Specification No. 3-1308-8837-Z1000

EICM-1000

SMART CARD IC CARD READER MODULE
(Rev 1.00)

For product information, application engineering assistance or pricing, contact us at:

Peripheral Dynamics Inc.
5150 Campus Drive
Plymouth Meeting, PA 19462-1123
Toll-Free: 800-523-0253
Phone: 610-825-7090 Fax: 610-834-7708
E-mail: sales@pdiscan.com Internet: www.pdiscan.com

Specification No. 3-1308-8837-Z1000

June 24, 2013

CONTENTS

Specification No. 3-1308-8837-Z1000

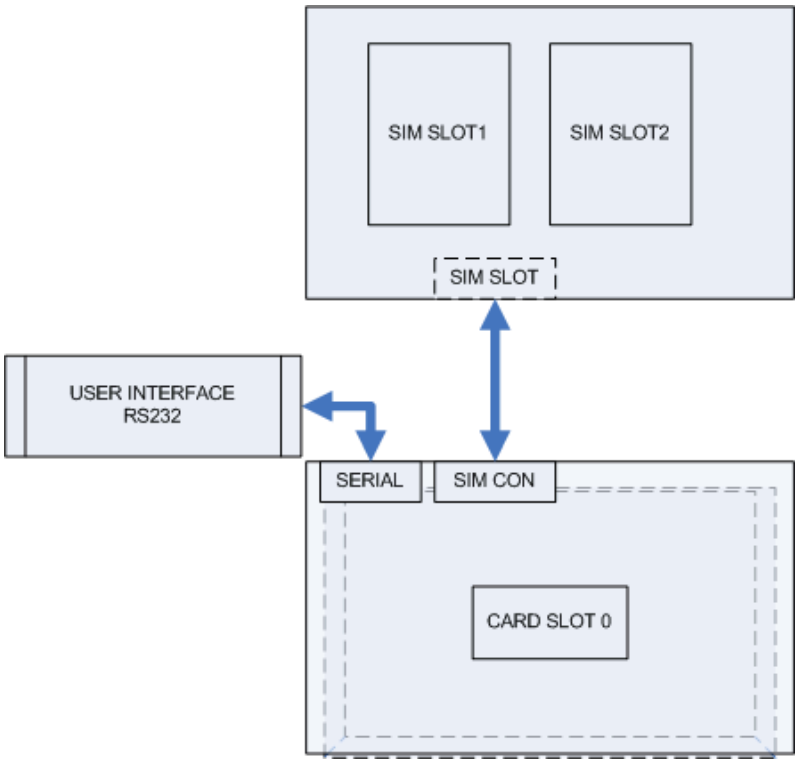
1. System Overview

EICM-1000 is a highly integrated RS232 Smart Card reader module.
Highly integration enables the lowest BOM cost of RS232 Smart Card reader.
The EICM-1000 supports multiple international standards including ISO7816 for IC card standard, PC/SC 1.0 for windows smart card standard, Microsoft WHQL, EMV for Europay MasterCard Visa standard.
The application of EICM-1000 can be generally applied to Smart Card read/write terminal device, such as ATM, POS terminal, Public telephone, E-Commerce, personal consumption on Internet, personal certification, prepay system, loyalty system…etc.

2. General Specifications

Interface Type	Serial, RS232
Interface Protocol	PC/SC 1.0 / PC Smart card industry standard
Type Approve	EMV Level 1 / Based on ISO7816
Other Support	CT-API Microsoft Smart Card for Windows Protocol and Parameter Selection
IC Card Application	Standard To, T1
Other Card Application	I2C Memory card – SLE4418, SLE4428, SLE4432, SLE4442 AT88SC1608, AT45D041
Power Voltage	5[V] (3.3[V] Optional)
Current Consumption	Standby : TBD [mA] Operating : TBD [mA] (IC Card Activate) @ 5[V]

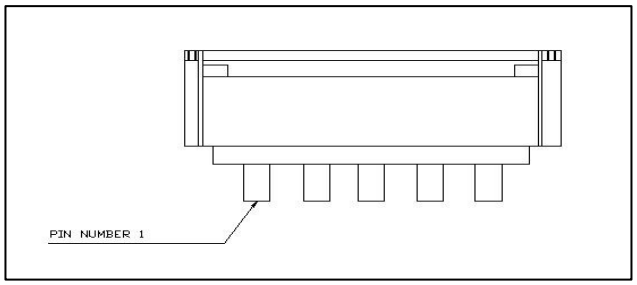
Block Diagram with SAM Board



SIM Connector Pin Description (CON2, Host Interface)

Part No	12505WR-05
Pin 1	POWER(DC +5V)
Pin 2	TXD (EICM-1000 to Host)
Pin 3	RXD(Host to EICM-1000)
Pin 4	SIM Slot Select (Low : SIM1, High : SIM2)
Pin 5	GND

