EZNR Series
API Reference Manual
v1.1

Version: 1.1
Date: July 25, 2001
Castles Technology Co., LTD.
Tel: 886 (2) 8665-0530
Fax: 886 (2) 8665-0531
E-Mail: casauto@casauto.com.tw
Web: www.casauto.com.tw

WARNING

Information in this document is subject to change without prior notice.

ABOUT THIS MANUAL

This manual describes the application programming interface (API) functions; the application software developers should refer to this manual to develop their own software to drive the EZ100/200NR Smart Card Reader.

Change history:

API Version	Date	Descriptions
V1.0.0.0	2001.05.14	Created

Table Of Contents

1. Introduction

The EZ100/200NR Smart Card Reader is powerful equipment for Smart Card usage. It not only supports CPU cards, but a variety of memory cards. Section 2 is a detailed explanation of all API structures and functions.

Appendix A lists all status code that may be returned by our API functions.

2. Application Programming Interface

The API description contains the following content: all data structure that is used in the API, literal definitions, and public functions.

2.1 Data Structure

We have defined these structures in order to simplify and clarify the inputs and outputs of our API. Some of them are aliases of basic types in ANSI C, while others are combinations (structures) of basic types. All structures are available after you include “ EZNRAPI.H” in your project.

2.1.1 Reader Type

You can use the following symbols to recognize the outputs of EzConnectReader() function.

EZ100N_PRODUCT	=	0x01xxxxxx
EZ200N_PRODUCT	=	0x02xxxxxx
RS232_INTERFACE	=	0xxx01xxxx
PS2_INTERFACE	=	0xxx02xxxx
USB_INTERFACE	=	0xxx04xxxx

2.1.2 Port Number

They are used in EzConnectReader() function.

COM1	=	0
COM2	=	1
COM3	=	2
COM4	=	3
PS2_PORT	=	4
USB1	=	5
USB2	=	6
USB3	=	7
USB4	=	8

2.1.3 Working Card

For EZ200 series, there are two choices of working card (MAIN card and SAM card), so you may assign one of the following symbols. For EZ100 series, you should use the default.

MAIN_CARD	=	0
SAM_CARD	=	1
DEFAULT_CARD	=	MAIN_CARD

2.1.4 Card Protocol

When connecting the card, you should assign the protocol that your card supports. Memory card has its own protocol type, while CPU cards may support T=0 protocol or T=1 protocol. To identify which protocol is to be used, please consult your card manufacturer.

EZSMC_PROTOCOL_T0	=	0x00000001	(T=0 protocol)
EZSMC_PROTOCOL_T1	=	0x00000002	(T=1 protocol)
EZSMC_PROTOCOL_SYNC	=	0x00000010	(Memory Card protocol)
EZSMC_PROTOCOL_UNDEFINED	=	0x00000000	(no active protocol)

Notes: It is recommended that you assign the protocol to "**EZSMC_PROTOCOL_T0 | EZSMC_PROTOCOL_T1 | EZSMC_PROTOCOL_SYNC**" for the system deciding it.

2.1.5 Card Type

We defined memory card and CPU card, which expressed the most significant classification for IC cards.

UNKNOWN_CARD	=	0
CPU_CARD	=	1
MEMORY_CARD	=	2

2.1.6 Card State

They are used in GetCardState() function, which tells if there is a card inserted or powered.

EZ_CARD_ABSENT	=	0
EZ_CARD_PRESENT	=	1
EZ_CARD_POWERED	=	2

2.1.7 Structure Size

We have limited the size of ATR bytes and reader type string. More than that, we introduce two constant to you so that you can use them to define the length of APDU.

MAX_ATR_SIZE	=	64
EZ_READER_TYPE_STR_LEN	=	17
DEFAULT_APDU_IN_SIZE	=	256
DEFAULT_APDU_OUT_SIZE	=	256

2.1.8 Handle of Reader

We use a double word as a handle of reader in order to recognize the working reader. On connecting the reader, you should use the address of the handle as input; on the other occasions, you should use the handle itself as input.

EZHANDLE	=	DWORD
PEZHANDLE	=	address of EZHANDLE variable

2.1.9 ATR Structure

It is used in EzConnectCard() function, which tells the card type. Moreover, in CPU card case, it contains the Answer-To-Reset bytes of the card.

EZ_ATR	=	ATR structure
PEZ_ATR	=	address of EZ_ATR variable

The C definition of EZ_ATR is as follows:

```
typedef struct{
          BYTE      CardType;
          struct{
               BYTE     Length;
               BYTE     Buffer[MAX_ATR_SIZE];
          }ATR;
}EZ_ATR;
```

For example, if you defined a variable “ATR” of type EZ_ATR, then

ATR.CardType	=	BYTE indicating card type (see “Card Type”)
ATR.ATR.Length	=	BYTE indicating the length of ATR bytes
ATR.ATR.Buffer	=	BYTE indicating the content of ATR bytes

2.1.10 APDU Structure

The are two ways to send and receive CPU card command and response data, one is use APDU structure, the other one is use transparent buffer (please refer to EzTransmit() function).

You must call EzAPDUInit() one time to allocate memory before using the APDU variable in sending and receiving commands.

EZ_APDU	=	APDU structure
PEZ_APDU	=	address of EZ_APDU variable

The C definition of EZ_APDU is as follows:

```
typedef struct{
            BYTE      CLA;
            BYTE      INS;
            BYTE      P1;
            BYTE      P2;
            union{
                  struct{    // for T1
                        BYTE b0;
                        BYTE b1;
                        BYTE b2;
                        BYTE Used; // if the protocol is T1, "Used"
indicates that the number of Lc
bytes will be sent. The range is
between 0 and 3.
                        }Lc;

                  BYTE P3; // for T0
            };

            struct{    // for T1
                  BYTE b0;
                  BYTE b1;
                  BYTE b2;
                  BYTE Used; // if the protocol is T1, "Used" indicates
 that the number of Le bytes will be
 sent. The range is between 0 and 3.
            }Le;
```

```
          struct{
               DWORD BufferSize; // the size of the DataIn buffer
               DWORD BufferLength; // the length of the data in this
buffer to be sent.
               PBYTE   Buffer; // data buffer to be sent.
          }DataIn;

          struct{

               DWORD BufferSize; // the size of the DataOut buffer
               DWORD BufferLength; // the length of the data that is
returned from the ICC.
               PBYTE   Buffer; // data buffer
          }DataOut;

          DWORD   Status; // ICC response SW1SW2
}EZ_APDU;
```

For example, if you defined a variable “APDU” of type EZ_APDU and using T=1 protocol card, then

Before call EzSendAPDU():
APDU.CLA :BYTE indicating CLA value
APDU.INS : BYTE indicating INS value
APDU.P1 :   BYTE indicating P1 value
APDU.P2 :   BYTE indicating P2 value

APDU.Lc.Used: BYTE indicating number of Lc byte to be sent.

APDU.Lc.b0, APDU.Lc.b1, apud.Lc.b2: BYTE indicating Lc value to be sent

APDU.Lc.Used: BYTE indicating number of Lc byte to be sent.

