAMPLE OF IDTECH RAW DATA DECRYPTION

Original Raw Data Forward Direction:
01D67C81020408102D448102040810204289 0A350854A2FB3EE4BA3D4065B67A9C391F582A42B9
9A858A90AF60852B14AA628A0D
028FC210842C18421084030092040B51581F24B56074404811160D

Original Raw Data Backward Direction:
01A28CAA51A9420DEA12A342B33A84A835F13872BCDB4C0578BA4EF9BE8A542158A12284081
02040810245681020408 1027CD60D
02D11024045C0D5A49F03515A0409201804210843068421087E20D

Note:

1. There is track number before each track. Track 1 is 01, Track 2 is 02, Track 3 is 03.
2. There is track separator after each track: 0D

Example of decryption of a two track ABA card with the original encryption format. For both Fix & DUKPT key management.
SecureHead Reader with default settings

Key for all examples is
0123456789ABCDEFFEDCBA9876543210

<u>Original Encryption Format</u>
original encryption format (this can be recognized because the high bit of the fourth byte underlined (00) is 0.
028700041B331A0027D2E435CEE303F007E977B598B7E3C57C76F4445E309F6916C0321A0F915B6
E490813498839049FE5204762327C3C758C5BF82542DEEDD8D6AF88019149A702FF2D43BD4AD6
0031FA450720B00D7808E15F3D5B29AE712C64A1212E9AF6F483BD40798A9FF2DDE77D046620B
55BCE94A4D5534CF57E7E07629949011A0000000001871D03

STX, Length (LSB, MSB), card type, track status, length track 1, length track 2, length track 3
02 8700 04 1B 33 1A 00

Track 1 & 2 encrypted length 0x33+0x1A rounded up to 8 bytes =0x4D -> 0x50 (80 decimal)
27D2E435CEE303F007E977B598B7E3C57C76F4445E309F6916C0321A0F915B6E490813498839049
FE5204762327C3C758C5BF82542DEEDD8D6AF88019149A702FF2D43BD4AD60031FA450720B00
D7808

Track 1 hashed
E15F3D5B29AE712C64A1212E9AF6F483BD40798A

Track 2 hashed

9FF2DDE77D046620B55BCE94A4D5534CF57E7E07

KSN
629949011A0000000001

LRC, checksum and ETX
87 1D 03

Key Value: 8A 60 A3 EB 80 87 63 52 B8 F5 05 CD A8 3C 33 70
KSN: 62 99 49 01 1A 00 00 00 00 01

Decrypted Raw Data:
01D67C81020408102D448102040810204289 0A350854A2FB3EE4BA3D4065B67A9C391F582A42B9
9A858A90AF60852B14AA628A
028FC210842C18421084030092040B51581F24B5607440481116

-=-=-=--=-=-=--=-=-=--=-=-=--=-=-=--=-=-=--=-=-=--=-=-=--=-=-=--=-=-=--=-=-=--=-=-=--=-=-=--=-

APPENDIX H: 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 inverval
is 40us in this demo code.
                              spi_Sendout(0xff); // Send out any data to get SPI slave input, delay 40us
in this subroutine too.
                              Command_INbuf[spilength++] = SPI_IPNT; // Save data into Command_buf.

                              if(spilength >= MAX_LEN){ // Quit while loop if read the end of input
buffer.
                                     break;
                              }
                     }
                     _SPI_SS = 1; // Read out all the data from SPI slave, set chip select pin to idle
high.


                     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]);
                                }

