Trident Message Protocol

Date Created: October, 2013
Version: Rev 2.1

Table of Contents

0. References

“VPOS3351 module communication protocol V2.1” - Author: Pan Xinyong - Publisher: Shenzhen ViewAt Technology Co.,LTD

1. Introduction

1. *Purpose*

This purpose of this document is to describe the data communication interface of the Flomio Trident ICCR+MSR+PICC reader. The document will discuss the method of communication and the message opcodes above the driver layer.

2. *Scope*

This document is intended for developers that want to interface to the Trident from a PC or mobile device. The main focus will be on the message structure and the details of each message supported.

3. *Definitions*

- ICC - Integrated Circuit Card reader (aka SmartCard)
- MSR - Magnetic Stripe Reader
- PICC - Proximity inductive coupling card reader (aka NFC)
- Accessory – Trident NFC reader that plugs into microUSB OTG connector.
- Client - any PC or mobile device that communicates to the Accessory through a USB interface.
- Interface - USB port used by Accessory and Client to exchange data. Implemented with HID driver.
- Passive tags - NFC tags that are battery-less
- Active peers - NFC readers that have batteries and operate in peer-to-peer mode using the Client

4. *Overview*

The Trident device supports bidirectional serial communication with Client over a USB interface. The Accessory operates in USB Device mode so requires a USB Host to work properly. Digital communication is accomplished using Asynchronous Interrupt transfers across an HID driver link.

All messages sent between the Accessory and Client will use the message format defined in this document. The message format will consist of an opcode, length, data, and CRC checksum. The Client and Accessory will also support ACK/NAKing each message which is configurable by the Client.

This protocol is only use for implementing a Client application. The serial port has been opened before communicate with application .The default serial communication format between Accessory module and Client at beginning is: 115200, 8, N, 1. the baud rate could be modified via protocol setup options.
After sending a data package, Accessory module will wait for a response flag until over time 1 second. If no response received until 1 second Accessory module will resend data package twice and report communication error if all fail.

After receive successful response flag, Accessory module will wait for Client's response data package (default over time 1 sec. can be re-set). If correct data package is received, send back response flag within 1 sec. If parity check of received data package fails Accessory module shall send response flag indicating parity check

wrong and wait for data package again.

Over time for receiving data package is 1 second. After data is received over time between characters is 100ms.

5. *Command type*

Master command

Module	Master command	Description
Accessory	0x09	Accessory functions

For each master command, sub commands would be derived through second command type.

Over time for receiving data package is 1 second. After data is received over time between characters is 100ms.

6. *Response code(ST: status)*

Module	Response code	Description
Accessory	00H-FFH	

2. ICC Card function

1. *ICC Check*

Client send:

Data package type	**Command**		**Data Length**		**Data contents**	**Check value**
	Master	**Second**	**High**	**Low**		
02H	02H	00H	00H	01H	Slot	EDC

Accessory module response:

Data package type	Command		Data Length		Data contents	Check value
	Master	Second	High	Low		
02H	02H	00H	00H	01H	<ST>	EDC

Parameter:

Function	int Lib_IccCheck(uchar slot)	
Data Content	<ST>	00H： Success FFH：Check error
	Slot	The ic card is inserted in slot 0
Remark		

Example:
Client send: 02 02 00 00 01 00 03
Accessory module response: 02 02 00 00 01 00 03

2. *ICC Open*

Client send:

Data package type	Command		Data Length		Data contents	Check value
	Master	Second	High	Low		
02H	02H	01H	00H	02H	Slot+VCC_ Mode	EDC

Accessory module response:

Data package type	Command		Data Length		Data contents	Check value
	Master	**Second**	**High**	**Low**		
02H	02H	01H	00H	1+the length of ATR	<ST>+<ATR>	EDC

Parameter:

Function	int Lib_IccOpen(uchar slot,uchar VCC_Mode,uchar *ATR)	
Data Content	<ST>	00H： Success F1H: Check error FFH：Open error
	ATR	The content of Answer Reset
	Slot	The ic card is inserted in slot 0
	VCC_Mode	The voltage appointed 1—5V 2—3V 3—1.8V
Remark		

Example:
Client send: 02 02 01 00 02 00 01 00
Accessory module response: 02 02 01 00 07 00 3B 62 00 00 81 82 5E

3. *ICC Command*

Client send:

Data package type	Command		Data Length		Data contents	Check value
	Master	**Second**	**High**	**Low**		
02H	02H	02H	00H	8+the length of command	Slot+CLA+INS+P1+P2+Lc(the length of comannd)+Le(the length of return data)+command	EDC

Accessory module response:

Data package type	Command		Data Length		Data contents	Check value
	Master	**Second**	**High**	**Low**		
02H	02H	02H	00H	3+the length of return data	<ST>+<return data>+<SW1>+<SW2>	EDC

Parameter:

Function	int Lib_IccCommand(uchar slot, APDU_SEND * ApduSend, APDU_RESP * ApduResp)
Data Content	<ST> 00H： Success FFH：Command error
	Return data The data return from IC card, the range of data length is 0~255。
	SW1 Status Byte 1
	SW2 Status Byte 2
Remark	Before using this command ,please use ICC OPEN command first.

Example:
Client send: 02 02 02 00 15 00 00 a4 04 00 0e ff 31 50 41 59 2E 53 59 53 2E 44 44 46 30 31 23

Accessory module response: 72 65 63 65 6C 65 6E 3D 32 33 02 02 02 00 1A 00 6F 15 84 0E 31 50 41 59 2E 53 59 53 2E 44 44 46 30 31 A5 03 88 01 01 90 00 33

4. *ICC Close*

Client send:

Data package type	**Command**		**Data Length**		**Data contents**	**Check value**
	Master	**Second**	**High**	**Low**		
02H	02H	03H	00H	01H	Slot	EDC

Accessory module response:

Data package type	Command		Data Length		Data contents	Check value
	Master	Second	High	Low		
02H	02H	03H	00H	01H	<ST>	EDC

Parameter:

Function	int Lib_IccClose(uchar slot)	
Data Content	<ST>	00H： Success FFH：Parameter error
	Slot	The ic card is inserted in slot 0
Remark		

Example:
Client send: 02 02 03 00 01 00 00
Accessory module response: 02 02 03 00 01 00 00

5. *ICC SetAutoResp*

Client send:

Data package type	Command		Data Length		Data contents	Check value
	Master	Second	High	Low		
02H	02H	04H	00H	02H	Slot+ Autoresp	EDC

Accessory module response:

Data package type	Command		Data Length		Data contents	Check value
	Master	Second	High	Low		
02H	02H	04H	00H	01H	<ST>	EDC

Parameter:

Function	int Lib_IccSetAutoResp(uchar slot,uchar autoresp)
Data Content	<ST> 00H： Success FFH：Parameter error
Remark	

Example:
Client send: 02 02 04 00 02 00 00 04
Accessory module response: 02 02 04 00 01 00 07

3. MSR Card function

1. *MSR Open*

Client send:

Data package type	Command		Data Length		Data contents	Check value
	Master	Second	High	Low		
02H	01H	00H	00H	00H	0	EDC

Accessory module response:

Data package type	Command		Data Length		Data contents	Check value
	Master	Second	High	Low		
02H	01H	00H	00H	02H	<ST>	EDC

Parameter:

Function	void Lib_McrOpen(void)	
Data Content	<ST>	0000H： Success FFFFH：Parameter error
Remark		

Example:
Client send: 02 01 00 00 00 01
Accessory module response: 02 01 00 00 02 00 00

2. *MSR Close*

Client send:

Data package type	Command		Data Length		Data contents	Check value
	Master	Second	High	Low		
02H	01H	01H	00H	00H	0	EDC

Accessory module response:

Data package type	Command		Data Length		Data contents	Check value
	Master	Second	High	Low		
02H	01H	01H	00H	02H	<ST>	EDC

Parameter:

Function	void Lib_McrClose(void)	
Data Content	<ST>	0000H： Success FFFFH：Parameter error
Remark		

Example:
Client send: 02 01 01 00 00 00
Accessory module response: 02 01 01 00 02 00 01

3. *MSR Reset*

Client send:

Data package type	Command		Data Length		Data contents	Check value
	Master	Second	High	Low		
02H	01H	02H	00H	00H	0	EDC

Accessory module response:

Data package type	Command		Data Length		Data contents	Check value
	Master	Second	High	Low		
02H	01H	02H	00H	02H	<ST>	EDC

Parameter:

Function	void Lib_McrReset(void)	
Data Content	<ST>	0000H： Success FFFFH：Parameter error
Remark		

Example:
Client send: 02 01 02 00 00 03
Accessory module response: 02 01 02 00 02 00 02

4. *MSR Check*

Client send:

Data package type	Command		Data Length		Data contents	Check value
	Master	Second	High	Low		
02H	01H	03H	00H	00H	0	EDC

Accessory module response:

Data package type	Command		Data Length		Data contents	Check value
	Master	Second	High	Low		
02H	01H	03H	00H	02H	<ST>	EDC

Parameter:

Function	int Lib_McrCheck(void)	
Data Content	<ST>	0000H： Success FFFFH：Timeout error
Remark		

Example:
Client send: 02 01 03 00 00 02
Accessory module response: 02 01 03 00 02 00 03

5. *MSR Read*

Client send:

Data package type	Command		Data Length		Data contents	Check value
	Master	Second	High	Low		
02H	01H	04H	00H	00H	0	EDC

Accessory module response:

Data package type	Command		Data Length		Data contents	Check value
	Master	Second	High	Low		
02H	01H	04H	00H	2+L1+L2+L3	<ST>+<L1>+<L2>+<L3>+<track 1 data>+<track 2 data>+<track 3 data>	EDC

Parameter:

Function	int Lib_McrRead(uchar *track1, uchar *track2, uchar *track3, uchar *key,ucha *Key_Len)	
Data Content	<ST>	0000H： Success FFFFH：Check error
	L1	the length of track 1 is L1
	L2	the length of track 2 is L2
	L3	the length of track 3 is L3
	Track 1 data	The track 1 data, the length is L1
	Track 2 data	The track 2 data, the length is L2
	Track 3 data	The track 3 data, the length is L3
Remark	Before using this command ,please use MCR OPEN command first.	

Example:
Client send: 02 01 04 00 00 05

Accessory module response: 02 01 04 00 E7 00 4F 28 6B 31 54 52 41 43 4B 31 32 33 34 35 36 37 38 39 30 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52 53 54 55 56 57 58 59 5A 3C 3E 5F 3D 5C 3A 40 5E 31 32 33 34 35 36 37 38 39 30 41 42 43 44 45 46 47 48 49 4A 4B 5B 56 49 45 5D 00 00 00 32 3D 31 32 33 34 35 36 37 38 39 30 3A 31 32 33 34 35 36 37 38 39 30 3C 39 38 37 36 35 34 33 32 31 30 3E 31 32 00 00 00 33 3D 31 32 33 34 35 36 37 38 39 30 3A 31 32 33 34 35 36 37 38 39 30 3C 39 38 37 36 35 34 33 32 31 30 3E 33 38 32 34 31 37 36 35 30 39 3D 32 34 36 35 31 33 38 37 30 39 3A 36 32 35 33 31 34 39 30 38 37 3C 37 32 31 36 39 35 34 38 30 33 3E 31 32 33 34 35 36 37 38 39 30 3D 3A 3D 3C 3D 3E 3D 31 32 33 34 35 36 37 38 00 00 00 B8

4. PICC Card function

1. *Picc Open*

Client send:

Data package type	Command		Data Length		Data contents	Check value
	Master	Second	High	Low		
02H	03H	00H	00H	00H	0	EDC

Accessory module response:

Data package type	Command		Data Length		Data contents	Check value
	Master	Second	High	Low		
02H	03H	00H	00H	02H	<ST>	EDC

Parameter:

Function	int Picc_Open2(void)	
Data Content	<ST>	0000H： Success FFFFH：Check error
Remark		

Example:
Client send: 02 03 00 00 00 03
Accessory module response: 02 03 00 00 02 00 02

2. *Picc Close*

Client send:

Data package type	Command		Data Length		Data contents	Check value
	Master	**Second**	**High**	**Low**		
02H	03H	01H	00H	00H	0	EDC

Accessory module response:

Data package type	Command		Data Length		Data contents	Check value
	Master	**Second**	**High**	**Low**		
02H	03H	01H	00H	02H	<ST>	EDC

Parameter:

Function	int Picc_Close2(void)	
Data Content	<ST>	0000H： Success FFFFH：Parameter error
Remark		

Example:
Client send: 02 03 01 00 00 02
Accessory module response: 02 03 01 00 02 00 03

3. *Picc Check*

Client send:

Data package type	**Command**		**Data Length**		**Data contents**	**Check value**
	Master	**Second**	**High**	**Low**		
02H	03H	02H	00H	01H	<Mode>	EDC

Accessory module response:

Data package type	**Command**		**Data Length**		**Data contents**	**Check value**
	Master	**Second**	**High**	**Low**		
02H	03H	02H	00H	2+2+ SerialNo[0]	<ST>+<CardType>+ <SerialNo>	EDC

Parameter:

Function	int Picc_Check(uchar mode,uchar *CardType,uchar *UUID)	
Mode	● ‘0’ search type A card firstly： ○ detect multiple type A card, return error； ○ there is no type A card, search type B card continue： ■ Detect multiple type B card, return error； ■ Detect a type B card, active it； ■ Detect no type B card, return no card； ○ Detect a type A card ,search type B card continue: ■ Detect more than one type B card,return error； ■ No type B card, active the A card detected； ● ‘a’ or ‘A’ or 0x0a --- only search the type A card once； ● ‘b’ or ‘B’ or 0x0b --- only search the type B card once；	
Data Content	<ST>	0: Success (-3503): Invalid mode (-3502): The PICC not opened (-3526): Collision Other: Failed
	CardType	The type bye of the card： CardType[0] ： ● ‘A’ —type A card detected ● ‘B’ —type B card detected CardType[1] ： ● ‘C’ —CPU card detected ‘M’ —M1 cad detected
	UUID	Universally Unique ID, the length storage UUID[0] (some cards will create random data)
Remark		

Example:
Client send: 02 03 02 00 01 00 00
Accessory module response: 02 03 02 00 07 00 42 43 D3 E4 9D 1C B1

4. *Picc Command*

Client send:

Data package type	Command		Data Length		Data contents	Check value
	Master	Second	High	Low		
02H	03H	03H	00H	7+the length of command	Slot+CLA+INS+P1+P2 +Lc(the length of comannd)+ CMD+ Le(the length of return data)	EDC

Accessory module response:

Data package type	Command		Data Length		Data contents	Check value
	Master	Second	High	Low		
02H	03H	03H	00H	4+the length of return data	<ST>+<return data>+<SW1>+ <SW2>	EDC

Parameter:

Function	int Picc_Command(APDU_SEND *ApduSend, APDU_RESP *ApduResp)
	Lc: the length of comannd CMD: command data Le: the length of expected return data
Data Content	<ST> 0: Success (-3503): Invalid parameter

		(-3502): The PICC not opened (-3524): Exchange data error
	Return data	The data return from IC card, the range of data length is 0~255。
	SW1	Status Byte 1
	SW2	Status Byte 2
Remark	Before using this command, please use PICC OPEN command first.	

