LAMPIRAN I

Program Utama

```
unit testerscard;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls, Forms,
  Dialogs, StdCtrls, DB, DBTables, ExtCtrls;

type
  TForm1 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Edit1: TEdit;
    Button6: TButton;
    Button7: TButton;
    Label6: TLabel;
    Edit2: TEdit;
    ComboBox1: TComboBox;
    Label10: TLabel;
    Label12: TLabel;
    Label13: TLabel;
    Edit3: TEdit;
    Label14: TLabel;
    Table1: TTable;
    Table1ID: TStringField;
    Table1Nama: TStringField;
    Table1JenisKendaraan: TStringField;
    Table1NomorPolisi: TStringField;
    Table1Reload: TFloatField;
    Table1Cost: TFloatField;
    Table1Saldo: TFloatField;
    Edit4: TEdit;
    Timer1: TTimer;
    Label1: TLabel;
    Edit5: TEdit;
    Label2: TLabel;
    Button4: TButton;
    Edit6: TEdit;
    Table2: TTable;
    Table2ID: TStringField;
    Table2JamMasuk: TStringField;
    Table2NomorPolisi: TStringField;
    Label3: TLabel;
    Button5: TButton;
```

```
    Label4: TLabel;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);


   procedure Button8Click(Sender: TObject);
   procedure Button7Click(Sender: TObject);

    procedure Button4Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Button10Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button6Click(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure Button5Click(Sender: TObject);

  private
    { Private declarations }
  public
    procedure ambildata;
    procedure kirimdata;
    procedure simpandata;
    { Public declarations }
  end;

var
  Form1: TForm1;
  hcom,h2com : THandle;
  fg,pl : PChar;

  kk,result : integer;
  DateTime : TDateTime;

const GPTR : LongWord = 64;

function CHexToBin( bin : PCHar; asc : PChar; len : word ):
PChar;
stdcall external 'wdcrwv.dll';
function BinToCHex( asc : PChar; bin : PChar; len : word ):
PChar;
stdcall external 'wdcrwv.dll';
function CT_open(portname : string; param : integer):THandle;
stdcall external 'wdcrwv.dll';
function CT_close(fd : THandle): integer;
stdcall external 'wdcrwv.dll';
function GlobalAlloc(wFlags : LongWord;dwBytes : LongWord):
THandle;
stdcall external 'kernel32.dll';
function MEM_SetType(fd : THandle; tipe : Char): Word;
stdcall external 'wdcrwv.dll';
function MEM_GetType(fd : THandle; tipe : PChar): Word;
```

```
stdcall external 'wdcrwv.dll';
function MEM_Reset(fd : THandle; len : string; resp : string):
Word;
stdcall external 'wdcrwv.dll';
function MEM_read_bin(fd : THandle; offset : LongWord;len :
LongWord; resp : PChar ): THandle;
stdcall external 'wdcrwv.dll';
function ICC_present(fd : THandle): word; stdcall external
'wdcrwv.dll';
function ICC_reset(fd : THandle; len : string;resp :string):
word;
stdcall external 'wdcrwv.dll';


implementation

{$R *.dfm}
procedure TForm1.Timer1Timer(Sender: TObject);

var
  str : string;

begin
  DateTime := Time;  // store the current date and time
  str := TimeToStr(DateTime); // convert the time into a
string
  Edit6.Text := str;
  Label1.Caption := str;


end;

procedure TForm1.ambildata;
begin

     if Table1.FindKey([Edit2.Text]) then
      begin
         Edit2.Text := Table1ID.AsString;
         Edit3.Text := Table1Nama.AsString;
         Edit4.Text := Table1Saldo.AsString;
      end
     else
