USER MANUAL

SecureHead™ Encrypted Magnetic Read Head

SPI Interface

80101502-001 Rev M
28 December 2020

ID TECH
10721 Walker Street, Cypress, CA 90630
Voice: (714) 761-6368 Fax: (714) 761-8880
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	Changes	By
A	01/19/2010	Initial Release	JW
A1	02/23/2010	Added fixed key encryption, security level 4, and authentication process	JW
B	06/03/2010	Revised section 2.5 Data Available Output and 2.6 Chip Select. 3.1 setup command structure Modified Appendix A default settings Removed California Driver License card format Added Encrypt External Data command Changed default device key General edits to improve consistency throughout the document	JW
B1	07/21/2010	Added commands for DUKPT key loading Added level 3 and 4 Raw data output format	JW
B2	07/29/2010	Added IDLE character for each response	JW
B3	08/06/2010	Added additional commands • Review Serial Number • Select Key Management Type • Review KSN • Review Security Level Revised check card format Added read head specification and dimension	JW
B4	08/27/2010	Added USB interface support	JW
C	10/25/2010	Separated the SPI and non-SPI interface Revised read status command Added decryption examples	JW
C1	11/12/2010	General revision throughout the document	JW
D	05/06/2011	Added SecureHead mounting option with drawing to indicate Track1 location Edited original and enhanced encryption output format Changed device serial number length from 8 byte to 10 byte	JW
E	05/23/2011	Added explanation for abnormal big current Modified communication timing Revised input voltage range	JW
F	06/21/2011	Added design guidelines for head installation	JW
G	10/10/2011	Support Dual DUKPT Key management, 4.10.2.2	JW
H	11/03/2011	Remove sleep current information	JW
J	07/03/2012	Added design guidelines on preloading the spring Added cable length	JW
K	06/06/2013	Update appendix I magnetic heads mechanical design guidelines Remove the key loading command	CH
L	11/09/2020	Style update Removed Level 4 Encryption examples; Level 4 is no longer supported Updated Appendix I: OEM Rails Installation and User guide	CB
M	12/28/2020	Removed commands related to Fixed Keys.	CB

Table of Contents

1. INTRODUCTION

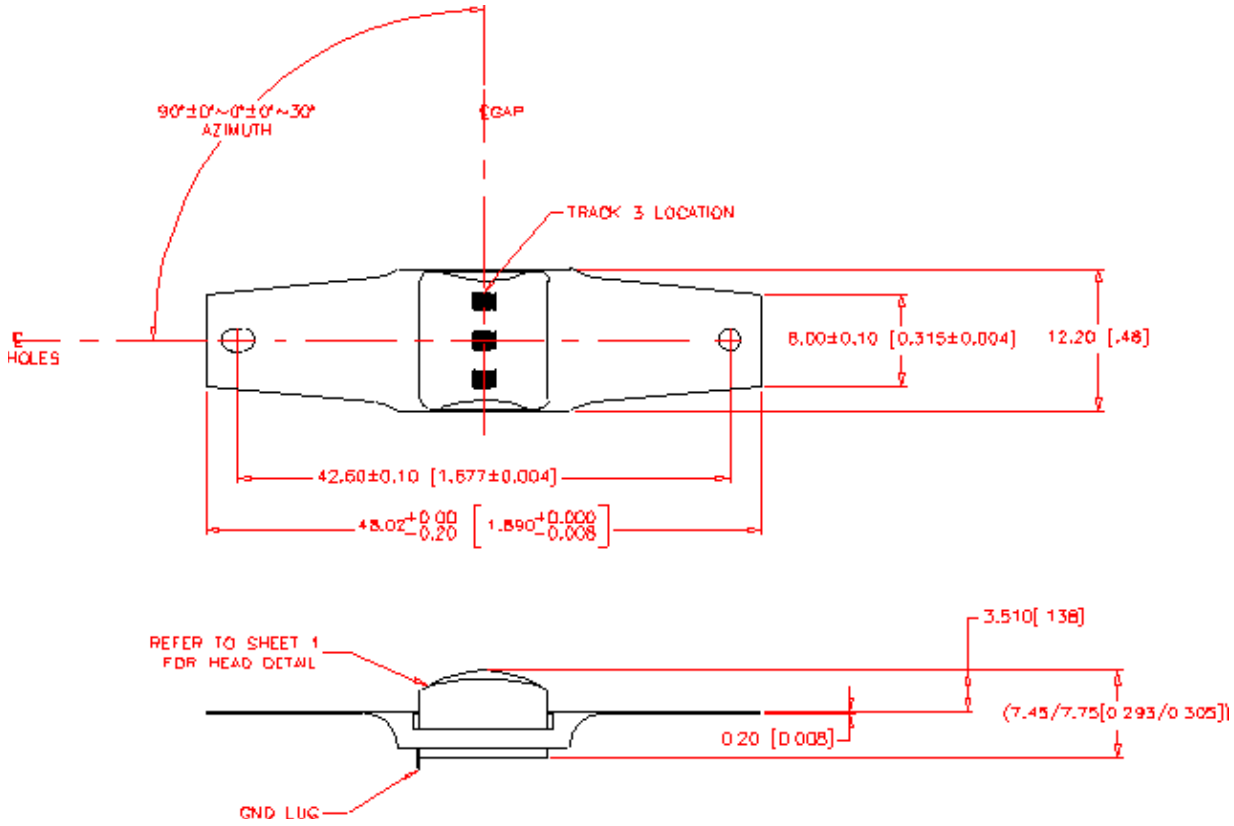
The SPI SecureHead™ magnetic stripe reader can read 1, 2, or 3 tracks of magnetic stripe information. When connected to the host, the SecureHead is completely compatible with SPI Specification. The raw data or decoded data send to host through the SPI.
The SecureHead supports both unencrypted and encrypted data output. When the 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	1.5 mA
ESD	+4kV discharge to head can
Environment	
Operating Temperature	0 ℃to 55 ℃
Storage Temperature	-40 ℃to 70 ℃
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, 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.