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

                        {
                            // .............                    Other subroutine to handle other
tasks
                        }
        }
        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 arec used to communicate with higher level functions ( user application )
// Here the globals 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 hasn't been completed.
                transmit_completed = 1;       // Set software complete flag
                SPI_IPNT = SPDAT;      // Store SPI input data in SPI_IPNT. SPDAT - Serial Peripheral Data
Register
    return;
        }
}
```

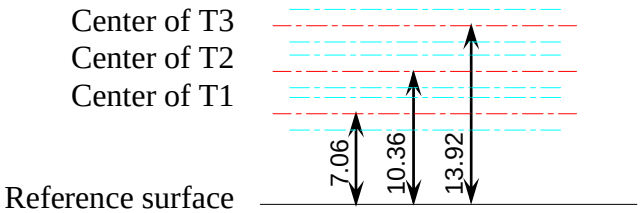
```
/*F**************************************************************************
* NAME: spi_set_speed
*-------------------------------------------------------------------------
* PARAMS: ratio: spi clock ratio/XTAL
* return: Uchar: status
*-------------------------------------------------------------------------
* 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 character want 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 transmition 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--)
```

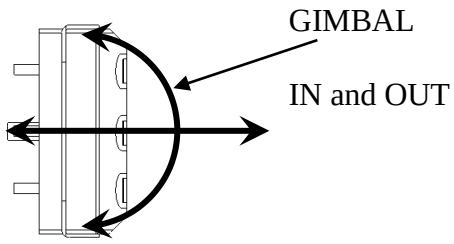
APPENDIX I: MAGNETIC HEADS MECHANICAL DESIGN GUIDELINES

This installation guide is specifically to be used when installing ID TECH's magnetic heads with spring mounts.

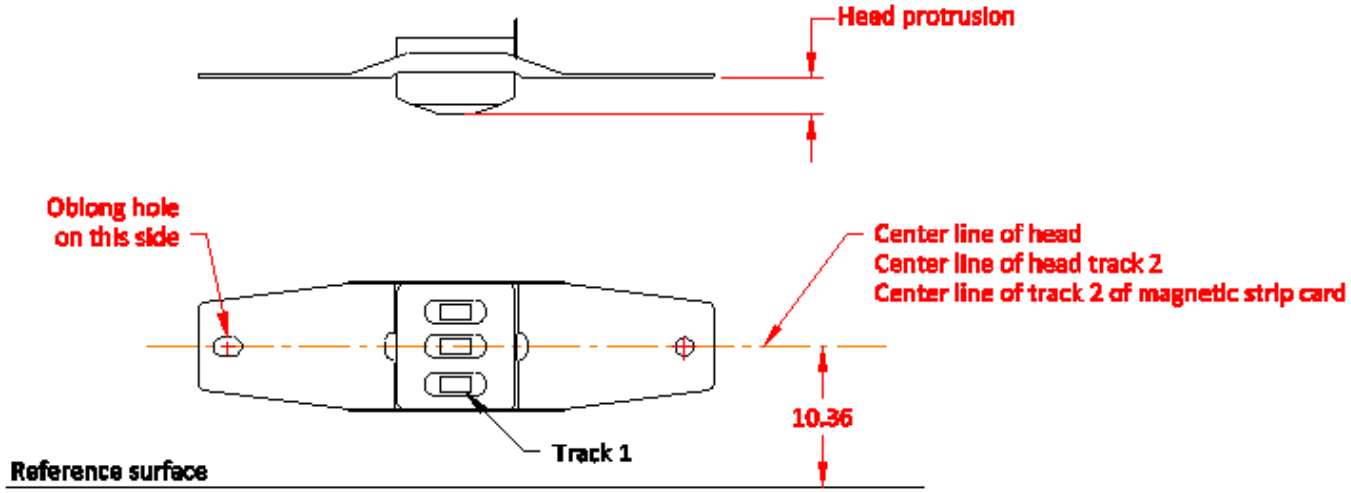
1. ISO 7810 and ISO 7811 standards define the specification for all "standard" magnetic stripe cards. The proper location of each magnetic track's centerline is shown in the figure below (Note: the reference surface for the card is the edge of the card; and it is the surface the card rides on when referring to the magnetic head).



2. The head mounting should allow the head to follow the magnetic stripe on the card. In other words, the magnetic head needs to have the freedom to gimbal (rotate about Track 2's centerline) and move in/out to remain in contact with the surface of the card, after head is assembled to the rail. Below figure shows the rotational and linear movements that the head mounting must allow.



3. The head has to be mounted in relation to the reference surface on which the card slides so that the magnetic tracks of the head are positioned at the same distance from the reference (bottom of slot) as the magnetic tracks on the card (refer to dimensions in #1 above). A typical ID TECH magnetic head with spring is shown below. The mounting holes (centered on Track 2's centerline) in the spring are used for mounting the head and positioning the track locations. (Note: the oblong hole in the spring must be oriented as shown in the drawing to locate tracks 1 through 3 properly)



The center line of head should be parallel to the reference surface.