      begin
         Table1.InsertRecord([Edit2.Text]);
      end;
end;
procedure TForm1.kirimdata;

begin
```

```
        Table1.Edit;
        Table1Cost.AsInteger := 1500;
        Table1Saldo.AsInteger := Table1Saldo.AsInteger - 1500;



         Table1.Post;


end;
procedure TForm1.simpandata;
begin


        if Table2.FindKey([Edit2.Text]) then
          begin
           Table2.Edit;
           Table2ID.AsString := Edit2.Text;
           Table2JamMasuk.AsString := TimeToStr(DateTime);
           Table2NomorPolisi.AsString := Edit5.Text;
           Table2.Post;
          end
        else
          begin

Table2.InsertRecord([Edit2.Text,Edit6.Text,Edit5.Text]);
        end;


end;

function ConvertPCharToString(PCharValue: PChar): String;
begin
   Result := StrPas(PCharValue);
end;
procedure TForm1.Button1Click(Sender: TObject);
var

   param : integer;
   ght : word;
begin
        param := 9600;
        hcom := CT_Open(ComboBox1.Text,param);
        ght := ICC_present(hcom);
          Label3.Caption := IntToStr(hcom);
          Label6.Caption := IntToStr(ght);
          if ght = 36864 then
          begin
            Edit1.Text := 'SMART CARD ADA DALAM READER';
          end
         else
```

```
          begin
            Edit1.Text := 'SMART CARD TIDAK ADA DALAM READER';
          end;



end;

procedure TForm1.Button2Click(Sender: TObject);
var
    aa : integer;
begin
      aa := CT_close(hcom);
      Application.Terminate;

end;

procedure TForm1.Button8Click(Sender: TObject);
var
    rst : word;
begin
    rst := ICC_present(hcom);

end;



procedure TForm1.Button7Click(Sender: TObject);
var
    b : PChar;
    c : PChar;
    a : PChar;
    d : PChar;
    h4com : THandle;
begin

    while c^ <> #0 do
       begin
         h4com := MEM_Reset(hcom,b,c);
         Inc(c);
       end;
       if h4com = 36864 then
        begin
           Label6.Caption := 'SUCCESS';
        end
       else
         begin
            Label6.Caption := 'FAILED';
         end;
```

```
end;



procedure TForm1.Button4Click(Sender: TObject);
begin

       simpandata;

end;

procedure TForm1.FormCreate(Sender: TObject);
begin
       ComboBox1.Clear;
       ComboBox1.Items.Add('COM1');
       ComboBox1.Items.Add('COM2');
       ComboBox1.Items.Add('COM3');
       ComboBox1.Items.Add('COM4');
       ComboBox1.Items.Add('USB1');
       ComboBox1.Items.Add('USB2');
       Table1.Open;
       Table2.Open;
end;

procedure TForm1.Button10Click(Sender: TObject);

begin
       ambildata;
end;

procedure TForm1.Button3Click(Sender: TObject);
var
    cc,ff,dd : PChar;
    km : word;
begin
         dd := pl;
         km := strlen(dd);
         ff := BinToCHex(cc,dd,km);
         Edit2.Text := ff;

end;

procedure TForm1.Button6Click(Sender: TObject);
begin
        kirimdata;
end;

procedure TForm1.Button5Click(Sender: TObject);
```

```
var
    kk : PChar;
    jj : PChar;
    ll : THandle;
    ii : word;
begin
       for ii:= 00 to 02 do
         begin
           ll := MEM_read_bin(hcom,ii,03,kk);

         end;
         Label4.Caption := IntToHex(ll,2);
         if ll = 36864 then
           begin
             pl := kk;
           end
         else
           begin
              Label4.Caption := 'FAILED';
           end;
end;
end.