APDU.Lc.b0, APDU.Lc.b1, apud.Lc.b2: BYTE indicating Lc value to be sent

APDU.DataIn.BufferLength: DWORD of the actual length of sending data. It is given by user before calling EzSendAPDU() function.

APDU.DataIn.Buffe : BYTE indicating the content of sending data bytes. It is also given by user before calling EzSendAPDU() function.

APDU.Le.Used: BYTE indicating number of Le byte to be sent.

APDU.Le.b0, APDU.Le.b1, apud.Le.b2: BYTE indicating Le value.

After call EzSendAPDU():
APDU.Status : DWORD indicating status for currently sending/receiving APDU commands.

APDU.DataOut.BufferLength: DWORD of the actual length of receiving data. Its value is assigned by our API.

APDU.DataOut.Buffer: BYTE indicating the content of receiving data bytes from ICC.

Maybe the usage of Lc and Le confuses you. Let' s make it clear. According to ISO/IEC 7816-4, there are two types of sending / receiving APDU commands: T=0 and T=1. If your CPU card only supports T=0, you can transfer at most 255 bytes for sending and 256 bytes for receiving per each APDU command. That is, use the following assignment for P3:

APDU.P3: BYTE for sending or expect receiving data length

If your CPU card supports T=1, you can transfer at most 65536 bytes per each APDU (ex. case 2E, case 3E, case 4E). If your command still needs at most 255 bytes to send and 256 bytes to response (ex. case 2S, case 3S, case 4S), use following assignment for Lc(or Le):

APDU.Lc.Used = 1 (indicate this is an ISO 7816 case 3S,4S command, i.e. only one Lc will be sent.)
APDU.Lc .b0 = BYTE for Lc value
APDU.DataIn.BufferLength: DWORD of the actual length of sending data.

APDU.Le.Used = 1(indicate this is an ISO 7816 case 2S,4S command, i.e. only one Le will be sent.)
APDU.Le .b0 = BYTE for expect receiving data length.

If your command needs more than 256 bytes to transfer, you should use following assignment for Lc(or Le, expected length)

instead:

APDU.Lc.Used = 3 (indicate this is an ISO 7816 case 3E,4E command, i.e. 3 bytes' Lc will be sent.)
APDU.Lc.b0 = first byte of Lc
APDU.Lc.b1 = second byte of Lc
APDU.Lc.b2 = third byte of Lc

APDU.Le.Used = 3 (indicate this is an ISO 7816 case 2E,4E command, i.e. 3 bytes' Le will be sent.)
APDU.Le.b0 = first byte of Le
APDU.Le.b1 = second byte of Le
APDU.Le.b2 = third byte of Le

The APDU structure is powerful, if it is still confuses you, you can use EzTransmit() function to sending and receiving command for CPU card.

2.2 General-purpose Functions

The following functions are used for basic controls over EZNR reader and card detection. They work no matter you use memory card or CPU card in your application. The data sturcture of each parameter is explained in " Data Structure" section.

2.2.1 EzConnectReader()

Description

Connect to the reader on a designated port and get the handle of the reader to access it.

Syntax

```
EZSMC_STATUS
EzConnectReader(
            ULONG          PortNumber,
            PEZHANDLE    pHandle,
            PULONG         pReaderType
);
```

Parameters

PortNumber –

Indicate that the port to which the reader is attached is to be connected. It can be assigned to one of these: (Please refer to " Port Number")

COM1
COM2
COM3
COM4

pHandle –

A Pointer to an EZHANDLE variable to get the reader' s handle. (Please refer to " Handle of Reader")

pReaderType –

A Pointer to a ULONG variable to get the connected reader type. (Please refer to " Reader Type")

Return Value

Please refer to Appendix A (Return Value Table).

2.2.2 EzDisconnectReader

Description

Disconnect the reader that is previously connected.

Syntax

```
EZSMC_STATUS
EzDisconnectReader(
            EZHANDLE    Handle
);
```

Parameters

Handle –

The handle of the connected reader that is to be disconnected from the port. (Please refer to “Handle of Reader”)

Return Value

Please refer to Appendix A (Return Value Table).

2.2.3 EzConnectCard

Description

Connect to the card in the designated reader. It will power on the card and set protocol.

Syntax

```
EZSMC_STATUS
EzConnectCard(
        EZHANDLE    Handle,
        ULONG       SelectCard,
        ULONG       PreferedProtocols,
        PULONG  pActiveProtocol,
        PEZ_ATR  pATR
);
```

Parameters

Handle –

The handle of the reader that the card is inserted into. (Please refer to " Handle of Reader")

SelectCard –

Indicate which card on the reader is to be access. It can be assigned to one of these:
(Please refer to " Working Card")
MAIN_CARD
SAM_CARD

PreferedProtocols –

Supplies a bit mask of acceptable protocols for connection. Possible Values, which may be combined with the OR operation, are:
(Please refer to " Card Protocol")
EZSMC_PROTOCOL_T0
EZSMC_PROTOCOL_T1
EZSMC_PROTOCOL_SYNC
Notes: It is recommended that you assign the protocol to " **EZSMC_PROTOCOL_T0 | EZSMC_PROTOCOL_T1 | EZSMC_PROTOCOL_SYNC**" for the system deciding it.

pActiveProtocol –

A Pointer to a ULONG variable that indicates the receiving current active protocol of the card. (Please refer to " Card Protocol") pATR –

A Pointer to an EZ_ATR structure that receives the ATR data from the card. (Please refer to " ATR Structure")

Return Value

Please refer to Appendix A (Return Value Table).

2.2.4 EzDisconnectReader()

Description

Disconnect the card in the designated reader. It will power off the card.

Syntax

```
EZSMC_STATUS
EzDisconnectCard(
            EZHANDLE     Handle,
            ULONG        SelectCard
);
```

Parameters

Handle –

The handle of the reader that the card is inserted into. (Please refer to “Handle of Reader”)

SelectCard –

Indicate which card on the reader is to be accessed. It can be assigned to one of these:
(Please refer to “Working Card”)
MAIN_CARD
SAM_CARD

Return Value

Please refer to Appendix A (Return Value Table).

2.2.5 EzGetState()

Description

Get the current state of the card.

Syntax

```
EZSMC_STATUS
EzGetState(
     EZHANDLE  Handle,
     ULONG     SelectCard,
     PULONG    pState
);
```

Parameters

Handle –

The handle of the reader that the card is inserted into.
(Please refer to “ Handle of Reader”)

SelectCard –

Indicate which card on the reader is to be access. It can be assigned to one of these:
(Please refer to “ Working Card”)

MAIN_CARD
SAM_CARD

pState –

Pointer to a ULONG variable that receives the current state of the card. The state can be:
(Please refer to “ Card State”)

EZ_CARD_ABSENT
EZ_CARD_PRESENT
EZ_CARD_POWERED

Return Value

Please refer to Appendix A (Return Value Table).

2.2.6 EzGetReaderName ()

Description

Get the reader type string of the designated reader.

Syntax

```
EzGetReaderName(
     EZHANDLE  Handle,
     PUCHAR    RecvBuffer,
     ULONG     RecvBufferSize,
     PULONG    RecvBufferLength
);
```

Parameters

Handle –

The handle of the connected reader, whose type is to be returned.