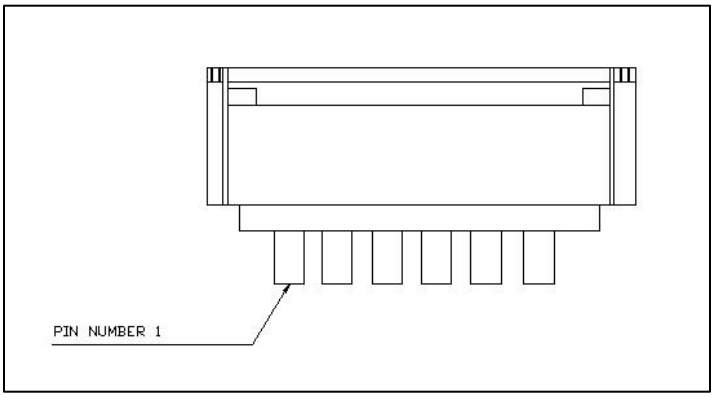
CON2 Pinmap
--->



SIM Connector Pin Description (CON1, SIM Interface)

Part No	12505WR-06
Pin 1	SIM POWER(DC +5V)
Pin 2	SIM select pin
Pin 3	SIM Clock pin
Pin 4	SIM Reset pin
Pin 5	SIM I/O pin
Pin 6	GND

CON1 Pinmap
--->



10019HR-10 connector (if)

Part No	12505WR-06
Pin 1	POWER(DC +5V)
Pin 2	TXD(Module → Host)
Pin 3	RXD(Host → Module)
Pin 4	GND
Pin 5	

Pin 6	
Pin 7	SIM CLOCK
Pin 8	SIM RST
Pin 9	SIM IO
Pin 10	GND

3. Communication Protocol

3-1. Physical Link

BAUDRATE : 4800, 9600, 19200, 38400(Default)
DATA BIT : 8 BIT
STOP BIT : 1 BIT
PAIRTY BIT : NONE

3-2. Package Format

Any message transferred over RS232 is attached with an additional LRC (Note1) byte to provide an error detection of data.

Note1. LRC: longitudinal redundancy check, which does the exclusive-or operation to each character in the string.
For example, if a message of 3 bytes D1, D2, and D3. Then LRC of this message is D1 xor D2 xor D3, and the stream would be D1, D2, D3, LRC.

☞ Commande Example:

PC sends a command MSG_ PC_to_RDR_GetSlotStatus”” to the slot_1 of the reader over RS232.

Command format of MSG_ PC_to_RDR_GetSlotStatus:

Offset	Field	size	Value	Description
0	bMessageType	1	65h	MSG_PC_to_RDR_Get_SlotStatus
1	dwLength	4	00000000h	Message-specific data length
5	bSlot	1	01h	Identifies the slot number for this command (Assume 1)
6	bSeq	1	00h	Sequence number for command. (Assume 0)
7	abRFU	3	000000h	Reserved for Future Use

Data for transferring would be MSG_ PC_to_RDR_GetSlotStatus + LRC:

bMessageType	**dwLength(LSB first)**				**bSlot**	**bSeq**	**abRFU(LSB first)**			**LRC**
65h	00h	00h	00h	00h	01h	00h	00h	00h	00h	64h

4. General Commands

We use general command to control the ISO-7816-3 cards.
All messages begin with a 10-bytes header, followed by message-specific data.
The header consists of a message type (1 byte), a length field (four bytes),
the slot number (1 byte), a sequence number field (1 byte), and either three message specific bytes, or a status field (1 byte), an error field and one message specific byte.
The purpose of the 10-byte header is to provide a constant offset at which message data begins across all messages.

Each command always receives at least one message in response.
The response messages always contain the exact same slot number, and sequence number fields from the header that was contained in command message.

The message type (bMessageType) identifies the message.

The length field (dwLength) is the length of the message not including the 10-byte header.

The slot number (bSlot) identifies which ICC slot is being addressed by the message, if the RS232 interface supports multiple slots.
The slot number is zero-relative, and is in the range of zero to FFh.

The sequence number (bSeq) is monotonically increasing by one counter of bulk messages sent to the RS232 interface.
Because the response to a command always uses the exact same sequence number contained in the command, the host can use the sequence number in a response message to verify that a particular response is the one expected in reply to a particular command. This sequence number is not related to any interaction between the RS232 interface and the ICC itself, but simply tracks the bulk message exchanges between the host and the RS232 interface.
The initial value of the sequence number is not important, but typically starts at zero.

1 byte Slot Status (bStatus) and 1byte Slot Error (bError) are returned in response message as the definition table below.

Table 1 Slot Status register

Offset	Field	size	Value	Description
0	bmICCStatus	2 bits	0, 1, 2	0 - An ICC is present and active (power is on and stable, RST is inactive) 1 - An ICC is present and inactive (not activated or shut down by hardware error) 2 - No ICC is present 3 - RFU
2	bmRFU	4 bits	0	RFU
6	bmCommandStatus	2 bits	0, 1, 2	0 - Processed without error 1 - Failed (error code provided by the error register) 2 - Time Extension is requested

Table 2 Slot error register when bmCommandStatus=1

Error Code	Error Name	Posible Causes
0XFF	CMD_ABORTED	Host aborted the current activity
0XFE	ICC_MUTE	timed out while talking to the ICC
0XFD	XFR_PARITY_ERROR	Parity error while talking to the ICC
0XFB	HW_ERROR	An all inclusive hardware error occurred
0XF8	BAD_ATR_TS	
0XF7	BAD_ATR_TCK	
0XF6	ICC_PROTOCOL-NOT_SUPPORTED	
0XF4	PROCEDURE_BYTE_CONFLICT	
0X01	CMD_FAILED	
0X00	CMD_NO_ERROR	
Others	User Defined and RFU	

Command and response pairs are summarized as below.