```

LAMPIRAN II

Program driver smart card Reader CRW-VI

```
CRW-VI series Card Reader/Writer Dynamic Library Functions
WDCRWV.H           Version 3.0
Suitable Card Reader/Writer:
* CRW-V series and CRW-VI series Card Reader/Writer

Suitable Operating System:
- Serial Port Reader/Writer: DOS, WINDOWS95 and above
- USB Reader/Writer: WINDOWS98(ver 2) and above, WINDOWS(ME)
Suitable IC Cards:
- CPU Cards that are compliant to ISO 7816-1 2 3 4 (T=0,1)
- I2C bus memory card
- SIEMENS- SLE4406 SLE4428 SLE4432 SLE4442
- ATMEL - AT24C01/02/04/08/16/32/64  AT88SC102 AT1604  AT1608
AT45D041

Note: Functions names that end with "VB" is used for Visual
Basic environment                                              --
-------------------------------------------------------------------
-----------------------------------------------------*/
#include <windows.h>

#ifdef __cplusplus
extern "C"{
#endif
/*-------------------------------------------------------------------
------------------------------
I¡¢Functions for CPU Card and Card Reader
-------------------------------------------------------------------
------------------------------
 1.  Opening a Serial Port or USB Port connecting to the IC
Card Reader  */
     HANDLE  WINAPI  CT_open(char *portname,unsigned int
param);                                    /*
Parameters-
portname: Name of the port connecting to the card reader.
Serial port card reader: " COM 1", " COM 2", " COM 3","COM4"
USB Card Reader: "USB1", "USB2","USB3","USB4"
param:  Serial port card reader: baud rate,9600bps,38400bps
USB Card Reader: mode of open Card Reader  1: share mode 2:
special mode
Return:   Less than 0 means failure to open the port
Greater than 0 means successful return of the descriptor of
the opened port, to be used for the operational function of
the access card
```

```
 ------------------------------------------------------------
--------------------------------
 2. Close a serial port or USB port connecting to the IC card
reader (the port must be opened with CT_open)
*/
int WINAPI CT_close(HANDLE fd);
/*
Parameters:    fd is the port descriptor returned by the
function CT_open.
Return:        -1 means failure, and 0 means success.
 ------------------------------------------------------------
------------------------------
 3. Reset IC Card
*/
    unsigned WINAPI ICC_resetVB(
                HANDLE fd,
                 int *len,
                 unsigned char *resp);
    unsigned WINAPI ICC_reset(
               HANDLE fd,
                 unsigned char *len,
                 unsigned char *resp);
/*
 Parameters-
fd:  the opened port descriptor.
len:      length of the returned data for resetting the IC
card
    resp: result of the reset data
Return:   0x9000    successful
          0x6200    no card
          0x6201    unknown protocol
               0xffff    communication failure
 ------------------------------------------------------------
------------------------------
  4.  Check Whether card is in Card Reader or not
*/
  unsigned WINAPI ICC_present(HANDLE fd);
/*
  Parameters  fd: the opened port descriptor.
    Return     0x9000  card is in Card Reader
                 0x6200  no card in Card Reader
 ------------------------------------------------------------
-----------------------------------
  5. Send a command to CPU card or card reader and receive the
response data
Structure of comm: CLA INS P1 P2 Lc DATA [Le]  where the DATA
length is Lc bytes.
Structure of resp: DATA,  where the DATA length is Le bytes
*/
    unsigned WINAPI ICC_tsi_apiVB(
                HANDLE fd,
```

```
                int len,
                unsigned char *comm,
                int *lenr,
                unsigned char *resp);

    unsigned WINAPI ICC_tsi_api(
                HANDLE fd,
                unsigned char len,
                unsigned char *comm,
              unsigned char *lenr,
                unsigned char *resp);
/*
         Parameters:     fd:     the opened port descriptor.
                         len:      length of the command comm
comm:  command sent to the card
                         lenr:   length of data received from
the card
                         resp:   data received from the card
 Return:
  0xffff communication failure (failure to send the command or
receive the returned data).
Others being the status returned from the card SW1 SW2.