4. The card thickness must be considered when designing the rail and head mounting. The distance between the head (located on the crown of the head) and opposing wall of card slot must be positioned so that it has a minimum of 0.010 inches (0.25mm) movement when a minimum card thickness is swiped, any less movement could result in unreliable reading. Or put another way: the distance between the crown of the head and the opposing slot wall should be only a fraction of the minimum card thickness that will slide through the reader, so the magnetic head always exerts pressure on the card. The pressure allows for proper contact of the head to stripe especially at high speeds.

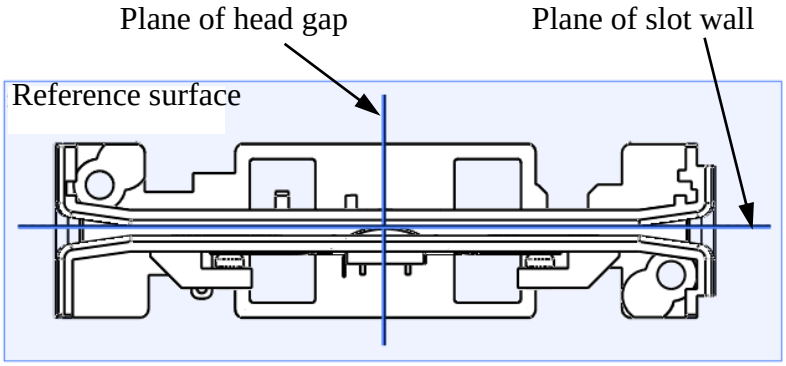
5. Standard card thickness is 0.76mm±10%, if only standard cards are to be used, the rule should be the Apex (crown of head) of the head should be a maximum of 0.25mm from opposing card slot wall. If a thinner or thicker than standard card is used, the distance the head is positioned from the opposing wall needs to be adjusted (this will require a unique rail design with either wider or narrower card slot width).

   The minimum slot width should be maximum card thickness plus 0.15~0.30mm. The suggested minimum slot width is $1.03^{+0.08}_{-0}$ mm when a standard card is used.

6. The design should ensure there is no excessive force or deformation of head spring during the assembly of head to the rail or after head is assembled to prevent permanent deformation of the head

spring. The head spring must be mounted so that it is free to gimbal about the spring holes.

7. The bottom of slot and the slot walls should not have any discontinuities and have to be flat (no deformation is allowed). The portion of the slot wall, about 10mm on each side of the magnetic head's crown, should not have draft and must be perpendicular to the bottom of slot (reference surface). The slot width in lead-in and lead-out area shall be greater and must have gradual transition with no edges, or angles to interfere with the smooth swiping of a card.



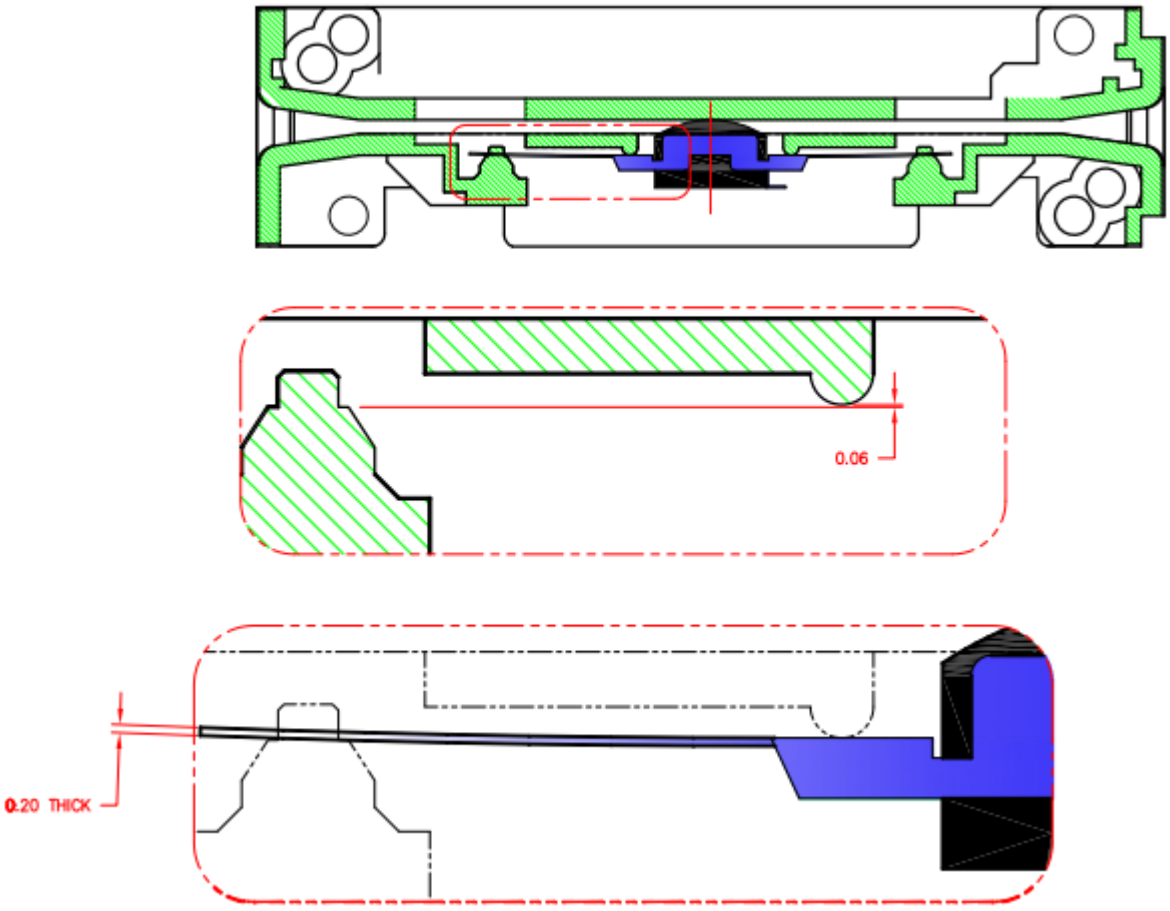
8. Depending on the requirement of swipe life cycles, a suitable material for the rail shall be decided. If the life of the reader is to be greater than 50,000 passes, the bottom of slot must embed a metal wear plate (stainless steel is the metal of preference to avoid corrosion), or the plastic material used for the slot needs to be significantly harder than the card material to ensure adequate rail life.

9. The back side (pin side) of magnetic head shall have enough reserved space to prevent interference with other parts during swiping of maximum thickness cards. The design must provide for a minimum of 1.25~1.52mm space behind the head to allow for proper gimbal and head movement during card swiping. The head opening in the rail must allow room for maximum gimbal action.

10. When the head is installed into the rail, the spring holding the head should be slightly preloaded. Preloading the spring will insure that the head has some stability at the first impact with the card, which is important especially if the card is swiped at high speed. (If the spring is not preloaded it will tend to vibrate when the card impacts the head; vibration would cause head to lose contact with the card.)

