LAMPIRAN A
FOTO ALAT

Foto Alat Pembacaan / Penulisan Kartu Tipe AET60

Foto 3 buah *Smart Card* ACOS2

LAMPIRAN B

REFERENCE MANUAL ACOS2

Advanced Card Systems Ltd.

ACOS2 Smart Card

INTRODUCTION

SCOPE

The purpose of this document is to describe in detail the features and functions of the ACS Smart Card Operating Systems Version 2 (ACOS2) developed by Advanced Card Systems Ltd.

Features

ACOS2 provides the following features:

- Full 8 Kbytes of EEPROM memory for application data
- Compliance with ISO 7816-3, T=0 protocol
- High-speed transmission possible with modifiable ATR.
- DES and MAC capabilities
- Five secret codes + Issuer Code
- PIN code that can be updated by card holder
- Key pair for mutual authentication
- Session key based on random numbers
- Linear files with fixed record length; record length can be different for different files
- Account data structure for highly secure payment applications as an optional function
- Completely backward compatible with ACOS1 cards.

EEPROM MEMORY MANAGEMENT

The 8 k Bytes EEPROM memory area provided by the card chip is basically segregated in Internal Data Memory and User Data Memory:

- The Internal Data Memory is used for the storage of configuration data and it is used by the card operating system to manage certain functions.
- The User Data Memory stores the data manipulated in the normal use of the card under control of the application.

Data Files

Access to both the Internal Data Memory area and the User Data Memory area is possible within the scopes of data files and data records. Data files in the Internal Data Memory are referred to as *Internal Data Files*. Data files in the User Data Memory are called *User Data Files*.

Data files are the smallest entity to which individual security attributes can be assigned to control the read and write access to the data stored in the EEPROM.

Data files are composed of data records. A data record is the smallest data unit that can individually be addressed in a data file. Each data file contains N data records. The record number must be specified when a record (or data within a record) is read from or written to a file. A data file can contain up to 255 records. The record length can be different for different files but is always fixed within a file.

The file structures of the Internal Data Files (file size, file identifier, record length, security attributes) are defined by the operating system and cannot be changed. The file structure for the User Data Memory is determined in the card personalization. After programming the parameter N_OF_FILE in the Personalization Stage, the file structure is fixed.

Access to all files is possible only through the READ RECORD and WRITE RECORD commands. The operating system does not keep track of which records have actually been written through the WRITE RECORD command. The data returned by the card in response to a READ RECORD command are the actual data read from the EEPROM memory, regardless of whether that data have ever been written.

Each file is identified by two bytes File Identifier. The File Identifier is assigned to the file when the file is being defined during the Personalization Stage. The operating system does not perform any checking on the uniqueness of each File Identifier. If the same identifier has been assigned to more than one file, a malfunction of the card may occur.

A value of FF_H of the first byte of the file identifier is used for Internal Data Files and cannot be used for User Data Files.

Before any READ RECORD or WRITE RECORD access to a file, the file must be opened through the SELECT FILE command. Only one file is selected at any time. The READ RECORD and WRITE RECORD commands refer to the most recently selected file.

Internal Data Files

With exception of the Account Data Structure, which has associated a special set of commands, the memory areas of the Internal Data Memory are processed as data files.

The attributes of the Internal Data Files are defined in the card operating system and cannot be changed. However, the security attributes depend on the card life cycle stage.

The following Internal Data Files are defined:

		File Security Attributes			Record Organization
Memory Area	**Internal File ID**	**Manufacturing Stage**	**Personalization Stage**	**User Stage**	
MCU-ID File	$FF\ 00_H$	R: FREE W: NO ACCESS	R: FREE W: NO ACCESS	R: FREE W: NO ACCESS	2 x 8 bytes
Manufacturer File	$FF\ 01_H$	R: FREE W: IC	R: FREE W: NO ACCESS	R: FREE W: NO ACCESS	2 x 8 bytes
Personalization File	$FF\ 02_H$	R: FREE W: IC	R: FREE W: IC	R: FREE W: NO ACCESS	3 x 4 bytes
Security File	$FF\ 03_H$	R: IC W: IC	R: IC W: IC	R: NO ACCESS W: IC	12 x 8 bytes
User File Management File	$FF\ 04_H$	R: FREE W: IC	R: FREE W: IC	R: FREE W: IC	N_OF_FILE x 6 bytes
Account File	$FF\ 05_H$	R: FREE W: IC	R: FREE W: IC	R: IC W: IC	8 x 4 bytes
Account Security File	$FF\ 06_H$	R: FREE W: IC	R: FREE W: IC	R: NO ACCESS W: IC	4 x 8 bytes
ATR File	$FF\ 07_H$	R: FREE W: IC	R: FREE W: IC	R: FREE W: IC	1 X 36 bytes
User File Data Area	file IDs: $xx\ yy_H$ $xx \neq FF_H$	According to the file definitions			

ISO COMPLIANCE AND ANSWER-TO-RESET

After a hardware reset (e.g., power up), the card transmits an Answer-To-Reset (ATR) in compliance with the standard ISO 7816, part 3. ACOS2 supports the protocol type T=0. The protocol type selection function is not implemented.

The direct convention is used for the coding of the bits in the communication with the card, i.e., logic level ONE corresponds to the Z state of the I/O line.

Fourteen bytes of data are transmitted in the historical bytes as described below.

The following data are transmitted in the ATR:

TS	T0	TA_1	TB_1	TD_1	14 Historical Bytes
$3B_H$	BE_H	11_H	00_H	00_H	

The 14 bytes string transmitted in the historical bytes is composed as follows:

T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14
41_H	01_H	10_H 20_H 38_H	Option registers			Personalization File bytes 4 - 8					Life-cycle Stage	90_H	00_H

Lifecycle Stage	Codes the current card lifecycle stage in a single byte 0 : User Stage 1 : Manufacturing Stage 2 : Personalization Stage
Version Bytes	The contents of the version bytes are: T1 = 41_H = ACOS T2 = 01_H = Version 1 T3 = 10_H / 20_H / 38_H = Revision 1.0 / Revision 2.0 / Revision 3.8
Option Registers	The contents of the three bytes option registers: T4 = Option Register T5 = Security Option Register T6 = N_OF_FILE
Personalization File memory	5 bytes following the Option Registers of the Personalization File in the EEPROM

COMMANDS

The following section describes in detail the format of all ACOS2 commands and the possible responses.