 NOTE: This function is suitable for all CPU card
  ------------------------------------------------------------
----------------------------------

6. Send a command to CPU card (T=0) and receive the response
data
Structure of comm: CLA INS P1 P2 Lc DATA [Le]  where the DATA
length is Lc bytes.
Structure of resp: DATA, where the DATA length is Le bytes
*/
       unsigned WINAPI ICC_tsi_apiT0VB(
                  HANDLE fd,
                    unsigned int len,
                    unsigned char *comm,
                unsigned int *lenr,
                    unsigned char *resp);

       unsigned WINAPI ICC_tsi_apiT0(
                  HANDLE fd,
                    unsigned int len,
                    unsigned char *comm,
                unsigned int *lenr,
                    unsigned char *resp);
/*
Parameters:    fd:     the opened port descriptor.
               len:    length of the command
comm:  command sent to the card
               lenr:   length of data received from the card
               resp:   data received from the card
```

```
Return:  0xffff communication failure (failure to send the
command or receive the returned data).
Others being the status returned from the card SW1 SW2.
NOTE : This function is only suitable for read/write CPU card
which supports T=0,  and the biggest transfer memory sizes of
the card should be 256 bytes.
As for the function ICC_tsi_api( ), it can only read 253 bytes
and write 250 bytes optimally at one time.

 ------------------------------------------------------------
---------------------------------
   7. Set the CPU card access address NAD
*/
    void WINAPI ICC_set_NAD(HANDLE fd,unsigned char nad);
/*
     Parameters:    fd:     the opened port descriptor.
NAD  access address
Note:  If it is not set, the system default is 0x12  -send
command to User Card or card reader
NAD=0x13 - send command to SAM Card

------------------------------------------------------------
---------------------------------

 8. Đ´CPU¿¨µÄ¶þ½øÖÆÎÄ¼þ
*/

      unsigned WINAPI ICC_write_file(
          HANDLE   fd,
           unsigned int offset,
           unsigned int len,
           unsigned char *data);
/*

       ²ÎÊý:
       fd     : ÒÑ´ò¿ªµÄ¶Ë¿ÚÃèÊö·û.
       offset : ¶þ½øÖÆÎÄ¼þµÄÆ«ÒÆÁ¿
       len    : ÒªÐ´Èë¿¨ÉÏµÄÊý¾Ý³¤¶È
       data   : ÒªÐ´Èë¿¨ÉÏµÄÊý¾Ý
     ·µ»Ø:
          0xffffÍ¨Ñ¸Ê§°Ü(·¢ËÍÃüÁî»ò½ÓÊÕ·µ»ØµÄÊý¾ÝÊ§°Ü.
        ÆäËüÎª´Ó¿¨ÉÏ·µ»ØµÄ×´Ì¬SW1 SW2
        ËµÃ÷:1.¸Ã¯¯ÊýÊÊÓÃÓÚËùÓÐCPU¿¨²Ù×÷
              2. ÓÃ»§±ØÐëÏÈÑ¡ÔñÒ»¶þ½øÖÆÎÄ¼þ


 ------------------------------------------------------------
---------------------------------

      9. ¶ÁCPU¿¨µÄ¶þ½øÖÆÎÄ¼þ
*/
```

```
          unsigned WINAPI ICC_read_file(
                      HANDLE  fd,
                      unsigned int offset,
                      unsigned int len,
                      unsigned char *data);
/*


        ²ÎÊý:
        fd     : ÒÑ´ò¿ªµÄ¶Ë¿ÚÃèÊö·û.
        offset : ¶þ½øÖÆÎÄ¼þµÄÆ«ÒÆÁ¿
        len    : Òª¶Á¿¨ÉÏµÄÊý¾Ý³¤¶È
        data   : Òª¶Á¿¨ÉÏµÄÊý¾Ý
      ·µ»Ø:
           0xffffÍ¨Ñ¸Ê§°Ü(·¢ËÍÃüÁî»ò½ÓÊÕ·µ»ØµÄÊý¾ÝÊ§°Ü.