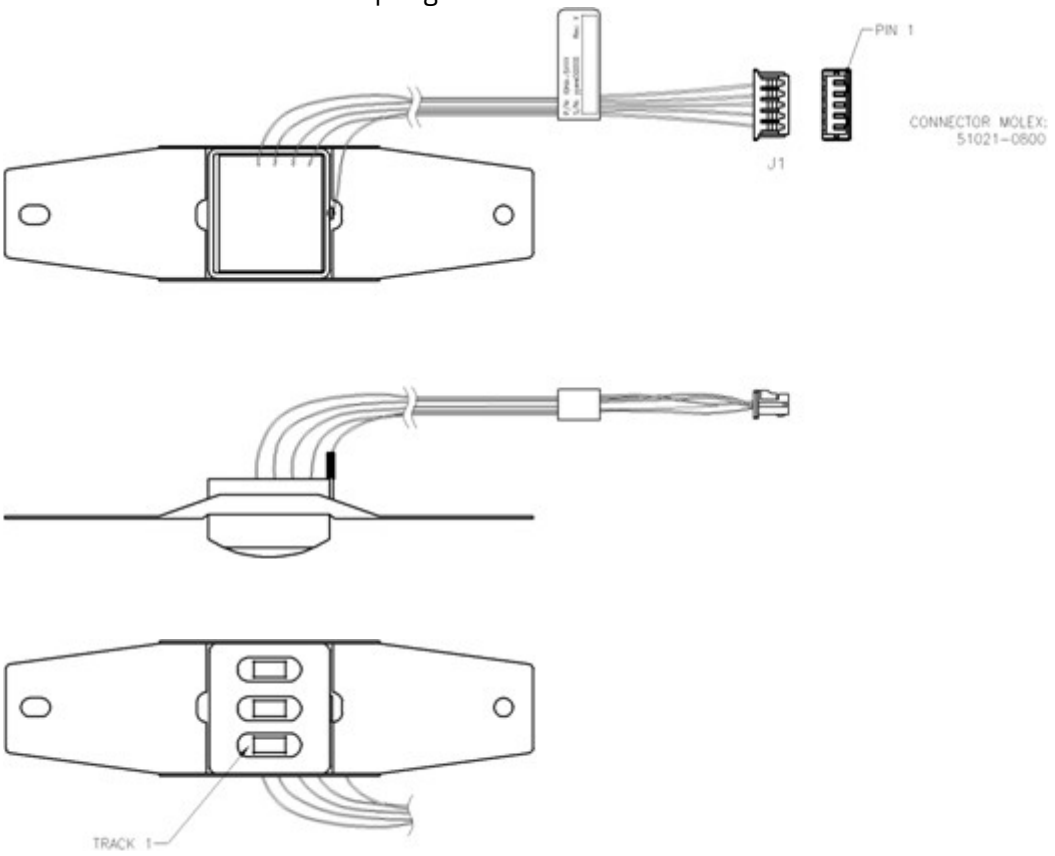
2.1. Dimensions



2.2. Mounting Options

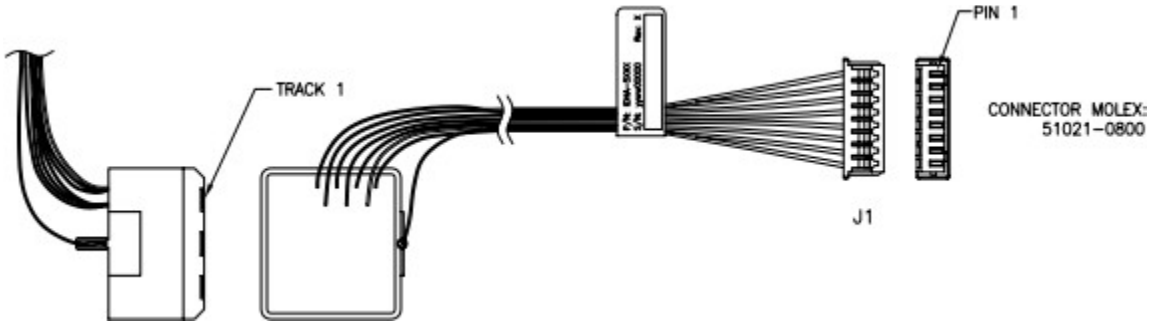
2.2.1. Wing spring mounting

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

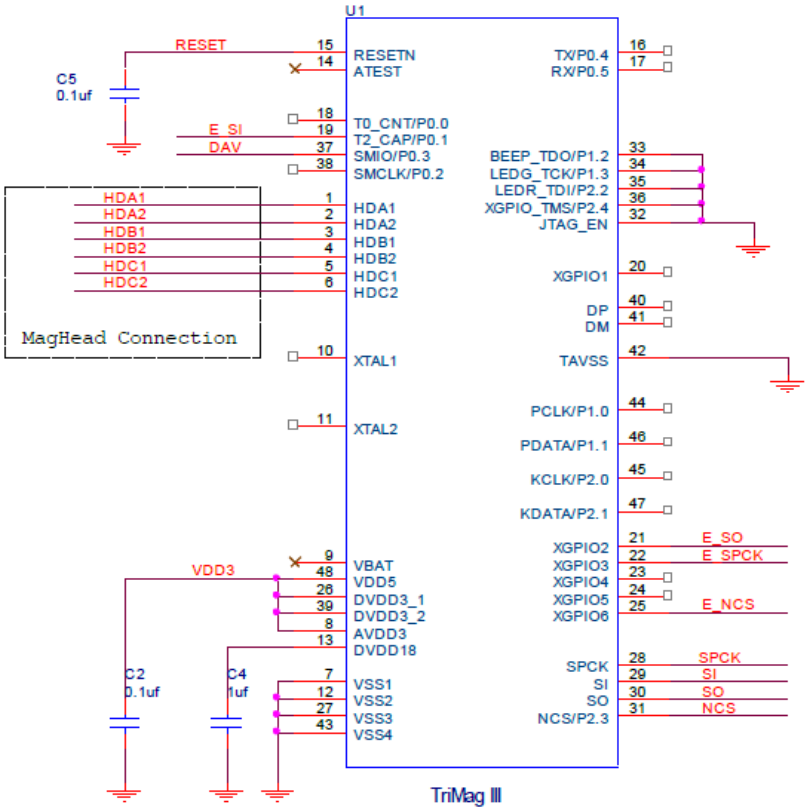
his option is provided for special applications.



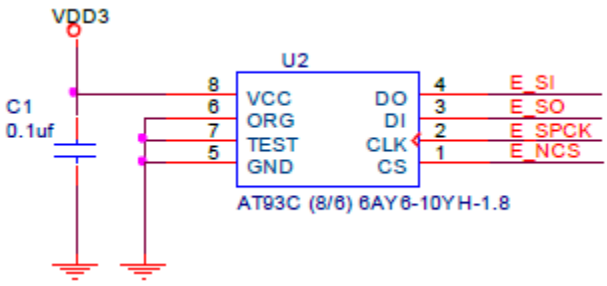
The mechanical interface is an eight-pin male Molex Connector 51021-0800 for option 1 and 2.