Table 3 Summary of command and response

Message Name	bMessage Type	Response Message(s)	bMessage Type
MSG_PC_to_RDR_IccPowerOn	62h	MSG_RDR_to_PC_DataBlock	80h
MSG_PC_to_RDR_IccPowerOff	63h	MSG_RDR_to_PC_SlotStatus	81h
MSG_PC_to_RDR_GetSlotStatus	65h	MSG_RDR_to_PC_SlotStatus	81h
MSG_PC_to_RDR_XfrBlock	6Fh	MSG_RDR_to_PC_DataBlock	80h
MSG_PC_to_RDR_GetParameters	6Ch	MSG_RDR_to_PC_Parameters	82h
MSG_PC_to_RDR_ResetParameters	6Dh	MSG_RDR_to_PC_Parameters	82h
MSG_PC_to_RDR_SetParameters	61h	MSG_RDR_to_PC_Parameters	82h
MSG_PC_to_RDR_Escape	6Bh	MSG_RDR_to_PC_Escape	83h
MSG_PC_to_RDR_IccClock	6Eh	MSG_RDR_to_PC_SlotStatus	81h
MSG_PC_to_RDR_T0APDU	6Ah	MSG_RDR_to_PC_SlotStatus	81h
MSG_PC_TO_RDR_SET_BAUDRATE	66h	MSG_RDR_TO_PC_SET_BAUDRATE	86h
MSG_PC_TO_RDR_GET_DESCRIPTOR	67h	MSG_RDR_TO_PC_GET_DESCRIPTOR	87h
		MSG_RDR_TO_PC_SLOT_CHANGE	85h

The MSG_RDR_TO_PC_SLOT_CHANGE message is sent whenever the RS232 device detects a change in the insertion status of an ICC slot.
If an ICC is either inserted or removed from a slot, this message must be sent.
The presence of this message notifies the host driver that a change has occurred.
When the host driver receives the message, it sends out 10 byte message with 64h header to response.

The next 12 commands are sent by host driver.

4.1 MSG_PC_to_RDR_IccPowerOn

Offset	Field	size	Value	Description
0	bMessageType	1	62h	
1	dwLength	4	00000000h	Message-specific data length
5	bSlot	1	00-FFh	Identifies the slot number for this command
6	bSeq	1	00-FFh	Sequence number for command.
7	bPowerSelect	1	00h, 01h, 02h	Voltage that is applied to the ICC 00h – Automatic Voltage Selection 01h – 5.0 volts 02h – 3.0 volts
8	abRFU	2		Reserved for Future Use

The response to this command message is
the MSG_RDR_to_PC_DataBlock response message and the data returned is
the Answer to Reset (ATR) data.

4.2 MSG_PC_to_RDR_IccPowerOff

Offset	Field	size	Value	Description
0	bMessageType	1	63h	
1	dwLength	4	00000000h	Message-specific data length
5	bSlot	1	00-FFh	Identifies the slot number for this command
6	bSeq	1	00-FFh	Sequence number for command.
7	abRFU	3		Reserved for Future Use

The response to this command message is
the MSG_RDR_to_PC_SlotStatus response message.

4.3 MSG_PC_to_RDR_GetSlotStatus

Offset	Field	size	Value	Description
0	bMessageType	1	65h	
1	dwLength	4	00000000h	Message-specific data length
5	bSlot	1	00-FFh	Identifies the slot number for this command
6	bSeq	1	00-FFh	Sequence number for command.
7	abRFU	3		Reserved for Future Use

The response to this command message is
the MSG_RDR_to_PC_SlotStatus response message.

4.4 MSG_PC_to_RDR_ GetParameters

Offset	Field	size	Value	Description
0	bMessageType	1	6Ch	
1	dwLength	4	00000000h	Message-specific data length
5	bSlot	1	00-FFh	Identifies the slot number for this command
6	bSeq	1	00-FFh	Sequence number for command.
7	abRFU	3		Reserved for Future Use

The response to this command message is
the MSG_RDR_to_PC_Parameters response message.

4.5 MSG_PC_to_RDR_ XfrBlock

Offset	Field	size	Value	Description
0	bMessageType	1	6Fh	
1	dwLength	4		Size of abData field of this message
5	bSlot	1	00-FFh	Identifies the slot number for this command
6	bSeq	1	00-FFh	Sequence number for command.
7	bBWI	1	00-FFh	Used to extend the Block Waiting Timeout for this current transfer. The RS232 interface will timeout the block after “this number multiplied by the Block Waiting Time” has expired.
8	wLevelParameter	2		Use changes depending on the exchange level: Character level: Size of expected data to be returned TPDU level, RFU, = 0000h Short APDU level, RFU, = 00000h Extended APDU level: indicates if APDU begins or ends in this command: 0000h the command APDU begins and ends with this command, 0001h the command APDU begins with this command, and continue in the next PC_to_RDR_XfrBlock, 0002h this abData field continues a command APDU and ends the APDU command, 0003h the abData field continues a command APDU and another block is to follow, 0010h empty abData field, continuation of response APDU is expected in the next MSG_RDR_to_PC_DataBlock
10	abData	Byte array		Data block sent to the RS232 interface. Depending on the exchange level, the interface may send this data “as is” to the ICC, or may modify it before sending it to the ICC.(0 to 65544 bytes)

The response to this command message is
the MSG_RDR_to_PC_DataBlock response message.