    IDTech's solution to preload the head spring is to add 2 symmetrical bumps one on each side of the head (head window), molded into the rail (see drawing bellow). We recommend that the difference between the spring resting surfaces and the crown of the bumps is 0.06+/_0.03 mm, which for a head spring that is 0.20 mm thick will result in a 0.14+/- 0.03 mm bow. The bumps should by cylindrical and their crown parallel with the slot wall opposite to the head crown; this will insure that when the

head is mounted into the rail, its crown will be parallel to the slot surface and will make good contact with the magnetic stripe on the card. Please see the below drawing:

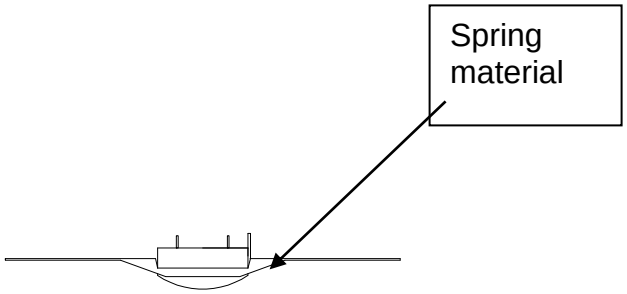


11. The length, width and height of rail's slot will affect the stability of reading performance.
    a. The length of the slot to be maximum permitted by dimensional constraints (if possible it should be 2 times the length of the card).
    b. The slot width to be approx. 0.20 mm bigger than the maximum thickness card that will be swiped through the slot.
    c. The height of the slot should be as big as the dimensional constraints allow, but shall not extend over the embossing area of the card unless there is a provision (recess) in the rail wall design to allow for such embossing.

12. The window in the rail wall through which the head protrudes into the slot should be big enough to allow free movement of the head.

The clearance between the head and the wall of the rail window depends on the amount of head travel and on the head protrusion. (The distance from the crown of the head to the surface of the spring)

For a standard rail with $1.03\ ^{+0.08}_{-0}$ mm wide slot and a standard ID Tech head with 3.50 head protrusion minimum 1.25 mm clearance has to be allowed on all sides of the head.
Note: guideline does not apply when low profile heads are used. The window must allow clearance for the portion of the spring welded to the magnetic head as shown in figure below.



13. ID TECH can provide samples of a rail and magnetic head for design reference. Order these through your local sales representative using the following part numbers:
    90mm rail 80006248-001 and
    Standard wing spring head 80027236-001

APPENDIX J: 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.

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).

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 to wait another 3 seconds after reading the response, then perform the following loading sequence.

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.

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

A. 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
        34 2e 30 34 39 03 17     4.049..
```

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.

C. 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).

D. 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..
```