         ÆäËüÎª´Ó¿¨ÉÏ·µ»ØµÄ×´Ì¬SW1 SW2
         ËµÃ÷:1.¸Ã¯ÊýÊÊÓÃÓÚËùÓÐCPU¿¨²Ù×÷
               2. ÓÃ»§±ØÐëÏÈÑ¡ÔñÒ»¶þ½øÖÆÎÄ¼þ

 ----------------------------------------------------------------
--------------------------------
II Functions for Memory Card

NOTICE
1.The performance of all kinds memory card is different,
please select right memory card when developing application
system, and ensure that The card reader supports it.
 2. Pls Reset IC Card Before read/write
 3. CRW-V /CRW-VI series can not correctly identify
AT24C01/02£¬when run MEM_Reset( ) or MEM_GetType( ),The zero
byte will be written, so The user data can not be stored in
zero byte.
 4. Function return code description
   0x9000,operation success
   0x6B00, Wrong offset
   0x6a86,  Wrong len
   0x6282 offset+len exceed card capacity or incorrect data
read
   0x63cx incorrect PIN, x retries possible
   0x6983 No retry possible
   0x6581 incorrect data wrote
   0x6ff0 System error
 ----------------------------------------------------------------
--------------------------------
   1. Reset memory Card
*/
   unsigned  WINAPI MEM_Reset(HANDLE fd, unsigned char
*len,unsigned char *resp);              /*
Parameters:    fd:     the opened port descriptor.
               len:    length of the received data.
resp:   data received from the card.
```

```
   NOTE
   1. PLS call MEM_Reset ( ) first before operate memory card

   2. AT24C01/02can not be automatically determined, Pls first
call MEM_Reset( ) and MEM_SetType then read/write
  ------------------------------------------------------------
----------------------------------
     2. Automatically Determine The Card Type
*/
   unsigned  WINAPI MEM_GetType(HANDLE fd,unsigned char
*Type);                                /*
Parameters:    fd:     the opened port descriptor.
               Type:  code return from the card.

NOTE
For memory card that are not compliant to ISO 7816- 3, The
card reader will automatically adjust internal status, after
re-insert card, must transfer this function. This function can
identify almost all general memory card
        Type           Return code of memory card type
#define I2CBUS      0x08           //AT24C01/02/04/08/16 and
compatible I2C card /
#define SLE441828   0x09           //3-wire bus 4418/ 4428 /
#define SLE443242   0x0A           //2-wire bus 4432/4442/
#define SLE440306   0x0C           //SLE4406/
#define AT102           0x0F            //ATMEL 102/
#define AT1604      0x12           //ATMEL 1604/
#define I2CBUS64        0x13              //AT24C32/64 I2C
card/
#define AT45D041        0x14           //512k byte card/
#define AT1608      0x15           //ATMEL 1608/
#define UNKNOW      0xff or 0x00        s//unknown card/
  ------------------------------------------------------------
----------------------------------
 3. Set The Card Type
*/
unsigned  WINAPI MEM_SetType(HANDLE fd, unsigned char Type);
/*
Parameters:    fd:     the opened port descriptor.
               Type:  code of the card type want set.

NOTE
When the reader meets a memory card and can't determine the
card type automatically, First reset the card, then use this
function.

  ------------------------------------------------------------
----------------------------------
 4. read memory card
*/
   unsigned WINAPI MEM_read_bin(
```

```
            HANDLE fd,
               long offset,
               unsigned int len,
               unsigned char *resp);
/*
Parameters:    fd:     the opened port descriptor.
               offset:  absolute address of memory card.
               len:    length of data want to read out.
resp:   data read out the card.

  ------------------------------------------------------------
----------------------------------
   5. read memory card with protect bit (SLE4432/42/18/28)
*/
   unsigned WINAPI MEM_read_bin_p(
            HANDLE fd,
               unsigned int offset,
               unsigned int len,
               unsigned char *resp);
/*
Parameters:    fd:     the opened port descriptor.
               offset:  absolute address of memory card.
               len:    length of data want to read out.
resp:   data read out the card. After each byte data with
protect bit explain.