4.6 MSG_PC_to_RDR_ ResetParameters

Offset	Field	size	Value	Description
0	bMessageType	1	6Dh	
1	dwLength	4	00000000h	Message-specific data length
5	bSlot	1	00-FFh	Identifies the slot number for this command
6	bSeq	1	00-FFh	Sequence number for command.
7	abRFU	3		Reserved for Future Use

The response to this command message is
the MSG_RDR_to_PC_Parameters response message.

4.7 MSG_PC_to_RDR_ SetParameters

Offset	Field	size	Value	Description
0	bMessageType	1	61h	
1	dwLength	4		Size of abProtocolDataStructure field of this message
5	bSlot	1	00-FFh	Identifies the slot number for this command
6	bSeq	1	00-FFh	Sequence number for command.
7	bProtocolNum	1	00h, 01h,	Specifies what protocol data structure follows. 00h = Structure for protocol T=0 01h = Structure for protocol T=1
8	abRFU	2		Reserved for Future Use
10	abProtocolDataStructure	Byte array		Protocol Data Structure

ProtocolData Structure for Protocol T=0(bProtocolNum=0) (dwLength=00000005h)

Offset	Field	size	Value	Description
10	bmFindexDindex	1		B7-4 – FI – Index into the table 7 in ISO/ IEC 7816-3:1997 selecting a clock rate conversion factor B3-0 – DI - Index into the table 8 in ISO/ IEC 7816-3:1997 selecting a baud rate conversion factor
11	bmTCCKST0	1	00h, 02h	For T=0 ,B0 – 0b, B7-2 – 000000b B1 – Convention used (b1=0 for direct, b1=1 for inverse) Note: Its value is determined by the first byte of the ICC's ATR data. It is here as a placeholder so the same data structure can be used for the MSG_PC_to_RDR_SetParameters and the MSG_RDR_to_PC_GetParameters messages This field is intended to be compatible with parameter rr in Table 2-6 of Part 4 of the PCSC specification.
12	bGuardTimeT0	1	00-FFh	Extra Guardtime between two characters. Add 0to 254 etu to the normal guardtime of 12etu. FFh is the same as 00h.
13	bWaitingIntegerT0	1	00-FFh	WI for T= 0 used to define WWT
14	bClockStop	1	00-03h	ICC Clock Stop Support 00h = Stopping the Clock is not allowed 00h = Stopping the Clock is not allowed 01h = Stop with Clock signal Low 02h = Stop with Clock signal High 03h = Stop with Clock either High or Low

Protocol Data Structure for Protocol T=1 (bProtocolNum=1) (dwLength=00000007h)

Offset	Field	size	Value	Description
10	bmFindexDindex	1		B7-4 – FI – Index into the table 7 in ISO/IEC 7816-3:1997 selecting a clock rate conversion factor B3-0 – DI - Index into the table 8 in ISO/IEC 7816-3:1997 selecting a baud rate conversion factor
11	bmTCCKST1	1	10h, 11h, 12h,13h	For T=1, B7-2 – 000100b B0 – Checksum type (b0=0 for LRC, b0=1 for CRC B1 – Convention used (b1=0 for direct, b1=1 for inverse) Note: Its value is determined by the first byte of the ICC's ATR data. It is here as a placeholder so the same data structure can be used for the MSG_PC_to_RDR_SetParameters and the MSG_RDR_to_PC_GetParameters Messages This field is intended to be compatible with parameter rr in Table 2-6 of Part 4 of the PCSC specification.
12	bGuardTimeT1	1	00-FFh	Extra Guardtime (0 to 254 etu between two characters). If value is FFh, then guardtime is reduced by 1 etu.
13	bWaitingIntegerT1	1	00-9Fh	B7-4 = BWI values 0-9 valid B3-0 = CWI values 0-Fh valid
14	bClockStop	1	00-03h	ICC Clock Stop Support 00 = Stopping the Clock is not allowed 01 = Stop with Clock signal Low 02 = Stop with Clock signal High 03 = Stop with Clock either High or Low
15	bIFSC	1	00-FEh	Size of negotiated IFSC
16	bNadValue	1		Value = 00h if interface doesn't support a value other then the default value. Else value respects ISO/IEC 7816-3, 9.4.2.1

4.8 MSG_PC_to_RDR_ Escape

Offset	Field	size	Value	Description
0	bMessageType	1	6Bh	
1	dwLength	4		Size of abData field of this message
5	bSlot	1	00-FFh	Identifies the slot number for this command
6	bSeq	1	00-FFh	Sequence number for command.
7	abRFU	3	,	Reserved for Future Use
10	abData	Byte array		data block sent to the RS232 interface

The response to this command message is
the MSG_RDR_to_PC_Escape response message.