RecvBuffer –

A Pointer to a buffer that is big enough to receive the data of the reader type.

RecvBufferSize –

Indicates the size of the receive buffer.

RecvBufferLength –

A Pointer to a ULONG variable that indicates the length of the receiving data in RecvBuffer.

Return Value

Please refer to Appendix A (Return Value Table).

2.2.7 EzWaitCardInsert()

Description

Windows: The function will detect if the card is in the designed reader. If absent, it will wait for a designed time until the card is inserted.

Dos: Not supported.

Syntax

```
EZSMC_STATUS
EzWaitCardInsert(
     EZHANDLE  Handle,
     ULONG       SelectCard,
     ULONG       WaitTime
);
```

Parameters

Handle –

The handle of the connected reader to wait a card to be inserted. (Please refer to " Handle of Reader")

SelectCard –

Indicate which card on the reader is to be access. It can be assigned to one of these:
(Please refer to " Working Card")

MAIN_CARD
SAM_CARD

WaitTime –

The time measured in milliseconds (ms) that the designed reader waits for. If no card is inserted beyond the waiting time, it will return EZSMC_TIMEOUT.
Assign "INFINITE" value means no timeout.

Return Value

Please refer to Appendix A (Return Value Table).

2.2.8 EzWaitCardRemove()

Description

Windows: The function will detect if the card is in the designed reader. If present, it will wait for a designed time until the card is removed.

Dos: Not supported.

Syntax

```
EZSMC_STATUS
EzWaitCardRemove(
     EZHANDLE  Handle,
     ULONG     SelectCard,
     ULONG     WaitTime
);
```

Parameters

Handle –

The handle of the connected reader to wait a card to be removed. (Please refer to “Handle of Reader”)

SelectCard –

Indicate which card on the reader is to be access. It can be assigned to one of these:
(Please refer to “Working Card”)
MAIN_CARD
SAM_CARD

WaitTime –

The time measured in milliseconds (ms) that the designed reader waits for. If the card is not removed beyond the waiting time, it will return EZSMC_TIMEOUT. Assign "INFINITE" value means no timeout.

Return Value

Please refer to Appendix A (Return Value Table).

2.3 Functions for CPU card

These functions are provided specifically for CPU card.

2.3.1 EzAPDUInit()

Description

Windows: Initialize the APDU structure and allocate "DataIn" buffer and "DataOut" buffer according to the BufferSize.

DOS: Initialize the APDU structure.

Syntax

```
EZSMC_STATUS
EzAPDUInit(
            PEZ_APDU    pAPDU,
            DWORD   DataInBufferSize,
            DWORD   DataOutBufferSize

);
```

Parameters

pAPDU –

A Pointer to an EZ_APDU structure that will be initialized.
(Please refer to “ APDU Structure”)

DataInBufferSize:

The size of DataIn.Buffer of APDU to be allocated.

DataOutBufferSize:

The size of DataOut.Buffer of APDU to be allocated.

Return Value

Please refer to Appendix A (Return Value Table).

2.3.2 EzSendAPDU()

Description

Send the APDU to the card in the designate reader.

Syntax

```
EZSMC_STATUS
EzSendAPDU(
     EZHANDLE  Handle,
     ULONG     SelectCard,
     PEZ_APDU  pAPDU
);
```

Parameters

Handle –

The handle of the reader that the card is inserted into. (Please refer to " Handle of Reader")

SelectCard –

Indicate which card on the reader is to be access. It can be assigned to one of these:
(Please refer to " Working Card")
MAIN_CARD
SAM_CARD

pAPDU –

A Pointer to an EZ_APDU variable that will be sent to the card. The variable must be initialized by EzAPDUInit() function. (Please refer to " APDU Structure")

Return Value

Please refer to Appendix A (Return Value Table).

2.3.3 EzTransmit()

Description

Send the APDU command to the card in the designate reader by transparent buffer.

Syntax

```
EZSMC_STATUS
EzSendAPDU(
            IN EZHANDLE         Handle,
            IN ULONG            SelectCard,
            IN LPBYTE               pbSendBuffer,
            IN DWORD                SendLength,
            OUT LPBYTE          pbRecvBuffer,
            IN OUT LPDWORD pRecvLength
);
```

Parameters

Handle –

The handle of the reader that the card is inserted into.

(Please refer to “Handle of Reader”)

SelectCard –

Indicate which card on the reader is to be access. It can be assigned to one of these:

(Please refer to “Working Card”)

MAIN_CARD

SAM_CARD

pbSendBuffer –

The APDU command buffer to be send.

For example:

“\x00\x84\x00\x00\x08”   //Generate Random number

SendLength –

The length of data to be send.

pbRecvBuffer –

The IC card response data.

pRecvLength –

The length of IC card response data.

Return Value

Please refer to Appendix A (Return Value Table).

2.4 Functions for SLE4442/SLE4432 Memory Card

These functions are provided specifically for SLE4442 or SLE4432 Memory Card. The only difference between the two cards is that SLE4442 card has a 3-byte Programmable Security Code (PSC), and update operation cannot work unless the user has verified the PSC. SLE4432 card does not have PSC verification.

To be clear, we defined

MM_ADDRESS = ULONG

In every function using the address of a memory card.

2.4.1 SLE4442_Read_Main_Memory ()

Description

Get data from the SLE4442 memory card.

It also supports SLE4432 memory card.

Syntax

```
EZSMC_STATUS
SLE4442_Read_Main_Memory(
            EZHANDLE hCard,
            LPBYTE pbRecvBuffer,
            MM_ADDRESS aStartAddr,
            LPDWORD pcbRecvLength
);
```

Parameters

hCard –

The handle of the reader that the card is inserted into. (Please refer to “Handle of Reader”)

pbRecvBuffer –

A pointer to a BYTE array that is capable to receive the returned data.

aStartAddr –

Specify the address of the first byte of the receiving data in the memory card. The range must be between 0 and 255.

pbRecvBuffer –

A pointer to a DWORD that specifies the length of expected returned data and receives the actual number of bytes received from the memory card. The range of this value must be between 1 and 256.

Return Value

Please refer to Appendix A (Return Value Table).

2.4.2 SLE4442_Update_Main_Memory ()

Description

Update the new data to memory card. This function will not check if the data is really written in the memory card, because it is possible that some address has been protected so that this function has no effect on these bytes.

It also supports SLE4432 memory card.

Syntax

```
EZSMC_STATUS
SLE4442_Update_Main_Memory(
            EZHANDLE hCard,
            LPBYTE pbTransmitBuffer,
            MM_ADDRESS aStartAddr,
            DWORD TransmitLength
);
```

Parameters

hCard –

The handle of the reader that the card is inserted into. (Please refer to “Handle of Reader”)

pbTransmitBuffer –

A pointer to a buffer that is to be updated to the memory card.

aStartAddr –

Specify the address of the first byte of the updating data in the memory card. The range must be between 0 and 255.

TransmitLength –

Specifies the length of data to update. The range of this value must be between 1 and 256.

Return Value

Please refer to Appendix A (Return Value Table).

