LAMPIRAN A
LISTING PROGRAM

A-1 Aplikasi Pembaca Kartu

A-1.1 Listing Program Utama

```
unit Gen;

interface
uses
  Windows, Messages, SysUtils,Variants, MyRoutines, Classes,
Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, ACR120U,ShellAPI,Registry;
type
  TForm1 = class(TForm)
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.dfm}
////////////////////////////////////////
/////BUAT REGISTER YG BELUM ADA///////////
////////////////////////////////////////
procedure DoAppToRun(RunName, AppName: string);
var
  Reg: TRegistry;
begin
  Reg := TRegistry.Create;
  with Reg do
  begin
    RootKey := HKEY_LOCAL_MACHINE;
    OpenKey('Software\Microsoft\Windows\CurrentVersion\Run',
True);
    WriteString(RunName, AppName);
    CloseKey;
    Free;
  end;
end;
////////////////////////////////////////
/////DELETE REGISTER YG UDAH ADA/////////
////////////////////////////////////////
procedure DelAppFromRun(RunName: string);
var
  Reg: TRegistry;
begin
  Reg := TRegistry.Create;
  with Reg do
  begin
    RootKey := HKEY_LOCAL_MACHINE;
    OpenKey('Software\Microsoft\Windows\CurrentVersion\Run',
True);
    if ValueExists(RunName) then DeleteValue(RunName);
    CloseKey;
    Free;
  end;
end;
procedure TForm1.FormCreate(Sender: TObject);
var
  //VARIABEL UNTUK PORT USB
  P_USB:integer;
  FirmVer:array[0..30] of Byte;
  infolen:Byte;
```

```
  Firm:String;
  temp:smallint;
  FirmVer1:array[0..19] of Byte;
  RState:tReaderStatus;
  //VARIABEL UNTUK SELECT KARTU
  HaveTag:array[0..49] of Byte;
  tmpArray:array[0..9] of Byte;
  tmpByte:Byte;
  Find:string;
  Find1:string;
  {SN:String;
  ctr:integer;
  ok:string;
  KeyType:byte;
  MSG:String;}
label
 ulang;

begin
////////////////////////////////////////////////////
////////////   BACA PORT USB ///////////////////////
////////////////////////////////////////////////////
    //====cek register ====
    if IsAppInRun('Genius') then
      begin
         DelAppFromRun('Genius');
      end;
      DoAppToRun('Genius', 'C:\Program Files\Genius
Technosoft\App\App.exe');
    //=======end cek register

    //===Find file driver
    Find := FileSearch('acr120u.dll', 'C:\WINDOWS\system32');
    Find1 := FileSearch('acr120.sys',
'C:\WINDOWS\system32\drivers');
    if (Find='') and (Find1='') then
      begin
        ShellExecute(0,'Open','SETUP.exe','','C:\Program
Files\Genius Technosoft\App',0);

        Application.Terminate;
        exit;
      end;
    //=====end find file driver
    //'=====inisialisasi port====
    P_USB:=0;
    //'open port connection to ACR120 Reader
    rHandle := ACR120_Open(P_USB);
    //'Check if Handle is Valid
    If Int(rHandle) < 0 Then
      begin
        MessageDlg('INVALID PORT USB', mtInformation, [mbOK], 0);
        Application.Terminate;
        exit;
      end
    Else
      begin
////////////////////////////////
////////////CARI DLL version /////
////////////////////////////////
        retcode := ACR120_RequestDLLVersion(@infolen, @FirmVer);
        Firm := '';
        for temp := 0 to infolen - 1 do
          begin
            Firm := Firm + Chr(FirmVer[temp]);
```

```
          end;
         ///END CARI DLL/////
         ////////////////////////////
        //Get the firmware version////
        ////////////////////////////
        retcode := ACR120_Status(rHandle, @FirmVer1, @RState);
        Firm := '';
        for temp := 0 to infolen - 1 do
        begin
          if (FirmVer1[temp] <> 0) And (FirmVer1[temp] <> 255)
Then
          begin
            Firm := Firm + chr(FirmVer1[temp]);
          end;
        end;
        //END BACA FIRMWARE/////
      end;//END PROSEDUR BACA PORT USB//
//////////////////////////////////////
/////SELECT KARTU ////////////////
//////////////////////////
        ulang:
{pilih card dan ambil ID cardnya}
        retcode:=
ACR120_Select(rHandle,@HaveTag,@tmpByte,@tmpArray);
        delay();
        delay();
//////////////////////////////
////BACA STATUS KARTU////////
//////////////////////////////
//Baca status
        Sec:=$06;
        BLCK:=2;
        dStatus:='';
        dStatus:=PilihSektor(Sec,BLCK);
//////////////
//Kalo Siswa//
//////////////
      if dStatus='siswa' then
        begin
//Baca NIS
        Sec:=$04;
        BLCK:=1;
        dNIS:='';
        dNIS:=PilihSektor(Sec,BLCK);
//Baca Password
        Sec:=$07;
        BLCK:=2;
        dPass:='';
        dPass:=PilihSektor(Sec,BLCK);
//Baca Angkatan
        Sec:=$06;
        BLCK:=1;
        dAng:='';
        dAng:=PilihSektor(Sec,BLCK);
//Buka shell
        Addr_URL:='http://NOC-PC-
012/Siswa/validSiswa.php?nis='+dNIS+'&password='+dPass+'&ang='+dA
ng;
        StrPCopy(A,Addr_URL);
        ShellExecute(0,'Open', A,'','',0);
        //MessageDlg((ErrDef(-3030)), mtInformation, [mbOK], 0);
        delay();
        goto ulang;
        end
//////////////
```

```
//Kalo Staff//
//////////////
      else if dStatus='staff' then
        begin
        ShellExecute(0,'Open','Log.exe','','C:\Program
Files\Genius Technosoft\Log',0);
        Application.Terminate;
        exit;
{//Baca NIK
        Sec:=$04;
        BLCK:=1;
        dNIK:='';
        dNIK:=PilihSektorKeyA(Sec,BLCK,LoginKey);
        //MessageDlg(dNIK, mtInformation, [mbOK], 0);
//Baca Password
        Sec:=$04;
        BLCK:=2;
        dPass:='';
        dPass:=PilihSektorKeyA(Sec,BLCK,LoginKey);
//Buka Shell
        Addr_URL:='http://NOC-PC-
012/Admin/validAdmin.php?nik='+dNIK+'&password='+dPass+'&staff='+
dStatus;
        StrPCopy(A,Addr_URL);
        ShellExecute(0,'Open', A,'','',0);
        delay();
        //MessageDlg(dStatus, mtInformation, [mbOK], 0);
        goto ulang;}
        end
//kalo salah or ga ada, balik ke atas
      else
        begin
          //MessageDlg((ErrDef(-3030)), mtInformation, [mbOK],
0);
          goto ulang;
        end;
////END STATUS KARTU
////END SELECT KARTU/////////
end;//END PROCEDUR FORM CREATE
end.
```