4.9 MSG_PC_to_RDR_ IccClock

Offset	Field	size	Value	Description
0	bMessageType	1	6Eh	
1	dwLength	4	00000000h	Message-specific data length
5	bSlot	1	00-FFh	Identifies the slot number for this command
6	bSeq	1	00-FFh	Sequence number for command.
7	bClockCommand	1		value = • 00h restarts Clock • 01h Stops Clock in the state shown in the bClockStop field of the MSG_PC_to_RDR_SetParameters command and MSG_RDR_to_PC_Parameters message.
8	abRFU	2		Reserved for Future Use

The response to this command message is

the MSG_RDR_to_PC_SlotStatus response message.

4.10 MSG_PC_to_RDR_ T0APDU

Offset	Field	size	Value	Description
0	bMessageType	1	6Ah	
1	dwLength	4	00000000h	Message-specific data length
5	bSlot	1	00-FFh	Identifies the slot number for this command
6	bSeq	1	00-FFh	Sequence number for command.
7	bmChanges	1	00h,01h, 02h,03h	The value is bitwise OR operation. Bit 0 is associated with field bClassGetResponse Bit 1 is associated with field bClassEnvelope Other bits are RFU. A bit cleared indicates that the associated field is not significant and that default behavior is selected. A bit risen indicates that the associated field is significant.
8	bClassGetResponse	1		Value to force the class byte of the header in a Get Response command. Value = FFh indicates that the class byte of the Get Response command echoes the class byte of the APDU.
9	bClassEnvelope	1		Value to force the class byte of the header in a Envelope command. Value = FFh indicates that the class byte of the Envelope command echoes the class byte of the APDU.

The response to this command message is
the MSG_RDR_to_PC_SlotStatus response message

4.11 MSG_PC_to_RDR_ Set_Baudrate

Offset	Field	size	Value	Description
0	bMessageType	1	66h	
1	dwLength	4	00000000h	Message-specific data length
5	bSlot	1	00-FFh	Identifies the slot number for this command
6	bSeq	1	00-FFh	Sequence number for command.
7	bBaudrateNum	1		value = • 00h 4800bps • 01h 9600bps • 02h 19200bps
8	abRFU	2		Reserved for Future Use

The response to this command message is MSG_RDR_to_PC_Set_Baudrate response message.

4.12 MSG_PC_to_RDR_ Get_Descriptor

Offset	Field	size	Value	Description
0	bMessageType	1	67h	
1	dwLength	4	00000000h	Message-specific data length
5	bSlot	1	00-FFh	Identifies the slot number for this command
6	bSeq	1	00-FFh	Sequence number for command.
7	bDevDesc	1	00-01	value = • 00h get device descriptor, offset 8-9 is RFU • 01h get string descriptor depending on offset 8

8	bStrDesc	1		value = • 00h get string0 descriptor • 01h get string1 descriptor • 02h get string2 descriptor
9	abRFU	1		Reserved for Future Use

The response to this command message is
the MSG_RDR_to_PC_Get_Descriptor response messge.

The following are the corresponding responses of a specific command.

4.13 MSG_RDR_to_PC_DataBlock

Offset	Field	size	Value	Description
0	bMessageType	1	80h	Indicates that a data block is being sent from the RS232 interface.
1	dwLength	4		Size of abData field of this message
5	bSlot	1	Same as command	Identifies the slot number for this command
6	bSeq	1	Same as command	Sequence number for the corresponding command.
7	bStatus	1		Slot status register as defined in table 1
8	bError	1		Slot error register as defined in table 2
9	bChainParameter	1		Depends on the exchange level: Character level, TPDU level, short APDU level, this field is RFU and =00h. Extended APDU level, indicates if the response is complete, to be continued or if the command APDU can continue 00h the response APDU begins and ends in this command 01h the response APDU begins with this command and is to continue 02h this abData field continues the response APDU and ends the response APDU 03h this abData field continues the response APDU and another block is to follow 10h empty abData field, continuation of the command APDU is expected in next MSG_PC_to_RDR_XfrBlock

10	abData	Byte array		This field contains the data returned by the interface. Depending on the exchange level, this may be data and status “as is” from the ICC, or the interface may “filter” the data and status before sending it to the host. It can be 0 -65,538 bytes in length

4.14 MSG_RDR_to_PC_SlotStatus

Offset	Field	Size	Value	Description
0	bMessageType	1	81h	
1	dwLength	4	00000000h	Message-specific data length
5	bSlot	1	same as command	Identifies the slot number for this command
6	bSeq	1	same as command	Sequence number for the corresponding command.
7	bStatus	1		Slot status register as defined in table 1
8	bError	1		Slot error register as defined in table 2
9	bProtocolNum	1	00h,01h, 02h,03h	value = 00h Clock running 01h Clock stopped in state L 02h Clock stopped in state H 03h Clock stopped in an unknown state All other values are RFU

4.15 MSG_RDR_to_PC_Parameters

Offset	Field	Size	Value	Description
0	bMessageType	1	82h	
1	dwLength	4		Size of abProtocolDataStructure field of this message

5	bSlot	1	same as command	Identifies the slot number for this command
6	bSeq	1	same as command	Sequence number for the corresponding command.
7	bStatus	1		Slot status register as defined in table 1
8	bError	1		Slot error register as defined in table 2
9	bProtocolNum	1	00h,01h	Specifies what protocol data structure follows. 00h = Structure for protocol T=0 01h = Structure for protocol T=1
10	abProtocolDataStructure	Byte array		Protocol Data Structure

abProtocolDataStructure can refer to MSG_ PC_to_RDR_SetParameters.