The command descriptions use the TPDU representation. All numeric values are given in HEX.

A summary of the status codes returned by the card is given in *8. STATUS CODES*.

The following commands are provided by ACOS2:

START SESSION	Start the Mutual Authentication Process
AUTHENTICATE	Authenticate the Card Accepting Device, authenticate the card and compute the Session Key
SUBMIT CODE	Submit a secret code
SELECT FILE	Select a data file for reading and writing
READ RECORD	Read data from a record of the currently open data file
WRITE RECORD	Write data to a record of the currently open data file
INQUIRE ACCOUNT	Read the balance and other Account information
CREDIT	Credit the Account
DEBIT	Debit the Account
REVOKE DEBIT	Revoke the preceding Debit transaction
CHANGE PIN	Change the PIN secret code
GET RESPONSE	Get response data available in the card

START SESSION

To read a random number from the card and start the mutual authentication process the result of which is the Session Key K_S.

Command:

CLA	INS	P1	P2	P3
80	84	00	00	08

Response:

Data	SW1 SW2
RND_C	Status

RND_C Eight bytes card random number

Status Codes:

SW1	SW2	Meaning
69	83	Terminal Authentication Key K_T is locked, authentication process cannot be executed

SUBMIT CODE

To submit a secret code - Application Code, PIN or Issuer Code - to the card.

Command:

CLA	INS	P1	P2	P3	DATA
80	20	Code No.	00	08	Code or DES(Code,#K_S)

Code No. Code reference: 1...5 = AC1...AC5
6 = PIN
7 = IC

Code Eight bytes Code

DES(Code,#K_S) Eight bytes Code encrypted with Session Key

NOTE: If the corresponding option bit xx_DES in the Security Option Register is set, code XX is submitted DES encrypted. If the option bit is not set, the code is submitted in plain without encryption.

NOTE: DES shall be 1-DES or 3-DES depending on the selection in *Option* Register

Response:

SW1 SW2
Status

Specific Status Codes:

SW1	SW2	Meaning
63	Cn	Wrong Code; n = remaining number of re-tries
69	83	The specified Code is locked
69	85	Mutual Authentication not successfully completed prior to the SUBMIT CODE command

SELECT FILE

To select a data file for subsequent READ RECORD and WRITE RECORD commands.

Command:

CLA	INS	P1	P2	P3	DATA
80	A4	00	00	02	File ID

File ID Two bytes file identifier

Response:

SW1 SW2
Status

Specific Status Codes:

SW1	SW2	Meaning
6A	82	File does not exist
91	xx	File selected (only for User Data Files) xx is the number of the record in the User File Management File (file ID: FF 04_H) which contains the File Definition Block of the selected file

READ RECORD

To read a number of bytes - up to the record length - from one record in the currently selected file.

Command:

CLA	INS	P1	P2	P3
80	B2	Rec No.	00	Len

Rec No. Logical record number to be read.
0..N-1 if RECORD_NUMBERING flag in Manufacturer file is zero
1..N if RECORD_NUMBERING flag in Manufacturer file is one

Len Number of data bytes to be read from the record Rec No.

Response:

Data	SW1 SW2
Byte 1 ... Byte N	Status

Byte 1 ... Byte N Data bytes read from the record

Specific Status Codes:

SW1	SW2	Meaning
69	82	Security status not satisfied - Secret code(s) not submitted
6A	83	Record not found - file too short
67	00	Specified Len is larger than record length
69	85	No file selected
6F	00	I/O error; data to be accessed resides in invalid address

WRITE RECORD

To write a number of bytes - up to the record length - to one record in the currently selected file.

Command:

CLA	INS	P1	P2	P3	DATA
80	D2	Rec No.	00	Len	Byte 1 ... Byte N

Rec No.	Logical record number to be read. 0..N-1 if RECORD_NUMBERING flag in Manufacturer file is zero 1..N if RECORD_NUMBERING flag in Manufacture file is one
Len	Number of data bytes to be written to the record *Rec No.*
Byte 1 ... Byte N	Data bytes to be written to the first *Len* bytes of the record

Response:

SW1 SW2
Status

Specific Status Codes:

SW1	SW2	Meaning
69	82	Security status not satisfied - Secret code(s) not submitted
6A	83	Record not found - file too short
67	00	Specified Len is larger than record length - invalid
69	85	No file selected
6F	00	I/O error; data to be accessed resides in invalid address

CARD PERSONALIZATION

This section describes the general procedure in the personalization of an ACOS2 smart card. While the card personalization may be carried out in separate processing steps, the personalization process generally requires the execution of the steps described below.

The personalization of a new ACOS2 smart card is suggested to be carried out according to the following sequence:

1. Power up and reset the card
2. Submit the default Issuer Code IC (the code is communicated to the card issuer by ACS; the code may be different for different batches of cards supplied)
3. Select Manufacturer File (File ID = FF 01_H) and set the related flags in the 1st byte of the 1st record.
4. Select the Personalization File (File ID = FF 02_H) and write the required settings to the Option *Register* and the parameter N_OF_FILE. **Caution: Do not accidentally set the Personalization Bit and do not change the *Security Option Register* at this stage!**
5. Perform a card reset. After the reset, ACOS2 reads the Personalization File and accepts the new value of N_OF_FILE and the option bits stored in the *Option Register,* as well as the new settings in the Manufacturer File.
6. Submit the default Issuer Code IC.
7. Select the User File Management File (File ID = FF 04_H) and write the File Definition Blocks for the required User Files (WRITE RECORD command) with the security attributes set to 'Free Access'.
8. Select the individual User Files and initialize the data in the files as required (WRITE RECORD command).
9. Select the User File Management File (File ID = FF 04_H) and write the required security attributes for all User Files (WRITE RECORD command). Verify the contents of the User File Management File (READ RECORD command). **Caution: Do not accidentally change the other parameters in the File Definition Blocks.**
10. If applicable, select the Account File (File ID = FF 05_H) and initialize the relevant data in the Account File (WRITE RECORD command). Verify the contents of the Account File (READ RECORD command).