2.4.3 SLE4442_Update_Main_MemoryA()

Description

Update the new data to memory card. Different to **SLE4442_Update_Main_Memory()**, this function will check if the data is really written in the memory card. If some of the addresses have been protected, the function will return fail. However, unprotected memory addresses are always updated, no matter the function returns protection fail or not.

It also supports SLE4432 memory card.

Syntax

```
EZSMC_STATUS
SLE4442_Update_Main_MemoryA(
        EZHANDLE hCard,
        LPBYTE pbTransmitBuffer,
        MM_ADDRESS aStartAddr,
        DWORD TransmitLength
);
```

Parameters

hCard –

The handle of the reader that the card is inserted into. (Please refer to “Handle of Reader”)

pbTransmitBuffer –

A pointer to a buffer that is to be updated to the memory card.

aStartAddr –

Specify the address of the first byte of the updating data in the memory card. The range must be between 0 and 255.

TransmitLength –

Specifies the length of data to update. The range of this value must be between 1 and 256.

Return Value

Please refer to Appendix A (Return Value Table).

2.4.4 SLE4442_Read_Protection_Memory()

Description

Get the protected bits of the addresses with protect bit. The bits are addressed from 0 to 31.

If the protected bit is 0, the corresponding address will not be allowed to update any more

It also supports SLE4432 memory card.

Syntax

```
EZSMC_STATUS
SLE4442_Read_Protection_Memory(
            EZHANDLE hCard,
            LPBYTE pbRecvBuffer
);
```

Parameters

hCard –

The handle of the reader that the card is inserted into. (Please refer to “Handle of Reader”)

pbRecvBuffer –

A pointer to a buffer that receives the returned data. The buffer size must be larger than or equal to 4. The returned data is four bytes (equal to 32 bits).

The relation between receive buffer and main memory as below:

Buffer[0].bit0 : address 0x00

Buffer[0].bit1 : address 0x01

…

Buffer[0].bit7 : address 0x07

Buffer[1].bit0 : address 0x08

…

Buffer[1].bit7 : address 0x0F

…

For example:

If buffer[0]=0x01, means the address 0x00 of main memory is read only even you verified the PSC correctly.

Return Value

Please refer to Appendix A (Return Value Table).

2.4.5 SLE4442_Write_Protection_Memory()

Description

Compare the input bytes with the data of the assigned address in memory card. If they the same, the protected bit of the address will be set to 0(protected). If the protected bit is 0, the corresponding address will not be allowed to update any more, and the protected bit will never be allowed to set back to 1(unprotected).

It also supports SLE4432 memory card.

Syntax

```
SLE4442_Write_Protection_Memory(
        SCARDHANDLE  hCard,
        LPBYTE  pbTransmitBuffer,
        MM_ADDRESS  aStartAddr,
        DWORD  TransmitLength
);
```

Parameters

hCard –

The handle of the reader that the card is inserted into. (Please refer to “Handle of Reader”)

pbTransmitBuffer –

A pointer to a buffer that is to be compared and protected.

aStartAddr –

Specify the address of the first byte of data to be compared and protected. The range must be between 0 and 31.

TransmitLength –

Specifies the length of data that will be compared and protected. The range of this value must be between 1 and 32.

Return Value

Please refer to Appendix A (Return Value Table).

2.4.6 SLE4442_Read_Security_Memory()

Description

Get the security data (EC and PSC) from the SLE4442 memory card.

It does not support SLE4432 memory card.

Syntax

```
EZSMC_STATUS
SLE4442_Read_Security_Memory(
            EZHANDLE hCard,
            LPBYTE pbRecvBuffer
);
```

Parameters

hCard –

The handle of the reader that the card is inserted into. (Please refer to " Handle of Reader")

pbRecvBuffer –

A pointer to a buffer that receives the returned data. The buffer size must be larger than or equal to 4. The returned data (EC and PSC) is 4 bytes.

Return Value

Please refer to Appendix A (Return Value Table).

2.4.7 SLE4442_Update_Security_Memory()

Description

Update the security data (PSC) in the memory card.
It does not support SLE4432 memory card.

Syntax

```
EZSMC_STATUS
SLE4442_Update_Security_Memory(
            EZHANDLE  hCard,
               LPBYTE pbTransmitBuffer,
               MM_ADDRESS aStartAddr,
               DWORD TransmitLength
);
```

Parameters

hCard –

The handle of the reader that the card is inserted into. (Please refer to “ Handle of Reader”)

pbTransmitBuffer –

A pointer to a buffer that is used to update the data in the memory card.

aStartAddr –

Specify the address of the first byte of data to be updated. The range must be between 1 and 3.

TransmitLength –

Specifies the length of data to be updated. The range of this value must be between 1 and 3.

Return Value

Please refer to Appendix A (Return Value Table).

2.4.8 SLE4442_Compare_Verification_Data()

Description

Compare the entered data with the PSC(Programmable Security Code) data in memory card. If success, update operations of the SLE4442 memory card could work.

It does not support SLE4432 memory card.

Syntax

```
EZSMC_STATUS
SLE4442_Compare_Verification_Data(
    EZHANDLE hCard,
    BYTE PSC1,
    BYTE PSC2,
    BYTE PSC3
);
```

Parameters

hCard –

The handle of the reader that the card is inserted into. (Please refer to “Handle of Reader”)

PSC1, PSC2, PSC3 –

The PSC code to be verified.

Return Value

Please refer to Appendix A (Return Value Table).

2.5 Functions for SLE4428/SLE4418 Memory Card

These functions are provided specifically for SLE4428 or SLE4418 Memory Card. The only difference between the two cards is that SLE4428 card has a 2-byte rogrammablel Security Code (PSC), and update peration cannot work unless the user has verified the PSC. SLE4418 card does not have PSC verification.

To be clear, we defined

MM_ADDRESS = ULONG

In every function using the address of a memory card.

2.5.1 SLE4428_Write_Data_With_Protect_Bit()

Description

Get data with protect bits from the memory card. Each address has its own protect bit. Once the protect bit of some address has been written (set to 0), the corresponding address cannot be updated anymore, but reading is always allowed.

It also supports SLE4418 memory card.

Syntax

```
EZSMC_STATUS
SLE4428_Write_Data_With_Protect_Bit(
    IN EZHANDLE hCard,
    IN LPBYTE pbTransmitBuffer,
    IN MM_ADDRESS aStartAddr,
    IN DWORD TransmitLength
);
```

Parameters

hCard –

The handle of the reader that the card is inserted into. (Please refer to “Handle of Reader”)

pbTransmitBuffer –

A pointer to a BYTE buffer that is to be sent to the memory card.

aStartAddr –

Specify the address of the first byte of data to be written. The range must be between 0 and 1023.

TransmitLength –

Specifies the length of data that will be written. The range of this value must be between 1 and 1024.

Return Value

Please refer to Appendix A (Return Value Table).

2.5.2 SLE4428_Write_Data_With_Protect_BitA()

Description

Updates the new data with protect bit written (set to 0) to memory card. Different to **SLE4428_Write_Data_With_Protect_Bit()**, this function will check if the data is really written in the memory card. If some of the addresses have been protected, the function will return fail. However, unprotected memory addresses are always updated, no matter the function returns protection fail or not.