2.4. Hardware design (3.3 V system)

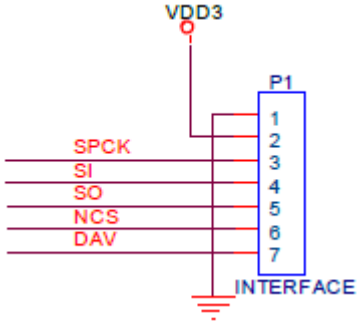
TriMag III



EEPROM



SPI Interface



3. SPI OPERATION

This section describes the SPI (Serial Peripheral Interface), the SPI bus interface timing, communication protocol, timeouts, and data output format. The below table shows the signals used in the SPI interface. The connector is a 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, the data is 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 an 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 would be:

```
A(bit 7) A(bit 6) … A(bit 0) B(bit 7) B(bit 6) … B(bit 0)
```

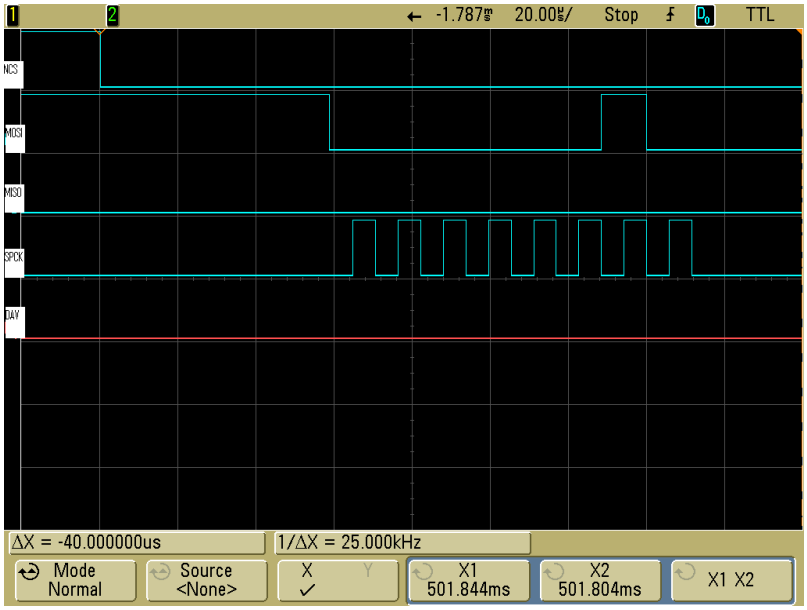
3.2. Clock Polarity and Phase

The clock polarity and phase can be configured with respect to the data. The serial clock input frequency can go up to 400k bps.

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

The signal is required to read card data from the device. See SPI clock phase and polarity property in section 3 for commands to configure clock phase and polarity.The device defaults to clock phase = 0 and clock polarity = 0

The following picture shows an example of clock polarity=0 and clock phase=0. The data is read on the rising edge of the clock and is changed on the falling edge. On MOSI line, the host sends out data 00000010, which is 02h.



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 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.

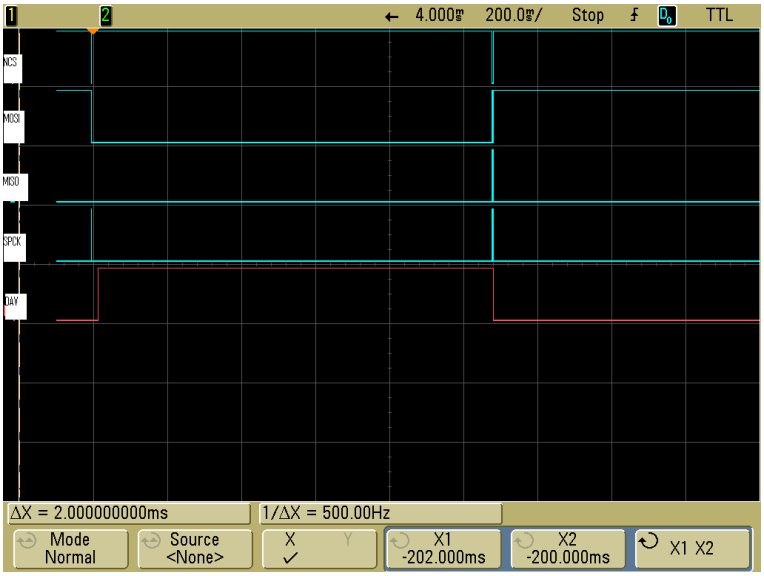
3.5. Data Available Output (DAV)

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 would set 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 reset as it would be in an indeterminate state.

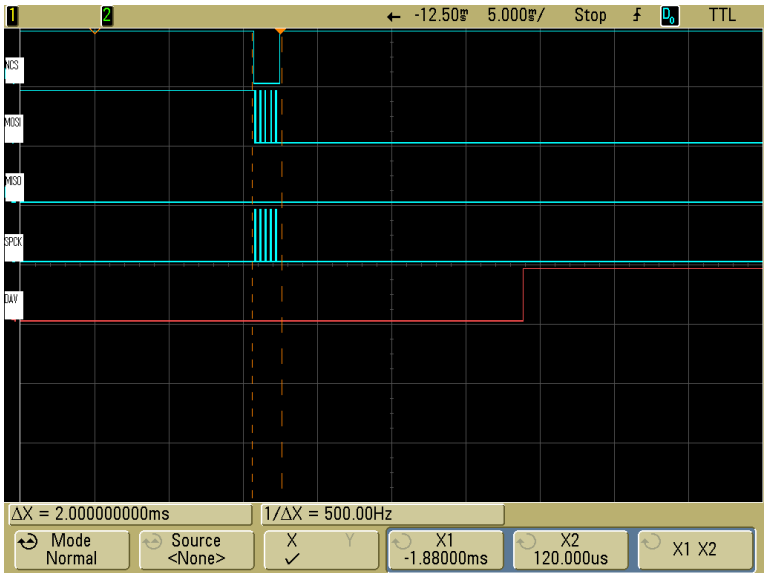
In the case when the DAV signal is not used, the host needs to poll the device periodically to determine if it has data to transmit. The host needs to toggle SCL to get card data from MISO. The first non-IDLE byte indicates the start of valid card data. IDLE is FF. For more details, please refer to the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

The last signal shown in the below graph is the DAV signal:

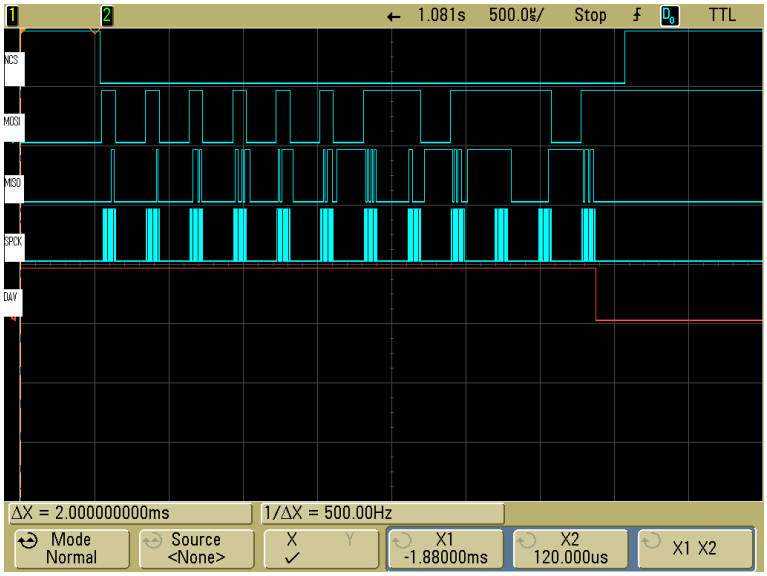


After the command is received and the response is ready, the DAV would be set to high for the host to receive response. After the response is received, the DAV would be low, indicating there is no more data to be transmitted.

About 20ms after receiving command, the response is ready and the DAV set to high. For some specific commands, the delay will be longer.