11. If applicable, select the Account Security File (File ID = FF 06_H) and initialize the account processing keys (WRITE RECORD command). Verify the contents of the Account Security File (READ RECORD command).

12. Select the Security File (File ID = FF 03_H) and initialize all keys and codes (WRITE RECORD command). Verify the contents of the Security File (READ RECORD command)

13. Select the Personalization File (File ID = FF 02_H) and initialize the *Security Option Register* and the remaining bytes of the Personalization File. **Set the Personalization Bit** (WRITE RECORD command). Verify the contents of the Personalization File (READ RECORD command). **Caution: Do not accidentally change the value of the *Option Register* and N_OF_FILE.**

14. Perform a card reset. The chip life cycle stage as indicated in the ATR should be 'User Stage'.

15. The correct personalization can be verified by submitting the secret codes and keys programmed in the card (AUTHENTICATE, SUBMIT CODE commands) and reading/writing the allocated data files and executing the Account commands.

STATUS CODES

The following is a summary of the status codes returned by the card.

SW1	SW2	Meaning
90	00	O.K.
91	nn	User Data File has been selected. The corresponding File Definition Block is stored in record no. nn in the File Management File
61	nn	O.K. - Issue GET RESPONSE command with L_e = nn to get response data
62	81	Data returned in response to the INQUIRE ACCOUNT command may be incorrect due to corrupted data in the Account Data Structure
63	Cn	Security related command failed - EXTERNAL AUTHENTICATION failed; incorrect Secret Code submitted; incorrect key used in CREDIT MAC generation; n = number of remaining trials
67	00	Wrong P3
69	66	Command not available (Manufacturing Stage, option bit not set, etc.)
69	82	Security status not satisfied - Secret Code, Issuer Code or PIN not submitted
69	83	Key or Secret Code is locked - no more verification or submission possible
69	85	Conditions of use not satisfied - no data for GET RESPONSE command available; CREDIT/DEBIT command executed without previous START TRANSACTION; Mutual Authentication not completed; no file selected
69	F0	Account data inconsistent / transaction interrupted - access to account only in privileged mode possible
6A	82	File does not exist; account not available
6A	83	Record not found - file too short
6A	86	P1-P2 incorrect
6B	20	Invalid amount in CREDIT/DEBIT command
6C	nn	Issue GET RESPONSE command with P3 = nn to get response data
6D	00	Unknown INS
6E	00	Invalid CLA
6F	10	Account Transaction Counter at maximum - no more DEBIT or CREDIT transaction possible