A-2 Aplikasi Pembayaran

A-2.1 Instruksi Form Utama

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls,ACR120U,ShellAPI,MyRoutines,Unit2;

type
  TForm1 = class(TForm)
    Panel1: TPanel;
    GroupBox1: TGroupBox;
    SignIn: TButton;
    GroupBox2: TGroupBox;
    Pulsa: TButton;
    Label1: TLabel;
    Button1: TButton;
    procedure FormCreate(Sender: TObject);
```

```
    procedure SignInClick(Sender: TObject);
    procedure PulsaClick(Sender: TObject);
    procedure Close(Sender: TObject; var Action: TCloseAction);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.FormCreate(Sender: TObject);
var
  //VARIABEL UNTUK PORT USB
  P_USB:integer;
  FirmVer:array[0..30] of Byte;
  Firm:String;
  infolen:Byte;
  temp:smallint;
  FirmVer1:array[0..19] of Byte;
  RState:tReaderStatus;

begin
    P_USB:=0;
    //'open port connection to ACR120 Reader
    rHandle := ACR120_Open(P_USB);
    //'Check if Handle is Valid
    If Int(rHandle) < 0 Then
      begin
        MessageDlg('INVALID PORT USB', mtWarning, [mbOK], 0);
        Application.Terminate;
        exit;
      end
    Else
      begin
////////////////////////////////
////////////CARI DLL version /////
////////////////////////////////
        retcode := ACR120_RequestDLLVersion(@infolen, @FirmVer);
        Firm := '';
        for temp := 0 to infolen - 1 do
          begin
            Firm := Firm + Chr(FirmVer[temp]);
          end;
         ///END CARI DLL/////