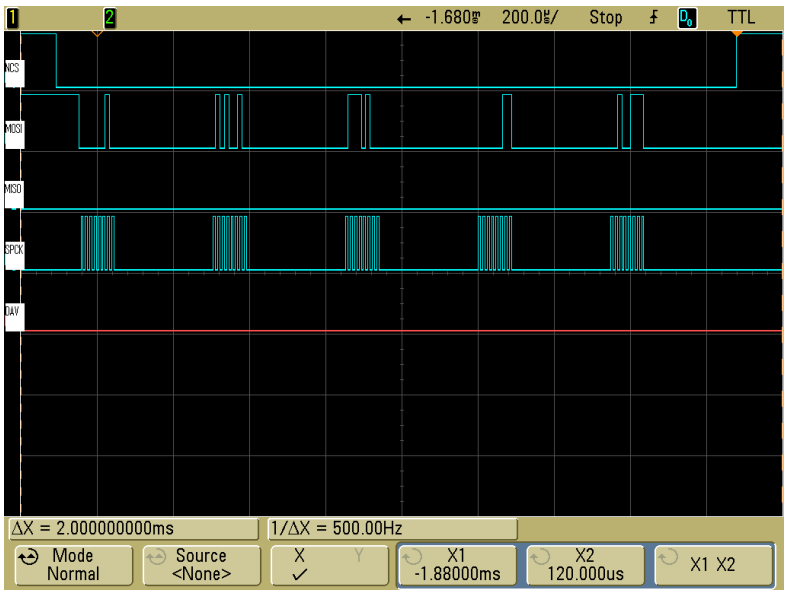
About 20us after the last byte of response is read out by host, DAV is pulled low. So if user polls DAV status to check whether there are data available, we suggest using 100us polling interval.



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 would only respond to SPCK and MOSI signals after a NCS is pulled low. NCS needs to be low the whole time when the host is communicating with the particular device. The Chip Select signal should be connected to ground if is not used by the host. Special case: in the situation of clock phase = 1, NCS needs to generate a falling edge for each command.

The NCS is pulled low when the host is communicating with the device.



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 simply clocks it out. When the device has a frame to send, it raises its data available (DAV) signal and waits for the host to 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 1 byte clocked out from the device after the DAV signal is asserted could be IDLE bytes 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 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. After programmed, these configuration settings are stored in the reader's non-volatile memory (so they are not affected by the cycling of power).

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	

4.1.3. 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 SecureHead has a 50ms start up period, during this period, it doesn't support communication and 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 process, 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 a minimum delay between two commands.

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

If the noise of working environment interferences the SPI communication, to the level that SecureHead can't receive commands correctly, SecureHead will restart to reduce the impact. Restart takes 600ms. 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 **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.

4.4.1. SPI Clock Phase and Polarity Settings

The clock phase and polarity of SPI interface can be adjusted. Both the host and device must be set to the same SPI setting to communicate correctly.

```
<STX><S><79h><01h><SPI Settings><ETX><CheckSum>
```

SPI Settings:
0: Clock phase = 0 and Polarity = 0. Data is read on the clock's rising edge and data is changed on a falling edge
1: Clock phase = 0 and Polarity = 1. Data is read on clock's falling edge and data is changed on a rising edge.
2: Clock phase = 1 and Polarity = 0. Data is read on clock's falling edge and data is changed on a rising edge.
'3' Clock phase = 1 and Polarity =1. Data is read on clock's rising edge and data is changed on a falling edge.

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 is 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**

4.7. 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, send 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 starts to being collected when the first 1 bit is detected. No checking is done except to verify track has or does not have magnetic data.

4.8. 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>

4.9. Review Firmware Version

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

This command is to get the device's firmware version.

4.10. Review Serial Number

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

This command is to get the device's 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.

```
<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.

<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.

<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.

<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.

```
<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)

4.12.1. 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 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.

4.13. 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.

4.13.1. 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.)

<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: Do not send start/end sentinel and send account # on Track2 3: Send start/end sentinel and send account number on Track2

4.14. Security Settings

4.14.1. 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)

4.14.2. Send External Authenticate Command Host -> Device:

<STX><S><74h><08h><8 bytes of original random data><ETX><CheckSum>

Device -> Host:
<ACK> (success)
<NAK> (fail)

4.14.3. 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 (Not for Raw Data Decoding in Both Directions, send out in other mode).

4.15. Review KSN (DUKPT Key management only)

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

This command is to get DUKPT key serial number and counter.

4.16. Review Security Level

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

This command is to get the current security level.

4.17. 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 and 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.

4.18. Encrypted Output for Decoded Data

4.18.1. Encrypt Functions

When a card is swiped through the Reader, the track data will be TDEA (Triple Data Encryption Algorithm, aka, Triple DES) or AES (Advanced Encryption Standard) encrypted 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 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.

4.18.2. 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 written one time. 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 bytes string which contains any hex data. This filed is used by the host to uniquely identify the present transaction. Its primary purpose is to prevent replays. It is only be used at Security Level 4. After a card is read, the Session ID will be encrypted, along with the card data, a supplied as part of the transaction message. The clear text 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

4.18.3. 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 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 before encrypted transactions can take place. The keys are to be injected by certified key injection facility.

There are five security levels available when using the 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 lifetime of DUKPT keys is 1 million. After the key's end of lifetime is

reached, user should inject DUKPT keys again before doing any more transactions.

- 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 clear text data are sent out. Users can select the data masking of the PAN area; the encrypted data format cannot be modified. Users can choose whether to send hashed data and whether to reveal the card expiration date. When the encryption is turned on, level 3 is the default security level.

- Level 4: When the reader is at Security Level 4, a correctly executed Authentication Sequence is required before the reader sends out data for each card swipe.

4.18.4. 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 in security level 3, for the encrypted fields, the original data is encrypted using the TDES/AES CBC mode with an Initialization Vector starting at all binary zeroes and the Encryption Key associated with the current DUKPT KSN.

4.18.5. 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 Track2 and the position of '=' is between 13th ~ 20th character.
- Total length of Track2 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)

4.18.6. MSR Data Masking

For cards need to be encrypted, both encrypted data and 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.

4.18.7. Level 1 and 2 Data Output Format

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

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

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 which separates data tracks.
- **Terminator:** A designated character which comes at the end of the last track of data, to separate card reads.

4.18.8. DUKPT Level 3 Data Output Original Format

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

A card swipe returns the following data
Card data is sent out in format of
<STX><LenL><LenH><Card Data><CheckLRC><CheckSum><ETX>

<STX> = 02h, <ETX> = 03h
<LenL><LenH> is a two byte length of <Card Data>.
<CheckLRC> is a one byte Exclusive-OR sum calculated for all <Card Data>.
<CheckSum> is a one byte Sum value calculated for all <Card data>.
<Card Data> card data format is shown below.