                         01 00 02 00 03 00 04 01 05 01 06 01
07 01 08 01
first three bytes can't write again, other five bytes can
write again.

NOTE
     This function be suit for SLE4418/28,SLE4432/42.

     ------------------------------------------------------------
----------------------------------
  6. bit read memory (SLE4406 AT88SC102 1604)
*/
  unsigned WINAPI MEM_read_bit(
           HANDLE fd,
             unsigned int offset,
             unsigned int len,
             unsigned char *resp);
/*
Parameters:    fd:     the opened port descriptor.
               offset:  absolute address of memory card.
               len:    length of data want to read out.
resp:   data read out the card.

 ------------------------------------------------------------
------------------------------
```

```
  7. write memory
*/
  unsigned WINAPI MEM_write_bin(
           HANDLE fd,
             long offset,
             unsigned int len,
             unsigned char *data);
/*
Parameters:    fd:     the opened port descriptor.
               offset:  absolute address of memory card.
               len:    length of data should be wrote.
resp:   data write in the card.

NOTE
1.   When the valid data length is less than len, uncertain
data would be wrote in the shot area.
2.   When offset +len bigger than the capacity of the card,
use this function maybe appear uncertain phenomena.
3.   Since each memory card will has a different meaning when
implement writing process in each certain area, Developers are
supposed to look for the technical data of each memory card.
4.   this function is unsuited for SLE4406.

  ----------------------------------------------------------
------------------------------
  8. write memory with protect bit or melt fuse
*/
  unsigned WINAPI MEM_write_bin_p(
           HANDLE fd,
             unsigned int offset,
             unsigned int len,
             unsigned char *data);
/*
Parameters:    fd:     the opened port descriptor.
               offset:  absolute address of memory card.(
address of melt fuse ,when melt fuse)
               len:    length of data should be write.
(insignificance, when melt fuse)
resp:   data write in the card. (insignificance, when melt
fuse)

NOTE
1.   The data wrote by the function cannot be revised
(SLE4432/42/18/28).
2.   resp is made up of tow parts: data and protect bit(0x00
or 0x01). 0x00 is show this byte can not write again, 0x01 is
show this byte can write again. For example
12003400560178019a00, this is five byte data +five protect
bit.
3.   len is the length of the data.
```

```
4.   SLE4418/28,SLE4432/42 use this function to write with
protect bit
5.   AT88SC102/1604/1608 use this function to melt fuse.
6.   AT88SC1608 melt fuse FAB\CMA\PER orderly. Parameter
offset is insignificant.
7.   We do not verify This function's melt fuse.

  ------------------------------------------------------------
--------------------------------
  9. bit write memory (SLE4406 AT88SC102 1604)
                          */
  unsigned WINAPI MEM_write_bit(
           HANDLE fd,
             unsigned int offset,
             unsigned int len,
             unsigned char *data);
/*
Parameters:    fd:     the opened port descriptor.
               offset:  absolute address of memory card.
               len:    length of data want to write.
resp:   data write in the card. data is made of two parts:
0x00 and 0x01. Show this bit's logistic voltage.

NOTE
1.   After verify write pin, continuous write OK
2.   SLE4406's 96~103 bit can use this function write 0x00.
3.   When the true data length less than the input length
(len), write uncertain data in shot area.

  ------------------------------------------------------------
--------------------------------
  10. bit erasure (SLE4406 AT88SC102 1604 )
*/
  unsigned WINAPI MEM_erase_bit(HANDLE fd,unsigned int
offset);                                          /*
Parameters:    fd:     the opened port descriptor.
               offset:  absolute address of memory card.
NOTE
1.   SLE4406's 64~103 bit can use this function, the bit
become 0x00 and after this one's bits become 0x01.