It also supports SLE4418 memory card.

Syntax

```
EZSMC_STATUS
SLE4428_Write_Data_With_Protect_BitA(
    IN EZHANDLE hCard,
    IN LPBYTE pbTransmitBuffer,
    IN MM_ADDRESS aStartAddr,
    IN DWORD TransmitLength
);
```

Parameters

hCard –

The handle of the reader that the card is inserted into. (Please refer to “Handle of Reader”)

pbTransmitBuffer –

A pointer to a BYTE buffer that is to be sent to the memory card.

aStartAddr –

Specify the address of the first byte of data to be written. The range must be between 0 and 1023.

TransmitLength –

Specifies the length of data that will be written. The range of this value must be between 1 and 1024.

Return Value

Please refer to Appendix A (Return Value Table).

Note:

In DOS LIB, you should use SLE4428_Write_With_P_BitA() instead of SLE4428_Write_Data_With_Protect_BitA()

2.5.3 SLE4428_Write_Data_Without_Protect_Bit()

Description

Updates the new data to memory card.

It also supports SLE4418 memory card.

Syntax

```
EZSMC_STATUSI
SLE4428_Write_Data_Without_Protect_Bit(
    EZHANDLE  hCard,
      LPBYTE pbTransmitBuffer,
    MM_ADDRESS  aStartAddr,
      DWORD TransmitLength
);
```

Parameters

hCard –

The handle of the reader that the card is inserted into. (Please refer to "Handle of Reader")

pbTransmitBuffer –

A pointer to a BYTE buffer that is to be sent to the memory card.

aStartAddr –

Specify the address of the first byte of data to be written. The range must be between 0 and 1023.

TransmitLength –

Specifies the length of data that will be written. The range of this value must be between 1 and 1024.

Return Value

Please refer to Appendix A (Return Value Table).

2.5.4 SLE4428_Write_Data_Without_Protect_BitA()

Description

Updates the new data to memory card. Different to **SLE4428_Write_Data_Without_Protect_Bit()**, this function will check if the data is really written in the memory card. If some of the addresses have been protected, the function will return fail. However, unprotected memory addresses are always updated, no matter the function returns protection fail or not.

It also supports SLE4418 memory card.

Syntax

```
EZSMC_STATUSI
SLE4428_Write_Data_Without_Protect_BitA(
    EZHANDLE  hCard,
      LPBYTE pbTransmitBuffer,
    MM_ADDRESS  aStartAddr,
      DWORD TransmitLength
);
```

Parameters

hCard –

The handle of the reader that the card is inserted into. (Please refer to " Handle of Reader")

pbTransmitBuffer –

A pointer to a BYTE buffer that is to be sent to the memory card.

aStartAddr –

Specify the address of the first byte of data to be written. The range must be between 0 and 1023.

TransmitLength –

Specifies the length of data that will be written. The range of this value must be between 1 and 1024.

Return Value

Please refer to Appendix A (Return Value Table).

Note:

In DOS LIB, you should use SLE4428_Write_Without_P_BitA() instead of SLE4428_Write_Data_Without_Protect_BitA()

2.5.5 SLE4428_Write_Protect_Bit()

Description

Compare the data of the assigned address with entered data. If they are the same, the protect bit of the address will be written (set to 0).

It also supports SLE4418 memory card.

Syntax

```
EZSMC_STATUS
SLE4428_Write_Protect_Bit(
            EZHANDLE hCard,
            LPBYTE pbTransmitBuffer,
            MM_ADDRESS aStartAddr,
            DWORD TransmitLength
);
```

Parameters

hCard –

The handle of the reader that the card is inserted into. (Please refer to “ Handle of Reader”)

pbTransmitBuffer –

A pointer to a BYTE buffer that is to be sent to the memory card.

aStartAddr –

Specify the address of the first byte of data to be written. The range must be between 0 and 1023.

TransmitLength –

Specifies the length of data that will be written. The range of this value must be between 1 and 1024.

Return Value

Please refer to Appendix A (Return Value Table).

2.5.6 SLE4428_Read_Data_With_Protect_Bit()

Description

Get data with protect bits from the memory card. Each address has its own protect bit. Once the protect bit of some address has been written (set to 0), the address cannot be written anymore, but reading is always allowed.

It also supports SLE4418 memory card.

Syntax

```
EZSMC_STATUS
SLE4428_Read_Data_With_Protect_Bit(
     EZHANDLE hCard,
     LPBYTE pbRecvBuffer,
     LPBYTE pbProtectBuffer,
     MM_ADDRESS aStartAddr,
     LPDWORD pcbRecvLength
);
```

Parameters

hCard –

The handle of the reader that the card is inserted into. (Please refer to " Handle of Reader")

pbRecvBuffer –

A pointer to a buffer that receives the returned data.

pbProtectBuffer –

A pointer to a buffer that receives the protect bits of each address.

aStartAddr –

Specify the address of the first byte of data that will be read. The range must be between 0 and 1023.

pcbRecvLength –

A pointer to a DWORD that specifies the length of expected returned data and receives the actual number of bytes received from the memory card. The range of this value must be between 1 and 1024.

Return Value

Please refer to Appendix A (Return Value Table).

2.5.7 SLE4428_Read_Data_Without_Protect_Bit()

Description

Get the data from the SLE4428 memory card.
It also supports SLE4418 memory card.

Syntax

```
EZSMC_STATUS
SLE4418_Read_Data_Without_Protect_Bit(
    EZHANDLE hCard,
    LPBYTE pbRecvBuffer,
    MM_ADDRESS aStartAddr,
    LPDWORD pcbRecvLength
);
```

Parameters

hCard –

The handle of the reader that the card is inserted into. (Please refer to “Handle of Reader”)

pbRecvBuffer –

A pointer to a buffer that receives the returned data.

aStartAddr –

Specify the address of the first byte of data that will be read. The range must be between 0 and 1023.

pcbRecvLength –

A pointer to a DWORD that specifies the length of expected returned data and receives the actual number of bytes received from the memory card. The range of this value must be between 1 and 1024.

Return Value

Please refer to Appendix A (Return Value Table).

2.5.8 SLE4428_PSC_Verification()

Description

Compare the entered data with the PSC data in memory card. If success, update operation of this memory card could work.

It does not support SLE4418 memory card.

Syntax

```
EZSMC_STATUS
SLE4428_PSC_Verification(
     EZHANDLE hCard,
     BYTE PSC1,
     BYTE PSC2
);
```

Parameters

hCard –

The handle of the reader that the card is inserted into.

(Please refer to “Handle of Reader”)

PSC1, PSC2 –

The PSC code to be verified.

Return Value

Please refer to Appendix A (Return Value Table).

2.6 Functions for SLE4404 Memory Card

These functions are provided specifically for SLE4404 Memory Card. This kind of card has two sets of commands: User Memory commands, which have effect on the user memory of the card; (All) Memory commands, which have effect on all memory of the card. Either command set is active only after its Security Code verification is done.

To be clear, we defined
MM_ADDRESS = ULONG
In every function using the address of a memory card.