ISO/ABA Data Output Format:

- card encoding type (0: ISO/ABA, 4: for Raw Mode)
- track status (bit 0,1,2:T1,2,3 decode, bit 3,4,5:T1,2,3 sampling)
- Track1 unencrypted length (1 byte, 0 for no track1 data)
- Track2 unencrypted length (1 byte, 0 for no track2 data)
- Track3 unencrypted length (1 byte, 0 for no track3 data)
- Track1 masked (Omitted if in Raw mode)
- Track2 masked (Omitted if in Raw mode)
- Track3 data (Omitted if in Raw mode)
- Track1 encrypted (AES/TDES encrypted data)
- Track2 encrypted (AES/TDES encrypted data)
- Track3 encrypted (Only used in Raw mode)
- Track1 hashed (20 bytes SHA1-Xor)
- Track2 hashed (20 bytes SHA1-Xor)

- DUKPT serial number (10 bytes)

Non-ISO/ABA Data Output Format

- Card encoding type (1: AAMVA, 3: Others)
- Track status (bit 0,1,2:T1,2,3 decode, bit 3,4,5:T1,2,3 sampling)
- Track1 length (1 byte, 0 for no track1 data)
- Track2 length (1 byte, 0 for no track2 data)
- Track3 length (1 byte, 0 for no track3 data)
- Track1 data
- Track2 data
- Track3 data

Description:

Track1, Track2, and Track3 Unencrypted Length

This one-byte value is the length of the original Track data. It indicates the number of bytes in the Track masked data field. It should be used to separate Track1, Track2, and Track3data after decrypting Track encrypted data field.

Track3 Unencrypted Length

This one-byte value indicates the number of bytes in Track3 data field.

Track1 and Track2 Masked

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).

Track1, Track2, and Track3 Encrypted

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

Encrypted Data Length

Track1 and Track2 data are encrypted as a single block. To get the number of bytes for encrypted data field, we need to get Track1 and Track2 unencrypted length first. The field length is always a multiple of 8 bytes for TDES or multiple of 16 bytes for AES. This value will be zero if there was no data on both tracks or if there was an error decoding both tracks.

After the encrypted data is decrypted, all padding 0 need to be removed. The number of bytes of decoded Track1 data is indicated by Track1 unencrypted length field. The remaining bytes are Track2 data, the length of which is indicated by Track2 unencrypted length filed.

Track1 and Track2 Hashed
SecureHead reader uses SHA-1 to generate hashed data for both Track1 and Track2 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.

4.18.9. Fixed Key Level 3 Data Output Original Format

Same as 4.14.8 DUKPT Level 3 Data Output Original Format, only change <DUKPT serial number> to <device serial number> plus two NULL bytes.

4.18.10. DUKPT Level 3 Data Output Enhanced Format

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.

1. Encryption Output Format Setting: Command: 53 85 01 <Encryption Format>
   Encryption Format:
   0:: Original Encryption Format
   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 – Track1 force encrypt bit1: 1 – Track2 force encrypt
   bit2: 1 – Track3 force encrypt
   bit3: 1 – Track3 force encrypt when card type is 0

Note:

- When force encrypt is set, this track will always be encrypted, regardless of card type. No clear/mask text will be sent.
- If and only if in enhanced encryption format, each track is encrypted separately. Encrypted data length will round up to 8 or 16 bytes.
- When force encrypt is not set, the data will be encrypted in original encryption format, that is, only Track1 and Track2 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.

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 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 (8 bytes) (Security level 4 only)
- 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

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: 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, 7: Reserved for future use

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: 0 Reserved for future use
Bit 4: 0 Reserved for future use
Bit 5: 0 Reserved for future use

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

4.18.11. Fix 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.

4.18.12. Level 4 Activate Authentication Sequence

The security level changes from 3 to 4 when the device enters authentication mode successfully. After the security level is changed to level 3 or 4, it cannot go back to a lower level.

4.18.13. Activate Authentication Mode Command

When the reader is in security level 4, it would only transmit the card data when it is in Authenticated Mode.

Authentication Mode Request
When sending the authentication request, the user also needs to specify a time limit for the reader to wait for the activation challenge reply command. The minimum timeout duration required is 120 seconds. If the specified time is less than the minimum, 120 seconds would be used for timeout duration. The maximum time allowed is 3600 seconds (one hour). If the reader times out while waiting for the activation challenge reply, the authentication failed.

Device Response
When authentication mode is requested, the device responds with two challenges: Challenge 1 and challenge 2. The challenges are encrypted using the current DUKPT key exclusive-or'ed with <F0F0 F0F0 F0F0 F0F0 F0F0 F0F0 F0F0 F0F0>.

The decrypted challenge 1 contains 6 bytes of random number followed by the last two bytes of KSN. The two bytes of KSN may be compared with the last two bytes of the clear text KSN sent in the message to authenticate the reader. The user should complete the Activate Authentication sequence using Activation Challenge Reply command.

Command Structure Host -> Device:
<STX><R><80h><02h><Pre-Authentication Time Limit><ETX><CheckSum>

Device -> Host:
<ACK><STX><Device Response Data><ETX><CheckSum> (success)
<NAK> (fail)

Pre-Authentication Time Limit: 2 bytes of time in seconds
Device Response Data: 26 bytes data, consists of <Current Key Serial Number> <Challenge 1> <Challenge 2>

Current Key Serial Number: 10 bytes data with Initial Key Serial Number in the leftmost 59 bits and Encryption Counter in the rightmost 21 bits.
Challenge 1: 8 bytes challenge used to activate authentication. Encrypted using the key derived from the current DUKPT key.
Challenge 2: 8 bytes challenge used to deactivate authentication. Encrypted using the key derived from the current DUKPT key.

Activation Challenge Reply Command

This command serves as the second part of an Activate Authentication sequence. The host sends the first 6 bytes of Challenge 1 from the response of Activate Authenticated Mode command, two bytes of Authenticated mode timeout duration, and eight bytes Session ID encrypted with the result of current DUKPT Key exclusive- or'ed with <3C3C 3C3C 3C3C 3C3C 3C3C 3C3C 3C3C 3C3C>.

The Authenticated mode timeout duration specifies the maximum time in seconds which the reader would remain in Authenticated Mode. A value of zero forces the reader to stay in Authenticated Mode until a card swipe or power down occurs. The minimum timeout duration required is 120 seconds. If the specified time is less than the minimum, 120 seconds would be used for timeout duration. The maximum time allowed is 3600 seconds (one hour).

Session ID information is included. If the command is successful, the Session ID will be changed.
The Activate Authenticated Mode succeeds if the device decrypts Challenge Reply response correctly. If the device cannot decrypt Challenge Reply command, Activate Authenticated Mode fails and DUKPT KSN advances.

Command Structure Host -> Device:
<STX><S><82h><10h><Activation Data><ETX><CheckSum>

Device -> Host:
<ACK> (success)
<NAK> (fail)

Activation Data: 16 bytes, structured as <Challenge 1 Response> <Session ID>

Challenge 1 Response: 6 bytes of Challenge 1 random data with 2 bytes of Authenticated mode timeout duration. It's encrypted using the key derived from the current DUKPT key.
Session ID: 8 bytes Session ID, encrypted using the key derived from the current DUKPT key.