         ////////////////////////////
        //Get the firmware version////
        ////////////////////////////
        retcode := ACR120_Status(rHandle, @FirmVer1, @RState);
        Firm := '';
        for temp := 0 to infolen - 1 do
        begin
          if (FirmVer1[temp] <> 0) And (FirmVer1[temp] <> 255)
Then
          begin
            Firm := Firm + chr(FirmVer1[temp]);
          end;
```

```
        end;
        //END BACA FIRMWARE/////
      end;//END PROSEDUR BACA PORT USB//
end;

procedure TForm1.SignInClick(Sender: TObject);
var
  //VARIABEL UNTUK SELECT KARTU
  HaveTag:array[0..49] of Byte;
  tmpArray:array[0..9] of Byte;
  tmpByte:Byte;
begin
  retcode:= ACR120_Select(rHandle,@HaveTag,@tmpByte,@tmpArray);
  if retcode<0 then
  begin
    MessageDlg('Invalid Card', mtWarning, [mbOK], 0);
    exit;
  end
  else
  begin
//////////////////////////////
////BACA STATUS KARTU////////
//////////////////////////////
//Baca status
        Sec:=$06;
        BLCK:=2;
        dStatus:='';
        dStatus:=PilihSektor(Sec,BLCK,rHandle);
//Baca NIK
        LoginKey:=ACR120_LOGIN_KEYTYPE_A;
        Sec:=$04;
        BLCK:=1;
        dNIK:='';
        dNIK:=PilihSektorKeyA(Sec,BLCK,LoginKey,rHandle);
        //MessageDlg(dNIK, mtInformation, [mbOK], 0);
//Baca Password
        Sec:=$04;
        BLCK:=2;
        dPass:='';
        dPass:=PilihSektorKeyA(Sec,BLCK,LoginKey,rHandle);
//Buka Shell
        Addr_URL:='http://NOC-PC-
012/Admin/validAdmin.php?nik='+dNIK+'&password='+dPass+'&staff='+
dStatus;
        StrPCopy(A,Addr_URL);
        ShellExecute(0,'Open', A,'','',0);
end;//dari else
end;//baca status kartu
procedure TForm1.PulsaClick(Sender: TObject);
begin
  FormPulsa.ShowModal;
end;

procedure TForm1.Close(Sender: TObject; var Action:
TCloseAction);
begin
  ACR120_Close(rHandle);
  ShellExecute(0,'Open','App.exe','','C:\Program Files\Genius
Technosoft\App',0);
  Application.Terminate;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
ACR120_Close(rHandle);
```

```
  ShellExecute(0,'Open','App.exe','','C:\Program Files\Genius
Technosoft\App',0);
  Application.Terminate;
end;

end.
```

A-2.2 Instruksi Form Pembayaran

```
unit Unit2;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls, Forms,
  Dialogs, StdCtrls, CheckLst,
ExtCtrls,MyRoutines,ACR120U,ShellAPI;

type
  TFormPulsa = class(TForm)
    Panel1: TPanel;
    GroupBox1: TGroupBox;
    ButtCek: TButton;
    EditCek: TEdit;
    Label1: TLabel;
    GroupBox2: TGroupBox;
    Label2: TLabel;
    ButtTambah: TButton;
    EditPulsa: TEdit;
    ListBulan: TCheckListBox;
    GroupBox3: TGroupBox;
    EditIuran: TEdit;
    Label3: TLabel;
    Label4: TLabel;
    SubmitBut: TButton;
    Label5: TLabel;
    Label6: TLabel;
    Button4: TButton;
    Edit1: TEdit;
    procedure Button4Click(Sender: TObject);
    procedure ButtCekClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure ButtTambahClick(Sender: TObject);
    procedure SubmitButClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FormPulsa: TFormPulsa;

implementation

{$R *.dfm}

procedure TFormPulsa.Button4Click(Sender: TObject);
begin
  FormPulsa.Close;
end;

procedure TFormPulsa.ButtCekClick(Sender: TObject);
var
  //VARIABEL UNTUK SELECT KARTU
```