2.6.1 SLE4404_Read_Memory()

Description

Get data from the SLE4404 memory card.

Syntax

```
EZSMC_STATUS
SLE4404_Read_Memory(
    EZHANDLE  hCard,
      LPBYTE pbRecvBuffer,
      MM_ADDRESS aStartAddr,
      LPDWORD pcbRecvLength
);
```

Parameters

hCard –

The handle of the reader that the card is inserted into. (Please refer to “Handle of Reader”)

pbRecvBuffer –

A pointer to a BYTE array that is capable to receive the returned data.

aStartAddr –

Specify the address of the first byte of the receiving data in the memory card. The range must be between 0 and 51(0x33).

PcbRecvLength –

A pointer to a DWORD that specifies the length of expected returned data and receives the actual number of bytes received from the memory card. The range of this value must be between 1 and 51.

Return Value

Please refer to Appendix A (Return Value Table).

2.6.2 SLE4404_Read_User_Memory()

Description

Get data of user memory in the SLE4404 memory card.

Syntax

```
EZSMC_STATUS
SLE4404_Read_User_Memory(
    EZHANDLE  hCard,
      LPBYTE pbRecvBuffer,
      MM_ADDRESS aStartAddr,
      LPDWORD pcbRecvLength
);
```

Parameters

hCard –

The handle of the reader that the card is inserted into. (Please refer to “Handle of Reader”)

pbRecvBuffer –

A pointer to a BYTE array that is capable to receive the returned data.

aStartAddr –

Specify the address of the first byte of the receiving data in the memory card. The range must be between 0 and 25.

PcbRecvLength –

A pointer to a DWORD that specifies the length of expected returned data and receives the actual number of bytes received from the memory card. The range of this value must be between 1 and 26.

Return Value

Please refer to Appendix A (Return Value Table).

2.6.3 SLE4404_Compare_User_Code()

Description

Compare the entered data with the User Codes data in memory card. If success, the user memory without being written can be written once.

Syntax

```
EZSMC_STATUS
SLE4404_Compare_User_Code(
    EZHANDLE  hCard,
      BYTE Code1,
      BYTE Code2
);
```

Parameters

hCard –

The handle of the reader that the card is inserted into. (Please refer to " Handle of Reader")

Code1, Code2 –

The User Codes to be verified.

Return Value

Please refer to Appendix A (Return Value Table).

2.6.4 SLE4404_Compare_Memory_Code()

Description

Compare the entered data with the Memory Codes data in memory card. If success, any address in the memory can be updated.

Syntax

```
EZSMC_STATUS
SLE4404_Compare_Memory_Code(
    EZHANDLE  hCard,
      BYTE Code1,
      BYTE Code2,
      BYTE Code3,
      BYTE Code4
);
```

Parameters

hCard –

The handle of the reader that the card is inserted into. (Please refer to “Handle of Reader”)

Code1, Code2, Code3, Code4 –

The Memory Codes to be verified.

Return Value

Please refer to Appendix A (Return Value Table).

2.6.5 SLE4404_Erase_Memory()

Description

Erase the memory of any address in the SLE4404 memory card. This operation will work only after SLE4404_Compare_Memory_Code succeed. After this operation, the memory that has been erased can be written once.

Syntax

```
EZSMC_STATUS
SLE4404_Erase_Memory(
     EZHANDLE hCard,
     MM_ADDRESS aStartAddr,
     DWORD TransmitLength
);
```

Parameters

hCard –

The handle of the reader that the card is inserted into. (Please refer to “ Handle of Reader”)

aStartAddr –

Specify the address of the first byte of data to be erased. The range must be between 0 and 51(0x33).

TransmitLength –

Specifies the length of data that will be erased. The range of this value must be between 1 and 51.

Return Value

Please refer to Appendix A (Return Value Table).

2.6.6 SLE4404_Write_Memory()

Description

Write the new data to memory card. This function will not check if the data is really written in the memory card, because it is possible that some address has been protected so that this function has no effect on these bytes.

Syntax

```
EZSMC_STATUS
SLE4404_Write_Memory(
    EZHANDLE  hCard,
      LPBYTE pbTransmitBuffer,
      MM_ADDRESS aStartAddr,
      DWORD TransmitLength
);
```

Parameters

hCard –

The handle of the reader that the card is inserted into. (Please refer to “Handle of Reader”)

pbTransmitBuffer –

A pointer to a buffer that is to be written to the memory card.

aStartAddr –

Specify the address of the first byte of data to be written. The range must be between 0 and 51(0x33).

TransmitLength –

Specifies the length of data that will be written. The range of this value must be between 1 and 51.

Return Value

Please refer to Appendix A (Return Value Table).

2.6.7 SLE4404_Write_MemoryA()

Description

Write the new data to memory card. Different to **SLE4404_Write_Memory()**, this function will check if the data is really written in the memory card. If some of the addresses have been protected, the function will return fail. However, unprotected memory addresses are always updated, no matter the function returns protection fail or not.

Syntax

```
EZSMC_STATUS
SLE4404_Write_MemoryA(
    EZHANDLE  hCard,
      LPBYTE pbTransmitBuffer,
    MM_ADDRESS  aStartAddr,
      DWORD TransmitLength
);
```

Parameters

hCard –

The handle of the reader that the card is inserted into. (Please refer to " Handle of Reader")

pbTransmitBuffer –

A pointer to a buffer that is to be written to the memory card.

aStartAddr –

Specify the address of the first byte of data to be written. The range must be between 0 and 51(0x33).

TransmitLength –

Specifies the length of data that will be written. The range of this value must be between 1 and 51.

Return Value

Please refer to Appendix A (Return Value Table).

2.6.8 SLE4404_Write_User_Memory()

Description

Write the new data to the user memory region of the memory card. This function will not check if the data is really written in the memory card, because it is possible that some address has been protected so that this function has no effect on these bytes.

Syntax

```
EZSMC_STATUS
SLE4404_Write_User_Memory(
    EZHANDLE hCard,
     LPBYTE pbTransmitBuffer,
    MM_ADDRESS  aStartAddr,
     DWORD TransmitLength
);
```

Parameters

hCard –

The handle of the reader that the card is inserted into. (Please refer to “Handle of Reader”)

pbTransmitBuffer –

A pointer to a buffer that is to be written to the memory card.

aStartAddr –

Specify the address of the first byte of data to be written. The range must be between 0 and 25.

TransmitLength –

Specifies the length of data that will be written. The range of this value must be between 1 and 26.

Return Value

Please refer to Appendix A (Return Value Table).

2.6.9 SLE4404_Write_User_MemoryA()

Description

Write the new data to the user memory region of the memory card. Different to **SLE4404_Write_Memory()**, this function will check if the data is really written in the memory card. If some of the addresses have been protected, the function will return fail. However, unprotected memory addresses are always updated, no matter the function returns protection fail or not.

Syntax

```
EZSMC_STATUS
SLE4404_Write_User_MemoryA(
    EZHANDLE hCard,
     LPBYTE pbTransmitBuffer,
    MM_ADDRESS aStartAddr,
     DWORD TransmitLength
);
```

Parameters

hCard –

The handle of the reader that the card is inserted into. (Please refer to “Handle of Reader”)

pbTransmitBuffer –

A pointer to a buffer that is to be written to the memory card.

aStartAddr –

Specify the address of the first byte of data to be written. The range must be between 0 and 25.