Deactivate Authenticated Mode Command

This command is used to exit Authenticated Mode. Host needs to send the first 7 bytes of Challenge 2 (from the response of Activate Authenticated Mode command) and the Increment Flag (00h indicates no increment, 01h indicates increment of the KSN) encrypted with current DUKPT Key exclusive- or'ed with <3C3C 3C3C 3C3C 3C3C 3C3C 3C3C 3C3C 3C3C>.

If device decrypts Challenge 2 successfully, the device will exit Authenticated Mode. The KSN will increase if the Increment flag is set to 01h. If device cannot decrypt Challenge 2 successfully, it will stay in Authenticated Mode until timeout occurs or when customer swipes a card.

The KSN is incremented every time the authenticated mode is exited by timeout or card swipe action. When the authenticated mode is exited by Deactivate Authenticated Mode command, the KSN will increment when the increment flag is set to 01h.

Command Structure Host -> Device:
<STX><S><81h><08h for TDES or 10h for AES><Deactivation Data><ETX><CheckSum>

Device -> Host:
<ACK> (success)
<NAK> (fail)

<Deactivation data>: 8-bytes response to Challenge 2. It contains 7 bytes of Challenge 2 with 1 byte of Increment Flag, encrypted by the specified variant of current DUKPT Key

4.18.14. Get Reader Status Command

Command Structure Host -> Device:
<STX><R><83h><ETX><CheckSum>

Device -> Host:
<ACK><STX><83h><02h><Current Reader Status><Pre-conditon><ETX><CheckSum> (success)
<NAK> (fail)

Current Reader Status: 2-bytes data with one byte of <Reader State> and one byte of <Pre-Condition>

Reader State: indicates the current state of the reader

00h: The reader is waiting for Activate Authentication Mode Command. The command must be sent before the card can be read.

01h: The authentication request has been sent, the reader is waiting for the Activation Challenge Reply Command.

02h: The reader is waiting for a card swipe.

Pre-condition: specifies how the reader goes to its current state as follows

00h: The reader has no card swipes and has not been authenticated since it was powered up.

01h: Authentication Mode was activated successfully. The reader processed a valid Activation Challenge Reply command.

02h: The reader receives a good card swipe.

03h: The reader receives a bad card swipe or the card is invalid. 04h: Authentication Activation Failed.

05h: Authentication Deactivation Failed.

06h: Authentication Activation Timed Out. The Host fails to send an Activation Challenge Reply command within the time specified in the Activate Authentication Mode command.

07h: Swipe Timed Out. The user fails to swipe a card within the time specified in the Activation Challenge Reply command.

4.19. Other Command Protocol Settings

4.19.1. SPI Clock Phase and Polarity Settings

The clock phase and polarity of SPI interface can be adjusted. Both the host and device must be set to the same SPI setting in order to communicate correctly.

Command:
```
01 00 04 00 01 08 <SPI Clock Setting>
```

Refer General Selections for <SPI Clock Setting>, valid data is from 00h to 03h.

Command Response:
```
01 00 02 01 00
```

4.19.2. Set/Get Device Number

Set/Get eight byte device serial number.

Command:
Set Device Serial Number: `01 00 0B 00 01 01 <8 bytes of Device Serial Number>`
Get Device Serial Number: `01 00 03 00 00 01`

Command Response:
Set Device Serial Number: `01 00 02 01 00`
Get Device Serial Number: `01 00 0A 01 00 < 8 bytes of Device Serial Number >`

4.19.3. Enable/Disable Encryption

Enable or Disable the SecureHead Encryption output in other mode (non-ID TECH protocol). If encryption is disabled, original data will be sent out to the host. If it enabled, encrypted data will be send out to the host

Command:
`01 00 04 00 01 02 01` Enable Encryption
`01 00 04 00 01 02 00` Disable Encryption

Command Response: `01 00 02 01 00`

4.19.4. Get Challenge

Host gets 8 bytes random number from SecureHead in order to do external authentication.

Command:
01 00 03 00 00 04

Command Response:
`01 00 0A 01 00` <8 bytes of Challenge Data>

4.19.5. External Authenticate

SecureHead will use this command to authenticate the host by comparing the decrypted data from the host with its random data.

Command Format:
`01 00 06 00 05` <First four bytes of decrypted random data from Get Challenge>

Command Response:
`01 00 02 01 00` Success
`01 00 02 01 01` Fail

4.19.6. Load Security Key

The sixteen bytes key is used encryption, and its default value is `0000 0000 0000 0000 0000 0000 0000 0000`.

For security purpose, key injection only allowed after successful external authentication, and will be loaded by two components each with 16 bytes of key.
Those two components will be XORed to generate key for encryption.

Command Format:
`01 00 13 00 04 01` <16 bytes of First Key Component>
`01 00 13 00 04 02` <16 bytes of Second Key Component>

Command Response:
`01 00 02 01 00`

5. Appendix A: Default Setting Table

5.1. 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

6. Appendix B: Magnetic Stripe Standard Formats

6.1. ISO Credit Card Format

ISO standards for International Standards Organization 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