4.16 MSG_RDR_to_PC_Escape

Offset	Field	size	Value	Description
0	bMessageType	1	83h	
1	dwLength	4		Size of abData field of this message
5	bSlot	1	Same as command	Identifies the slot number for this command
6	bSeq	1	Same as command	Sequence number for the corresponding command.
7	bStatus	1		Slot status register as defined in table 1
8	bError	1		Slot error register as defined in table 2
9	bRFU	1	00h	Reserved for Future Use
10	abData	Byte array		Data sent from the RS232 interface

4.17 MSG_RDR_TO_PC_Set_Baudrate

Offset	Field	size	Value	Description
0	bMessageType	1	86h	

1	dwLength	4	00000000h	Message-specific data length
5	bSlot	1	Same as command	Identifies the slot number for this command
6	bSeq	1	Same as command	Sequence number for command.
7	abRFU	3		Reserved for Future Use

4.18 MSG_RDR_TO_PC_Get_Descriptor

Offset	Field	size	Value	Description
0	bMessageType	1	87h	
1	dwLength	4		Size of abData field of this message
5	bSlot	1	Same as command	Identifies the slot number for this command
6	bSeq	1	Same as command	Sequence number for the corresponding command.
7	bStatus	1		Slot status register as defined in table 1
8	bError	1		Slot error register as defined in table 2
9	bRFU	1	00h	Reserved for Future Use
10	abData	Byte array		Data sent from the RS232 interface

The device and string descriptor format is defined as below:

Data structure for device descriptor

Offset	Size	Value	descriptor
0	1	12h	Size of device descriptor in bytes
1	1	01h	Device descriptor type in usb
2	2	1001h	bcd USB in USB or buffer size in RS232
4	4	00000000h	RFU
8	2	8F05h	Vendor ID number
10	2	2595h	Product ID number
12	2	FFh,FFh	bcd device in USB
14	1	01h	Index of string descriptor describing manufacture
15	1	02h	In dex of string descriptor describing product
16	1	03h	Index of string descriptor describing the device serial number

17	1	01h	Number of possible configuration in USB

Data structure for string descriptor

Offset	Size	Value	descriptor
0	1		Size of string in bytes
1	1	03h	string descriptor type in USB
2	abData		String descriptor data

4.19 MSG_RDR_TO_PC_Slot_Change

Offset	Field	size	Value	Description
0	bMessageType	1	85h	
1	dwLength	4		Size of abData field of this message
5	bSlot	1		Identifies the slot number for this command
6	bSeq	1		Used as a placeholder to compatible with the data structure
7	bStatus	1		Slot status register as defined in table 1
8	bError	1		Slot error register as defined in table 2
9	bRFU	1	00h	Reserved for Future Use
10	abData	Byte array		Data sent from the RS232 interface

AbData structure is defined as follows:

Offset	Field	size	Value	Description
0	bSlotChange	1	50h	

1	bmSlotICCState			This field is reported on byte granularity. The size is (2 bits * number of slots) rounded up to the nearest byte. Each slot has 2 bits. The least significant bit reports the current state of the slot (0b = no ICC present, 1b = ICC present). The most significant bit reports whether the slot has changed state since the last MSG_RDR_to_PC_SlotChange message was sent (0b = no change, 1b = change). If no slot exists for a given location, the field returns 00b in those 2 bits. Example: A 2 slot interface reports a single byte with the following format: Bit 0 = Slot 0 current state Bit 1 = Slot 0 changed status Bit 2 = Slot 1 current state Bit 3 = Slot 1 changed status Bit 4 = 0b Bit 5 = 0b Bit 6 = 0b Bit 7 = 0b

5. Vendor Commands

We use MSG_PC_to_RDR_Escape and MSG_RDR_to_PC_Escape to control the cards other than ISO-7816-3 cards.
These cards include memory cards, synchronous cards and other special cards.
We embed our vendor commands in abData of command messages.
The format of these vendor commands are as following:

Note: For Slot0, bSlotNumMask = 0x00. For Slot1, bSlotNumMask = 0x08.

5.1 CMD_SWITCH_CARD_MODE, OP= (0x50 | bSlotNumMask)

Format:

(PC_to_RDR_Escape),40,OP,XL,00,00,00,00,00
XL:Card Mode Switch, the value definitions as follows: _
0x01: ASYNCHRONOUS_CARD_MODE
0x02: I2C_CARD_MODE
0x03: SYNCHRONOUS_CARD_SLE4428_MODE
0x04: SYNCHRONOUS_CARD_SLE4442_MODE
0x05: AT88SC_CARD_MODE
0x06: INPHONE_CARD_MODE
0x07: AT45D041_CARD_MODE

Return:
(RDR_to_PC_Escape)

5.2 CMD_POWER_ON, OP= (0x51 | bSlotNumMask)

Format:
(PC_to_RDR_Escape),40,OP,00,00,00,00,00,00
Return:
(RDR_to_PC_Escape)

5.3 CMD_POWER_OFF, OP= (0x52 | bSlotNumMask)

Format:
(PC_to_RDR_Escape),40,OP,00,00,00,00,00,00
Return:
(RDR_to_PC_Escape)

5.4 CMD_SET_I2C_ADD, OP= (0x60 | bSlotNumMask)

Format:
(PC_to_RDR_Escape),40,OP,Addr1,Addr2,PageSize,00,00,00
Addr1:
_ DeviceAddress | High byte of the word address.