LAMPIRAN C
LISTING PROGRAM

```
Dim con As New ADODB.Connection
Dim cmd As New ADODB.Command
Dim rs As New ADODB.Recordset
Option Explicit
Dim scInit, rdrLocked As Boolean
Dim retCode, numDev, numRdr, hContext, hCard, Protocol, RetLength As Long
Dim dwActProtocol, RecvLen, SendLen As Long
Dim ioRequest As SCARD_IO_REQUEST
Dim FpTemplate(0 To 767) As Byte
Dim SendBuff(0 To 262) As Byte
Dim RecvBuff(0 To 262) As Byte
Dim MatchResult As Integer
Dim apdu As APDURec
Dim Transaction As SCARD_IO_REQUEST
Dim rName, scannerName As String
Dim fpTemplateLength, procTimeOut As Long
Dim rList As String * 256
Dim desFar, desFrr, achFar, achFrr As Long
Const MAX_BUFFER_LEN = 256
Const INVALID_SW1SW2 = -450
```

```
Private Sub cmdInit_Click()
  Dim indx As Long
  rList = String(255, vbNullChar)
  numRdr = 255

  'Inisialisasi SC Reader
  retCode = SCardEstablishContext(SCARD_SCOPE_USER, 0, 0, hContext)
  If retCode <> SCARD_S_SUCCESS Then
    Call displayOut(0, retCode, "")
    Exit Sub
  End If

  retCode = SCardListReaders(hContext, rName, rList, numRdr)
  If retCode <> SCARD_S_SUCCESS Then
    Call displayOut(0, retCode, "")
    Exit Sub
  End If
  Call LoadListToControl(cbReader, rList)
  cbReader.ListIndex = 0

  'Inisialisasi FP Scanner
  retCode = AET60_Open()
    If retCode <> 0 Then
      Call displayOut(1, retCode, "")
      Exit Sub
    End If
  numDev = AET60_GetNumDevices()
    If numDev > 0 Then
      cbScanner.Clear
      scannerName = String(255, vbNullChar)
      RetLength = Len(scannerName)
      For indx = 0 To (numDev - 1)
        retCode = AET60_GetDeviceName(indx, _
                                      scannerName, _
                                      RetLength)
        cbScanner.AddItem scannerName
        cbScanner.ListIndex = 0
      Next indx
    Else
      Call displayOut(1, retCode, "")
      Exit Sub
```

```
    End If

  Call displayOut(2, 0, "AET60 SC Reader telah terhubung.")
  Call displayOut(3, 0, "Klik perintah Connect.")
  Call enableTemplate

End Sub

Private Sub cmdConnect_Click()
  Call displayOut(2, 0, "")
  Call displayOut(3, 0, "")

  'Connect ke smartcard
  retCode = SCardConnect(hContext, _
                    cbReader.Text, _
                    SCARD_SHARE_SHARED, _
                    SCARD_PROTOCOL_T0 Or SCARD_PROTOCOL_T1, _
                    hCard, _
                    Protocol)
  If retCode <> SCARD_S_SUCCESS Then
    Call displayOut(0, retCode, "")
    Exit Sub
  End If
  scInit = True

  'Mengunci fp scanner dan men-set callback
  retCode = AET60_LockReader(cbScanner.ListIndex)
  If retCode <> 0 Then
    Call displayOut(1, retCode, "")
    Exit Sub
  End If
  rdrLocked = True

  retCode = AET60_SetCallback(AddressOf CALLBACK_FUNCTION)
  If retCode <> 0 Then
    Call displayOut(1, retCode, "")
    Exit Sub
  End If

  Call displayOut(3, 0, "Pilih perintah yang akan dieksekusi.")
  Call displayOut(2, 0, "Smart card telah terhubung.")
  cmdConnect.Enabled = False
  cmdEnroll.Enabled = True
  cmdVerify.Enabled = True
  dataSC.Enabled = False
  cmdInit.Enabled = False

End Sub
```

```
Private Sub cmdEnroll_Click()
  Dim tmpStr As String
  Dim ctr, indx, fileNo, byteNo As Long
  Dim tmpArray(0 To 31) As Byte

  Call clearMessages
  Call clearImage

  'Format kartu ACOS2
  retCode = SubmitIC()
  If retCode <> SCARD_S_SUCCESS Then
    Exit Sub
  End If
```

```
'Pilih FF 02
retCode = selectFile(&HFF, &H2)
If retCode <> SCARD_S_SUCCESS Then
  Exit Sub
End If
tmpStr = ""
For indx = 0 To 1
  tmpStr = tmpStr + Right("00" & Hex(RecvBuff(indx)), 2) + " "
Next indx
If tmpStr <> "90 00 " Then
  Call displayOut(2, 0, "The return string is invalid. Value: " + tmpStr)
  Exit Sub
End If

'Menulis record pertama dari FF 02
tmpArray(0) = &H0       ' 00   Option registers
tmpArray(1) = &H0       ' 00   Security option register
tmpArray(2) = &H3       ' 01   No of user files
tmpArray(4) = &H0       ' 00   Personalization bit
retCode = writeRecord(0, &H0, &H4, &H4, tmpArray())
If retCode <> SCARD_S_SUCCESS Then
  Exit Sub
End If

'Melakukan reset
retCode = SCardDisconnect(hCard, SCARD_UNPOWER_CARD)
If retCode <> SCARD_S_SUCCESS Then
  Call displayOut(0, retCode, "")
  Exit Sub
End If
scInit = False
retCode = SCardConnect(hContext, _
                cbReader.Text, _
                SCARD_SHARE_SHARED, _
                SCARD_PROTOCOL_T0 Or SCARD_PROTOCOL_T1, _
                hCard, _
                Protocol)
If retCode <> SCARD_S_SUCCESS Then
  Call displayOut(0, retCode, "")
  Exit Sub
End If
scInit = True

'Mengirim kode IC
retCode = SubmitIC()
If retCode <> SCARD_S_SUCCESS Then
  Exit Sub
End If

'Pilih FF 04
retCode = selectFile(&HFF, &H4)
If retCode <> SCARD_S_SUCCESS Then
  Exit Sub
End If
tmpStr = ""
For indx = 0 To 1
  tmpStr = tmpStr + Right("00" & Hex(RecvBuff(indx)), 2) + " "
Next indx
If tmpStr <> "90 00 " Then
  Call displayOut(2, 0, "The return string is invalid. Value: " + tmpStr)
  Exit Sub
```

```
End If

'Menulis record pertama dari FF 04
tmpArray(0) = &H20     ' 32    Panjang record
tmpArray(1) = &H18     ' 24    24 record
tmpArray(2) = &H0      ' 00    Atribut baca
tmpArray(3) = &H0      ' 00    Atribut Tulis