TransmitLength –

Specifies the length of data that will be written. The range of this value must be between 1 and 26.

Return Value

Please refer to Appendix A (Return Value Table).

2.6.10 SLE4404_Enter_Test_Mode()

Description

Enter the test mode.

Syntax

```
EZSMC_STATUS
SLE4404_Enter_Test_Mode(
     EZHANDLE hCard
);
```

Parameters

hCard –

The handle of the reader that the card is inserted into.
(Please refer to “Handle of Reader”)

Return Value

Please refer to Appendix A (Return Value Table).

2.6.11 SLE4404_Exit_Test_Mode()

Description

Exit the test mode.

Syntax

```
EZSMC_STATUS
SLE4404_Exit_Test_Mode(
     EZHANDLE hCard
);
```

Parameters

hCard –

The handle of the reader that the card is inserted into. (Please refer to " Handle of Reader")

Return Value

Please refer to Appendix A (Return Value Table).

2.6.12 SLE4404_Blow_Fuse()

Description

Blow the fuse.

Syntax

```
EZSMC_STATUS
SLE4404_Blow_Fuse(
     EZHANDLE hCard
);
```

Parameters

hCard –

The handle of the reader that the card is inserted into. (Please refer to “Handle of Reader”)

Return Value

Please refer to Appendix A (Return Value Table).

2.7 Functions for SLE4406/4436/5536 Memory Card

These functions are provided specifically for SLE4406/4436/5536 Memory Card.

To be clear, we defined

MM_ADDRESS = ULONG

In every function using the address of a memory card.

2.7.1 SLE4436_Read_Memory()

Description

Read data from SLE4436 memory card.

It also supports SLE4406/5536 memory card.

Syntax

```
EZSMC_STATUS
SLE4436_Read_Memory(
            EZHANDLE hCard,
            LPBYTE pbRecvBuffer,
            MM_ADDRESS aStartAddr,
            LPDWORD pcbRecvLength
);
```

Parameters

hCard –

The handle of the reader that the card is inserted into. (Please refer to " Handle of Reader")

pbRecvBuffer –

A pointer to a BYTE array that is capable to receive the returned data.

aStartAddr –

Specify the address of the first byte of the receiving data in the memory card. The range must be between 0 and 47 for SLE4436/5536 and between 0 and 12 for SLE4406.

PcbRecvLength –

A pointer to a DWORD that specifies the length of expected returned data and receives the actual number of bytes received from the memory card. The range of this value must be between 1 and 48 for SLE4436/5536 and between 1 and 13 for SLE4406.

Return Value

Please refer to Appendix A (Return Value Table).

2.7.2 SLE4436_Read_Counter_Stages()

Description

Read the five counter stages from SLE4436 memory card.
It also supports SLE4406/5536 memory card.

Syntax

```
      EZSMC_STATUS
      SLE4436_Read_Counter_Stages(
                    EZHANDLE hCard,
                    BYTE Stage[5]
);
```

Parameters

hCard –

The handle of the reader that the card is inserted into.
(Please refer to " Handle of Reader")

Stage[5] –

A five-byte array that receives the five stages data.

Return Value

Please refer to Appendix A (Return Value Table).

2.7.3 SLE4436_Write_Memory()

Description

Write the new data to SLE4436 memory card.
It also supports SLE4406/5536 memory card.

Syntax

```
EZSMC_STATUS
SLE4436_Write_Memory(
            EZHANDLE hCard,
            LPBYTE pbTransmitBuffer,
            MM_ADDRESS aStartAddr,
            DWORD TransmitLength
);
```

Parameters

hCard –

The handle of the reader that the card is inserted into. (Please refer to “Handle of Reader”)

pbTransmitBuffer –

A pointer to a buffer that is to be written to the memory card.

aStartAddr –

Specify the address of the first byte of data to be written. The range must be between 0 and 47 for SLE4436/5536 and between 0 and 12 for SLE4406.

TransmitLength –

Specifies the length of data that will be written. The range of this value must be between 1 and 48 for SLE4436/5536 and between 1 and 13 for SLE4406.

Return Value

Please refer to Appendix A (Return Value Table).

2.7.4 SLE4436_Write_Counter_Stage()

Description

Write the value to the assigned stage of SLE4436 memory card.

It also supports SLE4406/5536 memory card.

Syntax

```
EZSMC_STATUS
SLE4436_Write_Counter_Stage(
            EZHANDLE hCard,
    BYTE Stage,
    BYTE Data
);
```

Parameters

hCard –

The handle of the reader that the card is inserted into. (Please refer to “Handle of Reader”)

Stage –

Indicates which stage will be written. This value must be between 1 and 5.

Data –

The value that will be written into the assigned stage.

Return Value

Please refer to Appendix A (Return Value Table).

2.7.5 SLE4436_Reload()

Description

Erase all bits of stage 1 and writes a bit of higher stage on SLE4436 memory card.

It also supports SLE4406/5536 memory card.

Syntax

```
EZSMC_STATUS
SLE4436_Reload(
            EZHANDLE hCard,
            BYTE BitAddr
);
```

Parameters

hCard –

The handle of the reader that the card is inserted into. (Please refer to " Handle of Reader")

BitAddr –

Indicates which bit will be written. The range must be between 0 and 31.

0 ~ 7 is stage 2 byte;

8 ~ 15(0Fh) is stage 3 byte;

16(10h) ~ 23(17h) is stage 4 byte;

24(18h) ~ 31(1Fh) is stage 5 byte.

Return Value

Please refer to Appendix A (Return Value Table).

2.7.6 SLE4436_Verify_Transport_Code()

Description

Compare the entered data with the Transport codes in SLE4436 memory card.

It also supports SLE4406/5536 memory card.

Syntax

```
EZSMC_STATUS
SLE4436_Verify_Transport_Code(
     EZHANDLE hCard,
     BYTE Code1,
     BYTE Code2,
     BYTE Code3
);
```

Parameters

hCard –

The handle of the reader that the card is inserted into.

(Please refer to " Handle of Reader")

Code1, Code2, Code3 –

The Transport Codes to be verified.

Return Value

Please refer to Appendix A (Return Value Table).

2.7.7 SLE4436_ Authentication()

Description

Perform authentication for SLE4436 memory card.
It also supports SLE5536 memory card.
It does not support SLE4406 memory card.

Syntax

```
EZSMC_STATUS
SLE4436_Authentication(
            EZHANDLE hCard,
            BYTE ReturnCode[2],
            BYTE Key,
            BYTE ClockPulse,
            BYTE Code[6]
);
```

Parameters

hCard –

The handle of the reader that the card is inserted into. (Please refer to “ Handle of Reader”)

ReturnCode[2] –

A two-byte array that receives the return codes from the memory card.

Key –

The value that is to be sent to the memory card.

ClockPulse –

Code[6] –

A six-byte of challenge data.

Return Value

Please refer to Appendix A (Return Value Table).