Example:
Client send: 02 03 03 00 14 00 00 a4 04 00 0e ff 31 50 41 59 2E 53 59 53 2E 44 44 46 30 31 23
Accessory module response: 02 03 03 00 16 00 6F 10 84 0E 31 50 41 59 2E 53 59 53 2E 44 44 46 30 31 90 00 17

5. *Picc Halt*

Client send:

Data package type	**Command**		**Data Length**		**Data contents**	**Check value**
	Master	**Second**	**High**	**Low**		
02H	03H	04H	00H	00H	0	EDC

Accessory module response:

Data package type	**Command**		**Data Length**		**Data contents**	**Check value**
	Master	**Second**	**High**	**Low**		
02H	03H	04H	00H	02H	<ST>	EDC

Parameter:

Function	int Picc_HLTA(void)	
Data Content	<ST>	0000H： Success FFFFH：Parameter error
Remark		

Example:
Client send: 02 03 04 00 00 07
Accessory module response: 02 03 04 00 02 00 06

6. *Picc M1Authority*

Client send:

Data package type	**Command**		**Data Length**		**Data contents**	**Check value**
	Master	**Second**	**High**	**Low**		
02H	03H	05H	00H	2+6+ SerialNo[0]	<Type>+<BlkNo>+< Pwd >+< SerialNo >	EDC

Accessory module response:

Data package type	**Command**		**Data Length**		**Data contents**	**Check value**
	Master	**Second**	**High**	**Low**		

02H	03H	05H	00H	02H	<ST>	EDC

Parameter:

Function	int Picc_M1Authority(uchar Type,uchar BlkNo,uchar *Pwd,uchar *UUID)	
Data Content	Type	0x0a or 0x0b represent Password A or Password B
	BlkNo	1bytes from 0 to 63
	Pwd	6bytes, all should be 0xff
	UUID	“Picc_Check( )”returns UUID
	<ST>	0: Success (-3503): Invalid mode (-3502): The PICC not opened (-3526): Collision Other: Failed
Remark	Picc_Check( ) should be successful before using this function	

Example:
Client send: 02 03 05 00 0c 0a 04 ff ff ff ff ff ff 1C 12 49 B4 F7
Accessory module response: 02 03 05 00 02 00 07

7. *Picc M1ReadBlock*

Client send:

Data package type	**Command**		**Data Length**		**Data contents**	**Check value**
	Master	**Second**	**High**	**Low**		
02H	03H	06H	00H	1	<BlkNo>	EDC

Accessory module response:

Data package type	Command		Data Length		Data contents	Check value
	Master	Second	High	Low		
02H	03H	06H	00H	12H	<ST>+<Blk Value>	EDC

Parameter:

Function	int Picc_M1ReadBlock(uchar BlkNo,uchar *BlkValue)	
Data Content	BlkNo	1bytes from 0 to 63
	BlkValue	16bytes return value
	<ST>	0000H： Success FFFFH：Parameter error
Remark		

Example:
Client send: 02 03 06 00 01 04 00
Accessory module response: 02 03 06 00 12 00 07 02 03 06 00 11 00 44 00 00 00 BB FF FF FF 44 00 00 00 04 FB 04 FB 50

8. *Picc M1WriteBlock*

Client send:

Data package type	Command		Data Length		Data contents	Check value
	Master	Second	High	Low		
02H	03H	07H	00H	11H	<BlkNo>+ <BlkValue>	EDC

Accessory module response:

Data package type	Command		Data Length		Data contents	Check value
	Master	Second	High	Low		
02H	03H	07H	00H	02H	<ST>	EDC

Parameter:

Function	int Picc_M1WriteBlock(uchar BlkNo,uchar *BlkValue)	
Data Content	BlkNo	1bytes from 0 to 63
	BlkValue	16bytes Write value
	<ST>	0000H： Success Other：Write error
Remark		