Addr2:
_ The low byte of the word address.
PageSize :
_ The size for page-write feature.
Return:
(RDR_to_PC_Escape)

5.5 CMD_WRITE_I2C, OP= (0x61 | bSlotNumMask)

Format:
(PC_to_RDR_Escape)40,OP,XL,XH,00,00,00,00,data
XL:
The low byte of the data length which writing to the card
XH:
The high byte of the data length which writing to the card
Return:
(RDR_to_PC_Escape)

5.6 CMD_ READ_I2C, OP= (0x62 | bSlotNumMask)

Format:
(PC_to_RDR_Escape)40,OP,XL,XH,00,00,00,00
XL: The low byte of the data length which reading from the card
XH: The high byte of the data length which reading from the card
Return:
(RDR_to_PC_Escape), data

5.7 CMD_ AT45D041_CARD_COMMAND, OP= (0x64 | bSlotNumMask)

Format:
(PC_to_RDR_Escape) 40,OP,XL,XH,00,00,YL,YH,data[0x00.. 0xYHYL]
XL: The low byte of the count which reading from the card
XH: The high byte of the count which reading from the card
YL: The low byte of the count which writing to the card
Y H: The high byte of the count which writing the card
Return:
(RDR_to_PC_Escape), data[0x00 .. 0xXHXL]

5.8 CMD_ SMC_COMMAND, OP= (0x70| bSlotNumMask)

Format:
(PC_to_RDR_Escape) 40,OP,XL,XH,00,00,YL,YH,data[0x00.. 0xYHYL]
XL: The low byte of the count which reading from the card
XH: The high byte of the count which reading from the card
YL: The low byte of the count which writing to the card
YH: The high byte of the count which writing the card
Return:
(RDR_to_PC_Escape), data[0x00 .. 0xXHXL]

5.9 CMD_ SLE4442_CARD_COMMAND, OP= (0x80 | bSlotNumMask)

Format:
(PC_to_RDR_Escape)40,OP,ZZ,YY,0P,XX,03,00,Cmd_1,Cmd_2,Cmd_3
ZZ: The low byte of the data length which reading from the card
YY: The high byte of the data length which reading from the card
03: The low byte of the data length which writing to the card
00: The high byte of the data length which writing to the card
0P: P is only valid when 0xYYZZ>0, i.e. the terminal reads data from the card.
P=1 means read protect bit included. P=0 means not read protect bit back.
XX:
If XX=1, then 0xYYZZ should be '1' and that means the returned one byte is the clock number for card completing the command.
If XX=0, firmware don't care it.

Note:
if 0xYYZZ=0, that means it's a WRITE or VERIFY commands, the firmware will send 1 byte of programming clock number to the driver. The driver or application could determinate if the command successes or fails by it.

Return:
(RDR_to_PC_Escape), data

5.10 CMD_ SLE4442_CARD_BREAK, OP= (0x81 | bSlotNumMask)

Format:
(PC_to_RDR_Escape) 40,OP,00,00,00,00,00,00
Return:
(RDR_to_PC_Escape)

5.11 CMD_ SLE4428_CARD_COMMAND, OP= (0x82 | bSlotNumMask)

Format:
(PC_to_RDR_Escape)40,OP,ZZ,YY,0P,XX,03,00,Cmd_1,Cmd_2,Cmd_3
ZZ: The low byte of the data length which reading from the card
YY: The high byte of the data length which reading from the card
03: The low byte of the data length which writing to the card
00: The high byte of the data length which writing to the card
0P: P is only valid when 0xYYZZ>0, i.e. the terminal reads data from the card.
P=1 means read protect bit included. P=0 means not read protect bit back.
XX:
If XX=1, then 0xYYZZ should be '1' and that means the returned one byte is the clock number for card completing the command.
If XX=0, firmware don't care it.

Note:
If 0xYYZZ=0, that means it's a WRITE or VERIFY commands, the firmware will send 1 byte of programming clock number to the driver. The driver or application could determinate if the command successes or fails by it.

Return:
(RDR_to_PC_Escape), data

5.12 CMD_ INPHONE_CARD_RESET, OP= (0x90 | bSlotNumMask)

Format:
(PC_to_RDR_Escape) 40,OP,00,00,00,00,00,00
Return:
(RDR_to_PC_Escape)

5.13 CMD_ INPHONE_CARD_Read, OP= (0x91 | bSlotNumMask)

Format:
(PC_to_RDR_Escape) 40,OP,XL,XH,00,00,00,00
XL: The low byte of the count which reading from the card
XH: The high byte of the count which reading from the card
Return:
(RDR_to_PC_Escape), data

5.14 CMD_ INPHONE_CARD_PROG, OP= (0x92 | bSlotNumMask)

Format:

(PC_to_RDR_Escape) 40,OP,XL,XH,00,00,00,00
XL: The low byte of the count which program to the card
XH: The high byte of the count which program the card
Note:
The command will return the data after programming.