2.7.8 SLE5536_Extended_ Authentication()

Description

Perform authentication for SLE5536 memory card.
It does not support SLE4406/4436 memory card.

Syntax

```
EZSMC_STATUS
SLE5536_Authentication(
            EZHANDLE hCard,
            BYTE ReturnCode[2],
            BYTE Key,
            BYTE ClockPulse,
            BYTE Code[6]
);
```

Parameters

hCard –

The handle of the reader that the card is inserted into. (Please refer to " Handle of Reader")

ReturnCode[2] –

A two-byte array that receives the return codes from the memory card.

Key –

The value that is to be sent to the memory card.

ClockPulse –

Code[6] –

A six-byte of challenge data.

Return Value

Please refer to Appendix A (Return Value Table).

2.8 Functions for I2C Memory Card

These functions are provided specifically for I2C Memory Card.
To be clear, we defined
MM_ADDRESS = ULONG
In every function using the address of a memory card.

2.8.1 I2C_Read_Memory()

Description

Read data from I2C memory card.

Syntax

```
EZSMC_STATUS
I2C_Read_Memory(
            EZHANDLE  hCard,
            BYTE ReadCommand,
            MM_ADDRESS aStartAddr,
            LPBYTE pbRecvBuffer,
            LPDWORD pcbRecvLength,
            BYTE Dummy,
            BYTE AddrByteNum
);
```

Parameters

hCard –

The handle of the reader that the card is inserted into. (Please refer to " Handle of Reader")

ReadCommand –

The reading command byte of I2C memory card.

aStartAddr –

Specify the address of the first byte of the receiving data in the memory card.

pbRecvBuffer –

A pointer to a BYTE array that is capable to receive the returned data.

pcbRecvLength –

A pointer to a DWORD that specifies the length of expected returned data and receives the actual number of bytes received from the memory card. This value must be more than 0.

Dummy –

Indicates whether or not to use dummy write address. If dummy equals to 0, dummy is not used; otherwise dummy is used.

AddrByteNum –

Indicates the number of the address bytes. The value must be between 0 and 2.

Return Value

Please refer to Appendix A (Return Value Table).

2.8.2 I2C_Write_Memory()

Description

Write data to I2C memory card.

Syntax

```
EZSMC_STATUS
I2C_Write_Memory(
            EZHANDLE  hCard,
            BYTE WriteCommand,
            MM_ADDRESS aStartAddr,
            LPBYTE pbTransmitBuffer,
            DWORD TransmitLength,
            BYTE AddrByteNum
);
```

Parameters

hCard –

The handle of the reader that the card is inserted into. (Please refer to " Handle of Reader")

WriteCommand –

The writing command byte of I2C memory card.

aStartAddr –

Specify the address of the first byte of data to be written to the memory card.

pbTransmitBuffer –

A pointer to a buffer that is to be written to the memory card.

TransmitLength –

Specifies the length of data that will be written. This value must be more than 0.

AddrByteNum –

Indicates the number of the address bytes. The value must be between 0 and 2.

Return Value

Please refer to Appendix A (Return Value Table).

Appendix A: Return Value Table

Value (in Hex)	Definition	Descriptions
0000	EZSMC_SUCCESS	The command is successfully executed.
0001	EZSMC_COMM_OPEN_FAIL	Cannot open the COM port.
0002	EZSMC_INVALID_HANDLE	The supplied handle was invalid.
0003	EZSMC_INVALID_PARAMETER	One or more of the supplied parameters could not be properly interpreted.
0004	EZSMC_NO_MEMORY	Not enough memory available to complete this command.
0005	EZSMC_TIMEOUT	The user-specified timeout value has expired.
0006	EZSMC_BUFFER_TOO_SMALL	The data buffer to receive returned data is too small for the returned data.
0007	EZSMC_UNKNOWN_READER	The specified reader name is not recognized.
0008	EZSMC_NO_MEDIA	The operation requires a Smart Card, but no Smart Card is currently in the device.
0009	EZSMC_UNKNOWN_CARD	The specified smart card name is not recognized.
000A	EZSMC_NOT_READY	The reader or smart card is not ready to accept commands.
000B	EZSMC_INVALID_VALUE	One or more of the supplied parameters values could not be properly interpreted.
000C	EZSMC_COMM_ERROR	An internal communications error has been detected.
000D	EZSMC_DEVICE_BUSY	The device is processing, transmitting, or receiving.
000E	EZSMC_INVALID_ATR	An ATR obtained from the device is not a valid ATR string.
000F	EZSMC_READER_UNSUPPORTED	The reader driver does not meet minimal requirements for support.
0010	EZSMC_CARD_UNSUPPORTED	The smart card does not meet minimal requirements for support.
0011	EZSMC_UNPOWERED_CARD	The smart card does not power.
0012	EZSMC_FAILURE	Other errors have occurred.
0013	EZSMC_DEVICE_DATA_ERROR	The data transmitting to or receiving from the reader have some error.
0014	EZSMC_CRC_ERROR	The CRC obtained from the reader is incorrect.

0015	EZSMC_LRC_ERROR	The LRC obtained from the reader is incorrect.
0016	EZSMC_IO_TIMEOUT	I/O operations timeout
0017	EZSMC_INSUFFICIENT_RESOURCE	The kernel resource is insufficient.
0018	EZSMC_NO_READER_FOUND	No reader is connected.
0019	EZSMC_MORE_PROCESSING_REQUIRED	There is more data to send.
001A	EZSMC_UNRECOGNIZED_MEDIA	Cannot recognize the card in reader or there is no card in reader.
001B	EZSMC_INVALID_DEVICE_REQUEST	Request to the device is invalid.
001C	EZSMC_NOT_SUPPORTED	The upper layer has requested an unsupported function.
001D	EZSMC_CARD_ABORT	ICC aborts the request.
001E	EZSMC_IFD_ABORT	IFD aborts the request.
001F	EZSMC_DEVICE_PROTOCOL_ERROR	ICC returns error.
0020	EZSMC_CONNECT_CARD_FAIL	Some error occurs when connecting to the card.
0021	EZSMC_READER_OPENED	The reader has been opened.
0022	EZSMC_DATA_OVERFLOW	The Internal Buffer too Small.
0023	EZSMC_NO_STATUS_DATA	No return of the status of APDU.
0101	EZSMC_M_CARD_ABSENT	No Memory card is in the reader.
0102	EZSMC_M_NO_RESPONSE	The Memory card does not response.
0103	EZSMC_M_POWER_FAIL	Powering Memory card failed.
0104	EZSMC_M_COMM_ERROR	Communication to Memory Card has error.
0105	EZSMC_M_VERIFY_FAIL	Verification codes fail.
0106	EZSMC_M_TYPE_ERROR	Memory card type error.
0107	EZSMC_M_COMMAND_ERROR	Sending command to the card error.
0108	EZSMC_M_COUNTER_EMPTY	Counter of card is empty.
0109	EZSMC_M_CARD_LOCKED	The card is locked.
010A	EZSMC_M_WRITE_ERROR	Writing to card error.
010B	EZSMC_M_CHECK_ERROR	Previously written data is not really written due to protection.
010C	EZSMC_M_OTHER_FAIL	Other Memory card errors.