2.   Other cards use this function by byte.

    ------------------------------------------------------------
--------------------------------
  11. verify pin
*/
  unsigned WINAPI MEM_verify(HANDLE fd,unsigned char
len,unsigned char *pin);                     /*
Parameters:    fd:     the opened port descriptor.
               len:    length of the pin.
               pin:    password.
```

NOTE

If the card have several password, len as follow:

For example SLE4404 card£¬main user password's len is 02, erasure password's len 14,

Relation of AT88SC102's password and len as follow:

len	password
02	main password
16	AZ1 erasure password
24	AZ1 erasure password
34	AZ1 erasure password (EC2=0)

Relation of AT88SC1604's password and len as follow:

	password	erasure password
main area	02	
Area1	12	22
Area 2	32	42
Area 3	52	62
Area 4	72	82

Relation of AT88SC1608's password and len as follow:£º

	write password	read password
0	03	83
1	13	93
2	23	A3
3	33	B3
4	43	C3
5	53	D3

```
          --------------
          |    6    |   63      |   E3       |
          --------------
          |    7    |   73      |   F3       |

  ------------------------------------------------------------
------------------------------------------
  12. change pin
*/
  unsigned WINAPI MEM_change_pin(
           HANDLE fd,
             unsigned char pin_len,
             unsigned char *oldpin,
             unsigned char *newpin);
/*
Parameters:    fd:     the opened port descriptor.
               pin_len: length of the pin.
          oldpin : old password
          newpin : new password

NOTE
1.   This function is only use for main password. Other
password use MEM_verify_pin function verify and erasure
password, then use MEM_write_bin function write password in
the area.
2.   This function is unsuited for AT88SC1608.

   ------------------------------------------------------------
----------------------------------
  13.     AT88SC1608 special verify function

                          */
    int WINAPI MEM_AT1608_Auth(
         HANDLE fd,
          unsigned char Q0[8],
          unsigned char Gc[8]);
/*
Parameters:    fd:     the opened port descriptor.
Q0[8]:  8 bytes random number.
               Gc[8]:  8 bytes key. The key can read from 0x30
byte of the setting area, before the card out of the provider.
Otherwise the key give out with the provider

------------------------------------------------------------
----------------------------------
  14.     AT88SC102/1604 card special, change logistic voltage
of FUSE
                                                          */
     int WINAPI MEM_AT88_SetFusepin(unsigned char level);
/*
Parameters:    level:  0x00 set FUSE as low.
```

```
                    0x01 set FUSE as high.
NOTE
     Change AT88SC102/1604 FUSE logistic voltage low or high,
just change security level.
-----------------------------------------------------------------
--------------------------------
III,Other Functions & Encryption Functions

  ---------------------------------------------------------------
--------------------------------
   1. Converting 16 hexadecimal string to binary data
*/
   unsigned char * WINAPI CHexToBin(
            unsigned char *bin,
               unsigned char *asc,
               unsigned int len);
/*
Parameters:    bin       Binary string results :
0x12,0x34,0xE1,0xFA
                  asc       Hexadecimal string eg"1234E1FA"
               len       Length of hexadecimal string
       ---------------------------------------------------------
-----------------------------------
   2. Converting Binary data into 16 Hexadecimal string
*/
   unsigned char * WINAPI BinToCHex(
            unsigned char *asc,
               unsigned char *bin,
               unsigned int len);
/*
   Parameters:   asc          Hexadecimal string eg"1234E1FA"
bin       Binary string results : 0x12,0x34,0xE1,0xFA
                    len:          Length of Binary string

  ---------------------------------------------------------------
--------------------------------
   3. SHA
void  WINAPI SHA1M(unsigned char *sour, unsigned len,unsigned
char *digest)
BOOL WINAPI SHA1MVB(unsigned char *sour, unsigned len,unsigned
char *digest)
Parameters:    len       length of original text to be
compressed
Sour      Original text to be compressed
          comdata   Compressed data of 20 bytes
  ---------------------------------------------------------------
--------------------------------
   4. DES Encryption
*/
   unsigned WINAPI SingleDES(char DESType,
                    unsigned char * SingleDESKey,
```

```
                        unsigned int  SourDataLen,
                        unsigned char *SourData,
                        unsigned char *DestData);
   unsigned WINAPI SingleDESVB(
                    char DESType,
                        unsigned char * SingleDESKey,
                        unsigned int SourDataLen,
                        unsigned char *SourData,
                        unsigned char *DestData);
/*

Parameters:    DESType:  =1 Encryption
=2 Decryption
SingleDESKey:  8 bytes Key
SourDataLen:        Length of plaintext
SourData:      Plaintext
DestData:          Encrypted text
Note: When the length of the text is not in multiples of 8,
this function will add a hexadecimal string of "80 00 00 ¡-."