Return:
(RDR_to_PC_Escape), data

5.15 CMD_ INPHONE_CARD_MOVE_ADDRESS, OP= (0x93 | bSlotNumMask)

Format:
(PC_to_RDR_Escape) 40,OP,00,00,XL,XH,00,00
XL: The low byte of the clock numbers which sent to the card
XH: The high byte of the clock numbers which sent to the card
Return:
(RDR_to_PC_Escape)

5.16 CMD_ INPHONE_CARD_AUTHENTICATION_KEY1, OP=(0x94 | bSlotNumMask)

Format:
(PC_to_RDR_Escape) 40,OP,XL,XH,00,00,YL,YH,data[0x00..0xYHYL]
XL: The low byte of the data length which reading from the card
XH: The high byte of the data length which reading from the card
YL: The low byte of the data length which writing to the card
YH: The high byte of the data length which writing to the card
Return:
(RDR_to_PC_Escape), data[0x00..0xXLXH]

5.17 CMD_ INPHONE_CARD_AUTHENTICATION_KEY2, OP=(0x95| bSlotNumMask)

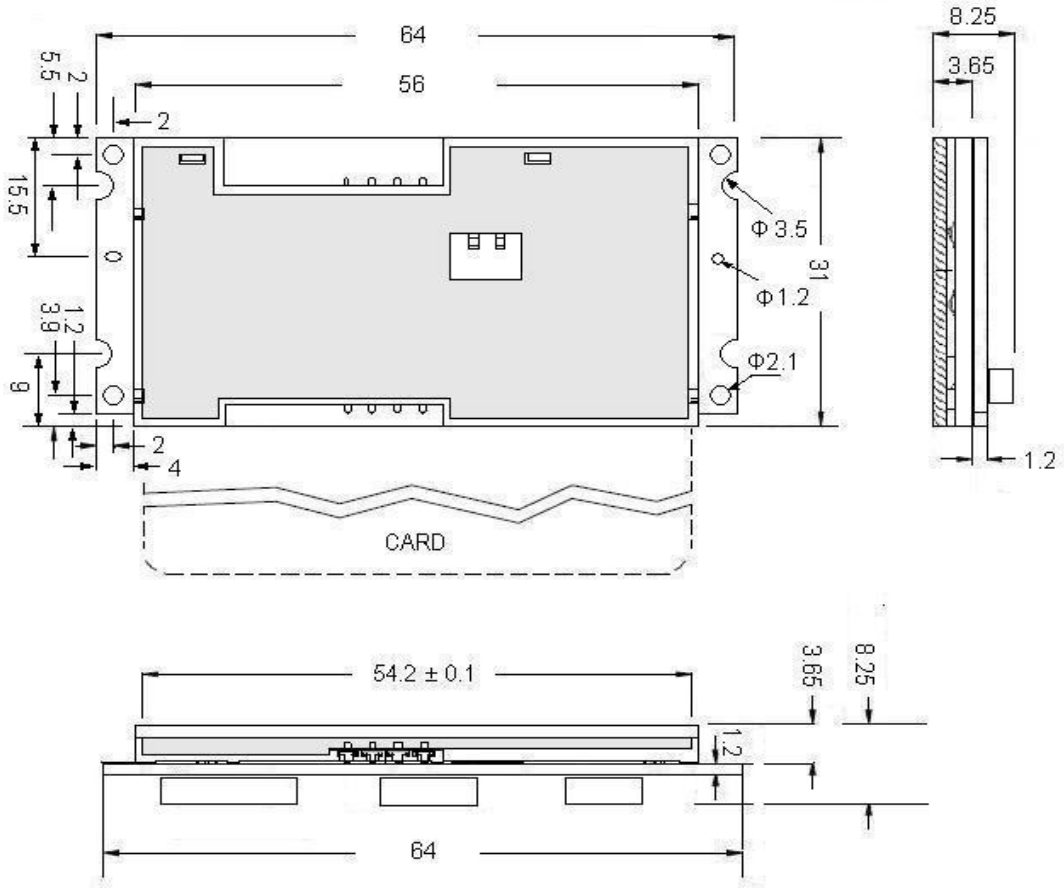
Format:
(PC_to_RDR_Escape) 40,OP,XL,XH,00,00,YL,YH,data[0x00..0xYHYL]
XL: The low byte of the data length which reading from the card
XH: The high byte of the data length which reading from the card
YL: The low byte of the data length which writing to the card
YH: The high byte of the data length which writing to the card
Return:
(RDR_to_PC_Escape), data[0x00..0xXLXH]

5.18 CMD_ SWITCH_PCSC_EMV_MODE, OP= (0xA1 | bSlotNumMask)

Format:
(PC_to_RDR_Escape) 40,OP,XL,XH,YL,YH,00,00,data[0x00..0xYHYL]

XL: The low byte of VID_0x05_
XH: The high byte of VID (0x8F)
YL: The low byte of PID
YH: The high byte of PID, the least important bit is to determine which mode to choose, 0 is PCSC and 1 is EMV mode.
Return:
(RDR_to_PC_Escape)

6. DIMENSIONS


64
56
2
5.5
2
15.5
3.9
1.2
9
2
4
Φ3.5
Φ1.2
Φ2.1
31
CARD
8.25
3.65
1.2
54.2 ± 0.1
3.65
8.25
1.2
64