Example:
Client send: 02 03 07 00 11 04 44 00 00 00 bb ff ff ff 44 00 00 00 04 fb 04 fb 55
Accessory module response: 02 03 07 00 02 00 05

9. *Picc M1Operate*

Client send:

Data package type	Command		Data Length		Data contents	Check value
	Master	Second	High	Low		
02H	03H	08H	00H	7	<Type>+<BlkNo>+<Value>+<UpdateBlkNo>	EDC

Accessory module response:

Data package type	Command		Data Length		Data contents	Check value
	Master	Second	High	Low		
02H	08H	08H	00H	02H	<ST>	EDC

Parameter:

Function	int Picc_M1Operate(uchar Type,uchar BlkNo,uchar *Value,uchar UpdateBlkN	
Data Content	Type	+ : means supplement (2b) - : means devalue (2d) = : means Wallet repair (3d)
	BlkNo	1bytes from 1 to 63
	Pwd	4bytes, the write value
	UpdateBlkNo	The terminal write block number
	<ST>	0000H：Success Other: Operate error
Remark	Picc_Check( ) should be successful before using this function	

Example:
Client send: 02 03 08 00 07 2b 04 00 00 00 00 04 27
Accessory module response: 02 03 08 00 02 00 0a

10. *Picc Remove*

Client send:

Data package type	**Command**		**Data Length**		**Data contents**	**Check value**
	Master	**Second**	**High**	**Low**		
02H	03H	09H	00H	00H	0	EDC

Accessory module response:

Data package type	Command		Data Length		Data contents	Check value
	Master	Second	High	Low		
02H	03H	09H	00H	02H	<ST>	EDC

Parameter:

Function	int Picc_Remove(void)	
Data Content	<ST>	0000H : Success (-3502): The PICC not opened (-3525): Card not moved
Remark		

5. System function

1. *Led Control*

Client send:

Data package type	Command		Data Length		Data contents	Check value
	Master	Second	High	Low		
02H	00H	00H	00H	02H	< ledno >< mode >	EDC

Accessory module response:

Data package type	Command		Data Length		Data contents	Check value
	Master	Second	High	Low		
02H	00H	00H	00H	02H	<ST>	EDC

Parameter:

Function	void Lib_SetLed(unsigned char ledno,unsigned char mode);	
Data Content	ledno	The led number from 1 to 2
	mode	1:led on 0:led off
	<ST>	0000H：Success FFFFH : Parameter error
Remark		

Example:
Client send: 02 00 00 00 02 01 01 02
Accessory module response: 02 00 00 00 02 00 01

2. *Beep Control*

Client send:

Data package type	Command		Data Length		Data contents	Check value
	Master	Second	High	Low		
02H	00H	01H	00H	01H	< time >	EDC

Accessory module response:

Data package type	Command		Data Length		Data contents	Check value
	Master	**Second**	**High**	**Low**		
02H	00H	01H	00H	02H	<ST>	EDC

Parameter:

Function	void Lib_Beep(void);	
Data Content	ledno	The led number from 1 to 2
	Time	The beep on time： 1: means 1time 2: means 2times And so on
	<ST>	0000H：Success FFFFH : Parameter error
Remark		

3. *Led Ctr*

Client send:

Data package type	Command		Data Length		Data contents	Check value
	Master	**Second**	**High**	**Low**		
02H	00H	02H	00H	07H	< ledno > < ledontime > < ledofftime > < time >	EDC

Accessory module response:

Data package type	Command		Data Length		Data contents	Check value
	Master	**Second**	**High**	**Low**		
02H	00H	02H	00H	02H	<ST>	EDC

Parameter:

Function	void LibLedCtr(unsigned char ledno,int ledontime,int ledofftime,int time)	
Data Content	ledno	The led number from 1 to 2
	ledontime	Two bytes, LED on time
	ledofftime	Two bytes, LED off time
	Time	Two bytes ,Delay time (10ms)
	<ST>	0000H：Success FFFFH : Parameter error
Remark		