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

6.2. AAMVA Driver's License Format

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

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

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

7. Appendix C: Other Mode Card Data Output

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

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

8. Appendix D: Guide to Encrypting and Decrypting Data

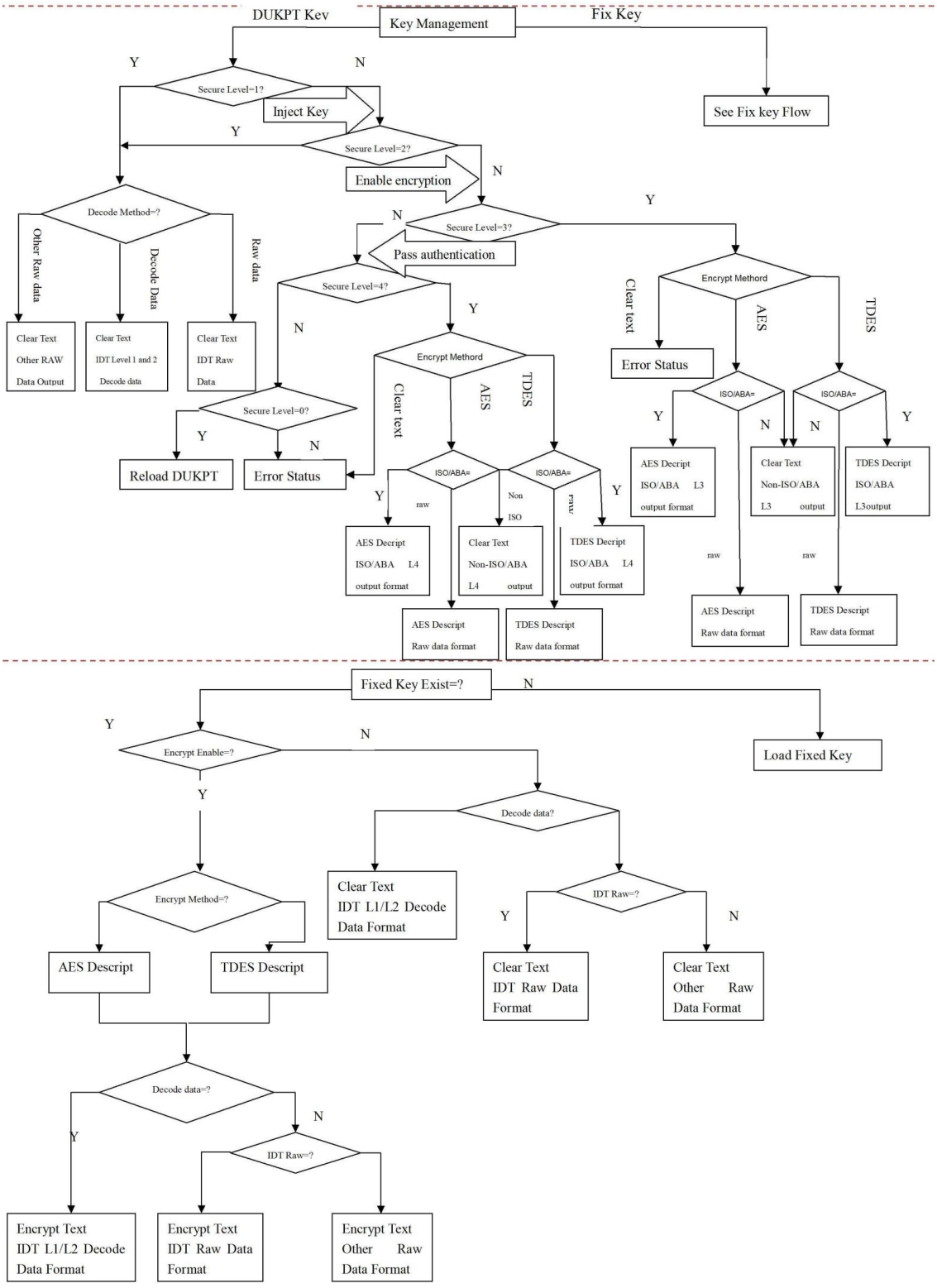
The encryption method used by SecureHead is called Cipher-block Chaining (CBC). With this method, each block of data is XOR'ed with the previous data block before being encrypted. The encryption of each block depends on all the previous blocks. As a result, each encrypted data block would need to be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

To encrypt the data, first generate an 8-byte random 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.

9. Appendix E: Key Management Flow Chart



10. Appendix F: Example of Decoded Data Decryption

Key for all examples is `0123456789ABCDEFFEDCBA9876543210`.

10.1. Security Level 3 Decryption: Original Encryption Format

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

Original encryption structure (if the structure starts original encryption format (this can be recognized because the high bit of the fourth byte (00) is 0).

```
027D01003F48236B252A343236362A2A2A2A2A2A2A393939395E42555348204A522F
47454F52474520572E4D525E2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A
2A2A2A2A2A2A2A2A3F2A3B343236362A2A2A2A2A2A2A393939393D2A2A2A2A2A2A2A
2A2A2A2A2A2A2A2A3F2A3B333333333333333333333736373637363037303730373736
373637363333333333333333333337363736373630373037303737363736373633333333
333333333333333736373637363037303730373737363736373633333333333333333333
373637363736303730373F32863E9E3DA28E455B28F7736B77E47A64EDDA3BF03A06E4
4F31D1818C0BCD7A353FB1AD70EFD30FFC3DA08A4FBC9372E57E8B40848BAEAA3FE724
B3550E2F4B223E6BF264BEAE9E39142B648CDB51FB8DAF8EA5B63913D29419B67582FC
CCE9B372660F03668CC453216D9449C6B67EF33418AC88F65E1DB7ED4D10973F99DFC8
463FF6DF113B6226C4898A9D355057ECAF11A5598F02CA3162994901190000000001399F03
```

STX, Length (LSB, MSB), card type, track status, length Track1, length Track2, length Track3: `02 7D 01 00 3F 48 23 6B`

The above broken down and interpreted

02: STX character
7D: Low byte of total length
01: High byte of total length
00: Card type byte (interpretation old format ABA card)
3F: Three tracks of data all good
48: Length of Track1
23: Length of Track2
6B: Length of Track3

Track1 data masked (length 0x48):

```
252A343236362A2A2A2A2A2A2A393939395E42555348204A522F47454F5247452057
2E4D525E2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A
3F2A
```

Track2 data in hex masked (length 0x23):

```
3B343236362A2A2A2A2A2A2A393939393D2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A3F2A
```

Track3 data unencrypted (length 0x6B):
3B3333333333333333333373637363736303730373037373637363736333333333333
3333333373637363736303730373037373637363736333333333333333333333373637
3637363037303730373736373637363333333333333333333337363736373630373037
3F32

Track1 & 2 encrypted length 0x48+0x23 rounded up to 8 bytes =0x6B -> 0x70 (112 decimal):
863E9E3DA28E455B28F7736B77E47A64EDDA3BF03A06E44F31D1818C0BCD7A35
3FB1AD70EFD30FFC3DA08A4FBC9372E57E8B40848BAEAA3FE724B3550E2F4B22
3E6BF264BEAE9E39142B648CDB51FB8DAF8EA5B63913D29419B67582FCCCE9B3
72660F03668CC453216D9449C6B67EF3

Track1 hashed: 3418AC88F65E1DB7ED4D10973F99DFC8463FF6DF

Track2 hashed: 113B6226C4898A9D355057ECAF11A5598F02CA31

KSN: 62994901190000000001

LRC, checksum and ETX: 39 9F 03

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

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

Track3 data unencrypted in ASCII:
;333333333767676070707767676333333333376767607070776767633333333337676
76760707077676763333333333767676070?2

Key Value: F8 2A 7A 0D 7C 67 46 F1 96 18 9A FB 54 2C 65 A3

KSN: 62 99 49 01 19 00 00 00 00 01

Decrypted Data in ASCII:
%B4266841088889999^BUSH JR/GEORGE
W.MR^080910110000110000000046000000?!;4266841088889999=08091011000004
6?0;333333333767676070707767676333333333376767607070776767633333333333
76767607070776767633333333376767607070?2

Decrypted Data in Hex:
2542343236363834313038383838393939395E42555348204A522F47454F5247452057
2E4D525E303830393130313130303030313130303030303030303436303030303030
3F213B343236363834313038383838393939393D303830393130313130303030303436
3F300000000000

10.2. 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)
3B343236362A2A2A2A2A2A2A393939393D2A2A2A2A2A2A2A2A2A2A2A2A2A2A2A3F2A

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^08091011000011000000000046000000?!
```

Track2 decrypted

```
;4266841088889999=0809101100000046?0
```

Track3 decrypted

```
;33333333376767607070776767633333333337676760707077676763333333333767
67607070776767633333333337676760707?2
```

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

```
2542343236363834313038383838393939395E42555348204A522F47454F5247452057
2E4D525E303830393130313130303030313130303030303030303034363030303030303
3F21
```

Track2 decrypted data in hex including padding zeros

```
3B343236363834313038383838393939393D303830393130313130303030303034363F30
0000000000
```

Track3 decrypted data in hex including padding zeros