tmpArray(4) = &HAA     ' AA    Nama file
tmpArray(5) = &HAA     ' AA    Nama file
retCode = writeRecord(0, &H0, &H6, &H6, tmpArray())
If retCode <> SCARD_S_SUCCESS Then
  Exit Sub
End If

'Menulis record kedua dari FF 04
tmpArray(0) = &H7      ' 7     Panjang record
tmpArray(1) = &H1      ' 1     1 record
tmpArray(2) = &H0      ' 00    Atribut baca
tmpArray(3) = &H0      ' 00    Atribut Tulis
tmpArray(4) = &HBB     ' BB    Nama file
tmpArray(5) = &HBB     ' BB    Nama file
retCode = writeRecord(0, &H1, &H6, &H6, tmpArray())
If retCode <> SCARD_S_SUCCESS Then
  Exit Sub
End If

'Data tidak lengkap
If Not validTemplate Then
  Call displayOut(2, 0, "Data tidak lengkap.")
  Exit Sub
End If

'Mengambil data dari fp scanner
procTimeOut = 15000
fpTemplateLength = 768
retCode = AET60_Enroll(FpTemplate(0), _
                  fpTemplateLength, _
                  procTimeOut)
If retCode <> 0 Then
  Call displayOut(1, retCode, "")
  Exit Sub
End If
Call displayOut(2, 0, "Proses enroll sukses.")

'Menulis data ke kartu
'Pilih file AA AA
retCode = selectFile(&HAA, &HAA)
If retCode <> SCARD_S_SUCCESS Then
  Exit Sub
End If
tmpStr = ""
For indx = 0 To 1
  tmpStr = tmpStr + Right("00" & Hex(RecvBuff(indx)), 2) + " "
Next indx
If tmpStr <> "91 00 " Then
  Call displayOut(2, 0, "The return string is invalid. Value: " + tmpStr)
  Exit Sub
End If

'Menulis data fp scanner ke file AA AA
Call displayOut(3, 0, "Tunggu sebentar, data sedang disimpan pada kartu...")
userAction.Refresh
```

```
 ctr = 0
 For fileNo = 0 To 23          ' 24 record
   For byteNo = 0 To 31        ' 32 bytes per record
    tmpArray(byteNo) = FpTemplate(ctr)
    ctr = ctr + 1
   Next byteNo
   retCode = writeRecord(0, fileNo, &H20, &H20, tmpArray())
   If retCode <> SCARD_S_SUCCESS Then
     Exit Sub
   End If
 Next fileNo

 'Pilih file BB BB
 retCode = selectFile(&HBB, &HBB)
 If retCode <> SCARD_S_SUCCESS Then
  Exit Sub
 End If
 tmpStr = ""
 For indx = 0 To 1
  tmpStr = tmpStr + Right("00" & Hex(RecvBuff(indx)), 2) + " "
 Next indx
 If tmpStr <> "91 01 " Then
  Call displayOut(2, 0, "The return string is invalid. Value: " + tmpStr)
  Exit Sub
 End If

 'Menulis NRP ke File BB BB
 tmpStr = tbNRP.Text
 For indx = 0 To Len(tmpStr) - 1
  tmpArray(indx) = Asc(Mid(tmpStr, indx + 1, 1))
 Next indx
 retCode = writeRecord(1, &H0, &H7, Len(tmpStr), tmpArray())
 If retCode <> SCARD_S_SUCCESS Then
  Exit Sub
 End If

 'Memasukkan data sampel ke database
 Dim tmpJK As String
 If obJenisL.Value = True Then
  tmpJK = "L"
 ElseIf obJenisP.Value = True Then
  tmpJK = "P"
 End If
 con.Open
 rs.ActiveConnection = con
 cmd.ActiveConnection = con
 cmd.CommandText = "Insert into AplikasiAET60 values('" & Format(tbNRP.Text, "0000000") & "','" &
 tbNama.Text & "','" & tmpJK & "','" & tbAlamat.Text & "','" & tbLahir & "','" & cbAgama & "','" & cbFakultas
 &  "',",0)"
 cmd.Execute
 con.Close

 Call displayOut(3, 0, "")
 Call displayOut(2, 0, "Proses enroll sukses..!")

End Sub
```

```
Private Sub cmdVerify_Click()
 Dim tmpStr As String
 Dim ctr, indx, fileNo, byteNo As Long
 Dim tmpTemplate(0 To 767) As Byte
```

```
Call clearMessages
Call clearTemplate
Call clearImage

'Melakukan re-connect untuk menghilangkan error pada kasus kartu belum dimasukkan
retCode = SCardDisconnect(hCard, SCARD_UNPOWER_CARD)
scInit = False
retCode = SCardConnect(hContext, _
                cbReader.Text, _
                SCARD_SHARE_SHARED, _
                SCARD_PROTOCOL_T0 Or SCARD_PROTOCOL_T1, _
                hCard, _
                Protocol)
If retCode <> SCARD_S_SUCCESS Then
  Call displayOut(0, retCode, "")
  Call displayOut(3, 0, "Insert your ID card into the reader.")
  Exit Sub
End If
scInit = True

'Membaca data dari kartu
'Pilih file AA AA
retCode = selectFile(&HAA, &HAA)
If retCode <> SCARD_S_SUCCESS Then
  Exit Sub
End If
tmpStr = ""
For indx = 0 To 1
  tmpStr = tmpStr + Right("00" & Hex(RecvBuff(indx)), 2) + " "
Next indx
If tmpStr <> "91 00 " Then
  Call displayOut(2, 0, "The return string is invalid. Value: " + tmpStr)
  Exit Sub
End If

'Membaca data sidik jari dari kartu
Call displayOut(3, 0, "Please wait while fingerprint data is read from card.")
userAction.Refresh
ctr = 0
For fileNo = 0 To 23            ' Loop through record no
  retCode = readRecord(fileNo, &H20)
  If retCode <> SCARD_S_SUCCESS Then
    Exit Sub
  End If
  For byteNo = 0 To 31          ' Loop through 32 bytes per record
    tmpTemplate(ctr) = RecvBuff(byteNo)
    ctr = ctr + 1
  Next byteNo
Next fileNo

'Mengambil data sidik jari (real) dan mengatur nilai FAR dan FRR
Select Case secLevel.Value
  Case 0
    desFar = 4083440
    desFrr = 346819
  Case 1
    desFar = 2515777
    desFrr = 358630
  Case 2
    desFar = 500364
    desFrr = 464931
  Case 3
```

```
    desFar = 310419
    desFrr = 500364
  Case 4
    desFar = 2148
    desFrr = 654983
End Select

'Membaca data dari fp scanner
procTimeOut = 15000
fpTemplateLength = 768
MatchResult = 0
retCode = AET60_Verify(tmpTemplate(0), _
                       fpTemplateLength, _
                       procTimeOut, _
                       MatchResult, _
                       desFar, _
                       desFrr, _
                       achFar, _