to the end of the text to make it a multiple of 8 before
encryption.
    Return: the length of the encrypted text
 ---------------------------------------------------------------
------------------------------
   5. 3DES Encryption
*/
   unsigned WINAPI TripleDES(
              char DESType,
                unsigned char * TripleDESKey,
               unsigned int  SourDataLen,
                unsigned char *SourData,
                unsigned char *DestData);

   unsigned  WINAPI TripleDESVB(
               char DESType,
                 unsigned char * TripleDESKey,
                 unsigned int  SourDataLen,
                 unsigned char *SourData,
                 unsigned char *DestData);

                                                      /*
Parameters:    DESType:       =1 Encryption
=2 Decryption
                    TripleDESKey:   16 bytes Key K1K2
SourDataLen         Length of plaintext
SourData:      Plaintext
DestData:          Encrypted text
Note: When the length of the text is not in multiples of 8,
this function will add a hexadecimal string of "80 00 00 ¡-."
to the end of the text to make it a multiple of 8 before
encryption. The Encryption process is as follows:
```

```
DES3-E({K1,K2},P)=E(K1,D(K2,E(K1,P)))  DES3-
D({K1,K2},C)=D(K1,E(K2,D(K1,P)))
Return: the length of the encrypted text
 ------------------------------------------------------------
------------------------------
6.DES MAC
*/
     unsigned WINAPI SingleMAC(
                unsigned char * SingleMACKey,
                 unsigned char *InitData,
                 unsigned int  SourDataLen,
                 unsigned char *SourData,
                 unsigned char *MACData);

     unsigned WINAPI SingleMACVB(
               unsigned char * SingleMACKey,
                 unsigned char *InitData,
                 unsigned int  SourDataLen,
                 unsigned char *SourData,
                 unsigned char *MACData);
                                              /*

Parameters:    SingleMACKey:  8 byte Key
InitData:           8 byte initial value
SourDataLen:        Length of original text used to generate
mac code
SourData:      Original text used to generate mac code
MACData:       Calculated MAC
Note: See the corresponding standard for the calculation of
mac
Return: the length of Calculated MAC is 8 bytes
 ------------------------------------------------------------
------------------------------
   7. 3DES MAC
*/
    unsigned WINAPI TripleMAC(
               unsigned char * SingleMACKey,
                unsigned char *InitData,
                unsigned int  SourDataLen,
                unsigned char *SourData,
                unsigned char *MACData);

     unsigned WINAPI TripleMACVB(
               unsigned char * TripleMACKey,
                unsigned char *InitData,
                unsigned int  SourDataLen,
                unsigned char *SourData,
                unsigned char *MACData);
                                             /*

Parameters:    TripleMACKey:  8 byte Key
```

```
InitData:           8 byte initial value
SourDataLen:        Length of original text used to generate
mac code
SourData:      Original text used to generate mac code
MACData:       Calculated MAC
*/


#ifdef __cplusplus
}
#endif
```