```
3B333333333333333333333736373637363037303730373736373637363633333333333333
333333337363736373630373037303730373736373637363633333333333333333333373637
363736303730373037373637363736373633333333333333333333373637363736303730373
3F320000000000
```

11. Appendix G: Example of ID TECH Raw Data Decryption

Original Raw Data Forward Direction:

```
01D67C81020408102D448102040810204289 0A350854A2FB3EE4BA3D4065B67A9C391F
582A42B99A858A90AF60852B14AA628A0D028FC210842C1842108403009 2040B51581F
24B56074404811160D
```

Original Raw Data Backward Direction:

```
01A28CAA51A9420DEA12A342B33A84A835F13872BCDB4C0578BA4EF9BE8A542158A122
840810204081024568102040810 27CD60D
02D11024045C0D5A49F03515A0409201804210843068421087E20D
```

Note:

1. There is track number before each track. Track1 is 01, Track2 is 02, Track3 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`

11.1. Original Encryption Format

Original encryption format (this can be recognized because the high bit of the fourth byte underlined (00) is 0.

```
028700041B331A0027D2E435CEE303F007E977B598B7E3C57C76F4445E309F6916C032
1A0F915B6E490813498839049FE5204762327C3C758C5BF82542DEEDD8D6AF88019149
A702FF2D43BD4AD60031FA450720B00D7808E15F3D5B29AE712C64A1212E9AF6F483BD407
98A9FF2DDE77D046620B55BCE94A4D5534CF57E7E07629949011A0000000001871D03
```

STX, Length (LSB, MSB), card type, track status, length Track1, length Track2, length Track3 02 8700 04 1B 33 1A 00

Track1 & 2 encrypted length 0x33+0x1A rounded up to 8 bytes =0x4D -> 0x50 (80 decimal)

```
27D2E435CEE303F007E977B598B7E3C57C76F4445E309F6916C0321A0F915B6E490813
498839049FE5204762327C3C758C5BF82542DEEDD8D6AF88019149A702FF2D43BD4AD6
0031FA450720B00 D7808
```

Track1 hashed `E15F3D5B29AE712C64A1212E9AF6F483BD40798A`

Track2 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 0A350854A2FB3EE4BA3D4065B67A9C391F
582A42B99A858A90AF60852B14AA628A028FC210842C1842108403009 2040B51581F24
B5607440481116

12. Appendix H: Example of SPI Master Chip Control

```
/*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 #defineSPI_RATIO_16       0x03 // FCLK PERIPH/16
#defineSPI_RATIO_32         0x80 // FCLK PERIPH/32
#defineSPI_RATIO_64         0x81 // FCLK PERIPH/64#defineSPI_RATIO_128    0x82 // FCLK PERIPH/128 #define
SPI_RATIO_INVALID0x83 // 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
/*_______DEC LARATION_*/


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

}


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.

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]);
}
}

{


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

}

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.


communication. command string.
```

```
chip select pin to idle high.

}

{

tasks
}
delay10us(); // Wait for high level get steady.
_SPI_SS = 0; // Pull chip select pin low, ready to start SPI


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

_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 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 endflag
```

```
m = 4;        // Delay 40us then poll for DAV pin status or send out next byte. do{
delay10us();
}while(m--)
```

13. Appendix I: 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).

13.1. 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 Track2; 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 Track2 centerline, ensuring the read head will be to the proper dimensions for all tracks.

Figure 1: Location of Magnetic Track Centerlines

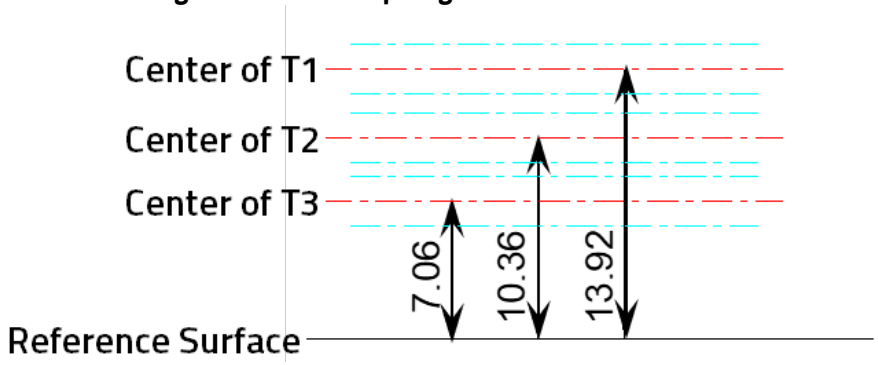
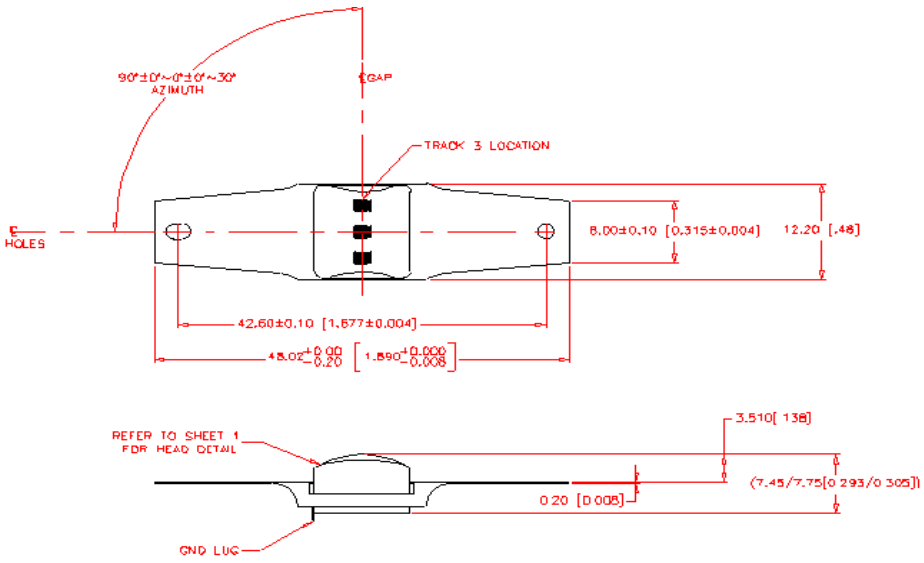


Figure 2: Standard Magnetic head assembly showing tolerances of Azimuth

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



13.2. 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.

13.3. 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.

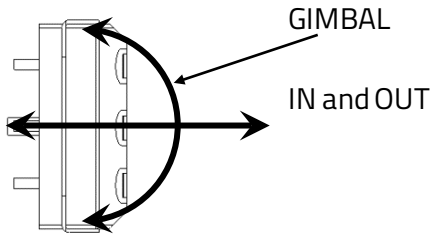
1. Magnetic heads need to be able to rotate on a gimbal (refer to **Figure 3** for an example) to compensate for tilting cards, and therefore must have a minimum deflection. For credit/debit cards used in financial transactions or other applications using standard thickness cards (0.030 inches +/- 10%), 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.