                       achFrr)
If retCode <> 0 Then
  Call displayOut(1, retCode, "")
  Exit Sub
End If

'Membandingkan data sidik jari dari kartu dengan data sidik jari dari fp scanner
If MatchResult = 1 Then
  Call displayOut(2, 0, "Data sidik jari cocok.")
Else
  Call displayOut(2, 0, "Data sidik jari tidak cocok.")
  Exit Sub
End If

'Menampilkan informasi user yang terdapat pada kartu
'Pilih file BB BB
retCode = selectFile(&HBB, &HBB)
If retCode <> SCARD_S_SUCCESS Then
  Exit Sub
End If
tmpStr = ""
For indx = 0 To 1
  tmpStr = tmpStr + Right("00" & Hex(RecvBuff(indx)), 2) + " "
Next indx
If tmpStr <> "91 01 " Then
  Call displayOut(2, 0, "The return string is invalid. Value: " + tmpStr)
  Exit Sub
End If

'Membaca dan menampilkan NRP
retCode = readRecord(&H0, &H7)
If retCode <> SCARD_S_SUCCESS Then
  Exit Sub
End If
tmpStr = ""
indx = 0
While RecvBuff(indx) <> &H0
  tmpStr = tmpStr + Chr(RecvBuff(indx))
  indx = indx + 1
Wend
tbNRP.Text = tmpStr
```

```
 'Memasukkan data absensi ke db
  Dim query As String
  con.Open
  rs.ActiveConnection = con
  cmd.ActiveConnection = con
  While Len(tbNRP.Text) < 7
      tbNRP.Text = "0" & tbNRP.Text
   Wend
  query = "update AplikasiAET60 set Absensi='" & Format(DateTime.Now, "ddMMMyyyy hh:mm:ss") &
  "',num=num+1 where NRP='" & tbNRP.Text & "'"
  cmd.CommandText = query
  cmd.Execute
  con.Close

 'Menampilkan data lain yang ada pada db
  Dim tmpJK As String
  con.Open
  rs.ActiveConnection = con
  rs.Open "Select * from AplikasiAET60 where NRP='" & tbNRP.Text & "'"
  If rs.EOF = False Then
      tbNama.Text = rs!Nama
      tbAlamat.Text = rs!Alamat
      tbLahir.Text = rs!TglLahir
      cbAgama.Text = rs!Agama
      cbFakultas.Text = rs!Fakultas
      If rs!JenisKelamin = "L" Then
        obJenisL.Value = True
      Else
        obJenisP.Value = True
      End If

      MsgBox "Anda telah berhasil melakukan absensi pada " & rs!Absensi, vbOKOnly, "Information"
      rs.Close
      con.Close
  End If

End Sub
```

```
Private Sub cmdReset_Click()
  'Menon-aktifkan FP scanner
  If rdrLocked Then
    retCode = AET60_UnlockReader()
  End If
  retCode = AET60_Close()

  'Menon-aktifkan SC reader
  If scInit Then
    retCode = SCardDisconnect(hCard, SCARD_UNPOWER_CARD)
  End If
  retCode = SCardReleaseContext(hContext)
  scInit = False
  rdrLocked = False

  Call displayOut(2, 0, "")
  Call displayOut(3, 0, "")
  Call disableTemplate
  cbReader.Clear
  cbScanner.Clear
  cmdInit.Enabled = True
  Call disableTemplate
  Call clearTemplate
  Call clearImage
```

```
  cbReader.Enabled = True
  cbScanner.Enabled = True

End Sub
```

```
Private Sub Form_Load()
  con.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & App.Path &
  "\DataMahasiswa.mdb;Persist Security Info=False"
  cbAgama.Clear
  cbAgama.AddItem "Kristen"
  cbAgama.AddItem "Khatolik"
  cbAgama.AddItem "Islam"
  cbAgama.AddItem "Hindu"
  cbAgama.AddItem "Budha"
  cbFakultas.Clear
  cbFakultas.AddItem "Teknik"
  cbFakultas.AddItem "Kedokteran"
  cbFakultas.AddItem "Psikologi"
  cbFakultas.AddItem "Ekonomi"
  cbFakultas.AddItem "Sastra"
  cbAgama.ListIndex = 0
  cbFakultas.ListIndex = 0
  Call clearTemplate
  Call disableTemplate
  scInit = False
  rdrLocked = False
End Sub
```

```
Private Sub cmdQuit_Click()
  Unload Me
End Sub
```

```
Private Sub enableTemplate()
  dataSampel.Enabled = True
  cmdConnect.Enabled = True
End Sub
```

```
Private Sub disableTemplate()
  dataSampel.Enabled = False
  cmdConnect.Enabled = False
  cmdEnroll.Enabled = False
  cmdVerify.Enabled = False
End Sub
```

```
Private Sub clearTemplate()
  obJenisL.Value = False
  obJenisP.Value = False
  tbNRP.Text = ""
  tbNama.Text = ""
  tbAlamat.Text = ""
  tbLahir.Text = ""
  cbAgama.ListIndex = 0
  cbFakultas.ListIndex = 0
End Sub
```

```
Private Sub clearImage()
  Dim dib As BITMAPINFO_256
  Dim tmpArray(0 To 91259) As Byte
  Dim indx As Long
```

```
  For indx = 0 To 91259
    tmpArray(indx) = &H0
  Next indx

  dib.bmiHeader.biSize = Len(dib.bmiHeader)
  dib.bmiHeader.biWidth = 256
  dib.bmiHeader.biHeight = 360
  dib.bmiHeader.biPlanes = 1
  dib.bmiHeader.biBitCount = 8
  dib.bmiHeader.biCompression = BI_RGB
  dib.bmiHeader.biSizeImage = 0
  dib.bmiHeader.biXPelsPerMeter = 0
  dib.bmiHeader.biYPelsPerMeter = 0
  dib.bmiHeader.biClrUsed = 0
  dib.bmiHeader.biClrImportant = 0
  dib.bmiColors(0).rgbBlue = 0
  dib.bmiColors(0).rgbGreen = 64
  dib.bmiColors(0).rgbRed = 128
  dib.bmiColors(0).rgbReserved = 0

  Call StretchDIBits(AET60Main.fpImage.hdc, _
       0, 0, _
       AET60Main.fpImage.Width / 16, AET60Main.fpImage.Height / 16, _
       0, 0, _
       256, 360, _
       tmpArray(0), _
       dib, _
       DIB_RGB_COLORS, _
       SRCCOPY)
  AET60Main.fpImage.Refresh
End Sub
```

```
Private Sub clearMessages()
  Call displayOut(2, 0, "")
  Call displayOut(3, 0, "")
  userAction.Refresh
  MssgBox.Refresh
End Sub
```

```
Private Sub ClearBuffers()
  Dim indx As Long
  For indx = 0 To 262
    SendBuff(indx) = &H0
    RecvBuff(indx) = &H0
  Next indx
End Sub
```

```
Private Sub displayOut(ByVal errType As Integer, ByVal retVal As Long, ByVal msgText As String)
  Select Case (errType)
   Case 0  ' Smartcard error
      userAction.Text = "Error!! Tolong jalankan perintah Reset."
      MssgBox.Text = GetScardErrMsg(retVal)
    Case 1  ' BioAPI error
      Dim StatusMsg As String
      StatusMsg = String(255, vbNullChar)
      userAction.Text = "Error!!"
      Call AET60_GetStatus(retVal, StatusMsg)
      MssgBox.Text = StatusMsg
    Case 2  ' Pesan
      MssgBox.Text = msgText
    Case 3  ' Syarat yang dibutuhkan
      userAction.Text = msgText
```

```
  End Select
End Sub
```

```
Private Function validTemplate() As Boolean
  If tbNRP.Text = "" Then
    tbNama.SetFocus
    Call displayOut(3, 0, "Tolong isi NRP anda!")
    validTemplate = False
    Exit Function
  End If
  If tbNama.Text = "" Then
    tbNama.SetFocus
    Call displayOut(3, 0, "Tolong isi nama anda!")
    validTemplate = False
    Exit Function
  End If
  If objJenisL.Value = False And objJenisP.Value = False Then
    Call displayOut(3, 0, "Tolong isi jenis kelamin anda!")
    validTemplate = False
    Exit Function
  End If
  If tbAlamat.Text = "" Then
    tbAlamat.SetFocus
    Call displayOut(3, 0, "Tolong isi alamat anda!")
    validTemplate = False
    Exit Function
  End If
  If tbLahir.Text = "" Then
    tbLahir.SetFocus
    Call displayOut(3, 0, "Tolong isi tanggal lahir anda!")
    validTemplate = False
    Exit Function
  End If
  validTemplate = True
End Function
```

```
Private Function SubmitIC() As Long
  Dim indx As Integer
  Dim tmpStr As String

  Call ClearBuffers
  SendBuff(0) = &H80      ' CLA
  SendBuff(1) = &H20      ' INS
  SendBuff(2) = &H7       ' P1
  SendBuff(3) = &H0       ' P2
  SendBuff(4) = &H8       ' P3
  SendBuff(5) = &H41      ' A
  SendBuff(6) = &H43      ' C
  SendBuff(7) = &H4F      ' O
  SendBuff(8) = &H53      ' S
  SendBuff(9) = &H54      ' T
  SendBuff(10) = &H45     ' E
  SendBuff(11) = &H53     ' S
  SendBuff(12) = &H54     ' T
  SendLen = &HD
  RecvLen = &H2
  retCode = SendAPDU()
  If retCode <> SCARD_S_SUCCESS Then
    Call displayOut(0, retCode, "")
    SubmitIC = retCode
    Exit Function
  End If
```

```
  tmpStr = ""
  For indx = 0 To 1
    tmpStr = tmpStr + Right("00" & Hex(RecvBuff(indx)), 2) + " "
  Next indx
  If tmpStr <> "90 00 " Then
    Call displayOut(2, 0, "The return string is invalid. Value: " + tmpStr)
    retCode = INVALID_SW1SW2
  End If
  SubmitIC = retCode
End Function
```

```
Private Function selectFile(ByVal HiAddr As Byte, ByVal LoAddr As Byte) As Long
  Call ClearBuffers
  SendBuff(0) = &H80      ' CLA
  SendBuff(1) = &HA4      ' INS
  SendBuff(2) = &H0       ' P1
  SendBuff(3) = &H0       ' P2
  SendBuff(4) = &H2       ' P3
  SendBuff(5) = HiAddr    ' Value of High Byte
  SendBuff(6) = LoAddr    ' Value of Low Byte
  SendLen = &H7
  RecvLen = &H2
  retCode = SendAPDU()
  If retCode <> SCARD_S_SUCCESS Then
    Call displayOut(0, retCode, "")
  End If
  selectFile = retCode
End Function
```

```
Private Function readRecord(ByVal RecNo As Byte, ByVal DataLen As Byte) As Long
  Dim indx As Integer
  Dim tmpStr As String

  Call ClearBuffers
  SendBuff(0) = &H80        ' CLA
  SendBuff(1) = &HB2        ' INS
  SendBuff(2) = RecNo       ' P1    Record No
  SendBuff(3) = &H0         ' P2
  SendBuff(4) = DataLen     ' P3    Length of data to be read
  SendLen = &H5
  RecvLen = SendBuff(4) + 2
  retCode = SendAPDU()
  If retCode <> SCARD_S_SUCCESS Then
    Call displayOut(0, retCode, "")
    readRecord = retCode
    Exit Function
  End If
  tmpStr = ""
  For indx = 0 To 1
    tmpStr = tmpStr + Right("00" & Hex(RecvBuff(indx + SendBuff(4))), 2) + " "
  Next indx
  If tmpStr <> "90 00 " Then
    Call displayOut(2, 0, "The return string is invalid. Value: " + tmpStr)
    retCode = INVALID_SW1SW2
  End If
  readRecord = retCode
End Function
```

```
Private Function writeRecord(ByVal caseType As Integer, ByVal RecNo As Byte, _
                    ByVal maxDataLen As Byte, ByVal DataLen As Byte, _
                    ByRef DataIn() As Byte) As Long
  Dim tmpStr As String
```

```
 Dim indx As Integer

 If caseType = 1 Then
  ' 1. Re-initialize card values to $00
  Call ClearBuffers
  SendBuff(0) = &H80           ' CLA
  SendBuff(1) = &HD2           ' INS
  SendBuff(2) = RecNo          ' P1    Record to be written
  SendBuff(3) = &H0            ' P2
  SendBuff(4) = maxDataLen     ' P3    Length
  For indx = 0 To maxDataLen - 1
    SendBuff(indx + 5) = &H0
  Next indx
  SendLen = maxDataLen + 5
  RecvLen = &H2
  retCode = SendAPDU()
  If retCode <> SCARD_S_SUCCESS Then
    Call displayOut(0, retCode, "")
    writeRecord = retCode
    Exit Function
  End If
  tmpStr = ""
  For indx = 0 To 1
    tmpStr = tmpStr + Right("00" & Hex(RecvBuff(indx)), 2) + " "
  Next indx
  If tmpStr <> "90 00 " Then
    Call displayOut(2, 0, "The return string is invalid. Value: " + tmpStr)
    retCode = INVALID_SW1SW2
    writeRecord = retCode
    Exit Function
  End If
 End If

 ' 2. Write data to card
   Call ClearBuffers
  SendBuff(0) = &H80           ' CLA
  SendBuff(1) = &HD2           ' INS
  SendBuff(2) = RecNo          ' P1    Record to be written
  SendBuff(3) = &H0            ' P2
  SendBuff(4) = DataLen        ' P3    Length
  For indx = 0 To DataLen - 1
    SendBuff(indx + 5) = DataIn(indx)
  Next indx
  SendLen = DataLen + 5
  RecvLen = &H2
  retCode = SendAPDU()
  If retCode <> SCARD_S_SUCCESS Then
    Call displayOut(0, retCode, "")
    writeRecord = retCode
    Exit Function
  End If
  tmpStr = ""
  For indx = 0 To 1
    tmpStr = tmpStr + Right("00" & Hex(RecvBuff(indx)), 2) + " "
  Next indx
  If tmpStr <> "90 00 " Then
    Call displayOut(2, 0, "The return string is invalid. Value: " + tmpStr)
    retCode = INVALID_SW1SW2
  End If
  writeRecord = retCode
End Function
```

```
Private Function SendAPDU() As Long
  ioRequest.dwProtocol = dwActProtocol
  ioRequest.cbPciLength = Len(ioRequest)

  retCode = SCardTransmit(hCard, _
                    ioRequest, _
                    SendBuff(0), _
                    SendLen, _
                    ioRequest, _
                    RecvBuff(0), _
                    RecvLen)
 SendAPDU = retCode
End Function
```

```
Public Function GetScardErrMsg(ByVal ReturnCode As Long) As String
  Select Case ReturnCode
    'Smartcard Reader Errors
    Case SCARD_E_CANCELLED
    GetScardErrMsg = "The action was canceled by an SCardCancel request."
    Case SCARD_E_CANT_DISPOSE
    GetScardErrMsg = "The system could not dispose of the media in the requested manner."
    Case SCARD_E_CARD_UNSUPPORTED
    GetScardErrMsg = "The smart card does not meet minimal requirements for support."
    Case SCARD_E_DUPLICATE_READER
    GetScardErrMsg = "The reader driver didn't produce a unique reader name."
    Case SCARD_E_INSUFFICIENT_BUFFER
    GetScardErrMsg = "The data buffer for returned data is too small for the returned data."
    Case SCARD_E_INVALID_ATR
    GetScardErrMsg = "An ATR string obtained from the registry is not a valid ATR string."
    Case SCARD_E_INVALID_HANDLE
    GetScardErrMsg = "The supplied handle was invalid."
    Case SCARD_E_INVALID_PARAMETER
    GetScardErrMsg = "One or more of the supplied parameters could not be properly interpreted."
    Case SCARD_E_INVALID_TARGET
    GetScardErrMsg = "Registry startup information is missing or invalid."
    Case SCARD_E_INVALID_VALUE
    GetScardErrMsg = "One or more of the supplied parameter values could not be properly interpreted."
    Case SCARD_E_NOT_READY
    GetScardErrMsg = "The reader or card is not ready to accept commands."
    Case SCARD_E_NOT_TRANSACTED
    GetScardErrMsg = "An attempt was made to end a non-existent transaction."
    Case SCARD_E_NO_MEMORY
    GetScardErrMsg = "Not enough memory available to complete this command."
    Case SCARD_E_NO_SERVICE
    GetScardErrMsg = "The smart card resource manager is not running."
    Case SCARD_E_NO_SMARTCARD
    GetScardErrMsg = "The operation requires a smart card, but no smart card is currently in the device."
    Case SCARD_E_PCI_TOO_SMALL
    GetScardErrMsg = "The PCI receive buffer was too small."
    Case SCARD_E_PROTO_MISMATCH
    GetScardErrMsg = "The requested protocols are incompatible with the protocol currently in use with the card."
    Case SCARD_E_READER_UNAVAILABLE
    GetScardErrMsg = "The specified reader is not currently available for use."
    Case SCARD_E_READER_UNSUPPORTED
    GetScardErrMsg = "The reader driver does not meet minimal requirements for support."
    Case SCARD_E_SERVICE_STOPPED
    GetScardErrMsg = "The smart card resource manager has shut down."
    Case SCARD_E_SHARING_VIOLATION
    GetScardErrMsg = "The smart card cannot be accessed because of other outstanding connections."
    Case SCARD_E_SYSTEM_CANCELLED
    GetScardErrMsg = "The action was canceled by the system, presumably to log off or shut down."
    Case SCARD_E_TIMEOUT
```

```
    GetScardErrMsg = "The user-specified timeout value has expired."
    Case SCARD_E_UNKNOWN_CARD
    GetScardErrMsg = "The specified smart card name is not recognized."
    Case SCARD_E_UNKNOWN_READER
    GetScardErrMsg = "The specified reader name is not recognized."
    Case SCARD_F_COMM_ERROR
    GetScardErrMsg = "An internal communications error has been detected."
    Case SCARD_F_INTERNAL_ERROR
    GetScardErrMsg = "An internal consistency check failed."
    Case SCARD_F_UNKNOWN_ERROR
    GetScardErrMsg = "An internal error has been detected, but the source is unknown."
    Case SCARD_F_WAITED_TOO_LONG
    GetScardErrMsg = "An internal consistency timer has expired."
    Case SCARD_S_SUCCESS
    GetScardErrMsg = "No error was encountered."
    Case SCARD_W_REMOVED_CARD
    GetScardErrMsg = "The smart card has been removed, so that further communication is not possible."
    Case SCARD_W_RESET_CARD
    GetScardErrMsg = "The smart card has been reset, so any shared state information is invalid."
    Case SCARD_W_UNPOWERED_CARD
    GetScardErrMsg = "Power has been removed from the smart card, so that further communication is not
    possible."
    Case SCARD_W_UNRESPONSIVE_CARD
    GetScardErrMsg = "The smart card is not responding to a reset."
    Case SCARD_W_UNSUPPORTED_CARD
    GetScardErrMsg = "The reader cannot communicate with the card, due to ATR string configuration conflicts."
    Case Else
    GetScardErrMsg = "?"
    End Select
End Function
```