```
  HaveTag:array[0..49] of Byte;
  tmpArray:array[0..9] of Byte;
  tmpByte:Byte;
begin
  retcode:= ACR120_Select(rHandle,@HaveTag,@tmpByte,@tmpArray);
  if retcode<0 then
  begin
    MessageDlg('Invalid Card', mtWarning, [mbOK], 0);
    Application.Terminate;
  end
  else
  begin
//////////////////////////////
////BACA STATUS KARTU////////
//////////////////////////////
//Baca status
        Sec:=$07;
        BLCK:=1;
        dPulsa:='';
        dPulsa:=PilihSektor(Sec,BLCK,rHandle);
        EditCek.Text:=dPulsa;
//end baca pulsa
  end;//end else
end;

procedure TFormPulsa.FormCreate(Sender: TObject);
var
  //VARIABEL UNTUK PORT USB
  P_USB:integer;
  FirmVer:array[0..30] of Byte;
  Firm:String;
  infolen:Byte;
  temp:smallint;
  FirmVer1:array[0..19] of Byte;
  RState:tReaderStatus;
begin
    P_USB:=0;
    //'open port connection to ACR120 Reader
    rHandle := ACR120_Open(P_USB);
    //'Check if Handle is Valid
    If Int(rHandle) < 0 Then
      begin
        MessageDlg('INVALID PORT USB', mtInformation, [mbOK], 0);
        Application.Terminate;
        exit;
      end
    Else
      begin
////////////////////////////////
////////////CARI DLL version /////
////////////////////////////////
        retcode := ACR120_RequestDLLVersion(@infolen, @FirmVer);
        Firm := '';

        for temp := 0 to infolen - 1 do
          begin
            Firm := Firm + Chr(FirmVer[temp]);
          end;
         ///END CARI DLL/////
         ////////////////////////////
        //Get the firmware version////
        ////////////////////////////
        retcode := ACR120_Status(rHandle, @FirmVer1, @RState);
        Firm := '';
        for temp := 0 to infolen - 1 do
```

```
        begin
          if (FirmVer1[temp] <> 0) And (FirmVer1[temp] <> 255)
Then
          begin
            Firm := Firm + chr(FirmVer1[temp]);
          end;
        end;
        //END BACA FIRMWARE/////
      end;//END PROSEDUR BACA PORT USB//
end;

procedure TFormPulsa.ButtTambahClick(Sender: TObject);
var
  //VARIABEL UNTUK SELECT KARTU
  HaveTag:array[0..49] of Byte;
  tmpArray:array[0..9] of Byte;
  tmpByte:Byte;

  //variabel lokalnya
  totInt,x,y:integer;
  tambah:String;
  retkode: smallint;
begin
 retcode:= ACR120_Select(rHandle,@HaveTag,@tmpByte,@tmpArray);
  if retcode<0 then
  begin
    MessageDlg('Invalid Card', mtWarning, [mbOK], 0);
    Application.Terminate;
  end
  else
  begin

//////////////////////////////
////BACA STATUS KARTU////////
//////////////////////////////
//Baca status
        Sec:=$07;
        BLCK:=1;
        dPulsa:='';
        dPulsa:=PilihSektor(Sec,BLCK,rHandle);
        tambah:='';
        tambah:=EditPulsa.Text;
        if tambah='' then
          exit;

        totInt:=StrToInt(dPulsa)+ StrToInt(tambah);
        tambah:=IntToStr(totInt);

        if retcode<0 then
        begin
          MessageDlg(ErrDef(retcode), mtWarning, [mbOK], 0);
          exit;
          Application.Terminate;
        end;
        //convert ke byte
        Edit1.Text:=tambah;
        x:=length(Edit1.Text);
        for y:=0 to x do
          begin
            dout[y]:=ord(Edit1.Text[y+1]);
          end;
        BLCK := Sec* 4 + BLCK;
        retkode:=ACR120_Write(rHandle,BLCK,@dout);
        if retkode<0 then
        begin
```

```
          MessageDlg(ErrDef(retkode), mtWarning, [mbOK], 0);
        end;
//==========kirim ke web=============
//Baca NIS
        Sec:=$04;
        BLCK:=1;
        dNIS:='';
        dNIS:=PilihSektor(Sec,BLCK,rHandle);
//Baca Angkatan
        Sec:=$06;
        BLCK:=1;
        dAng:='';
        dAng:=PilihSektor(Sec,BLCK,rHandle);
//Buka Shell
        Addr_URL:='http://NOC-PC-
012/Administrasi/validPulsa.php?pls='+tambah+'&ang='+dAng+'&nis='
+dNIS;
        StrPCopy(A,Addr_URL);
        ShellExecute(0,'Open', A,'','',0);
        exit;

//end baca pulsa
  end;//end else
end;

procedure TFormPulsa.SubmitButClick(Sender: TObject);
var
   //VARIABEL UNTUK SELECT KARTU
  HaveTag:array[0..49] of Byte;
  tmpArray:array[0..9] of Byte;
  tmpByte:Byte;

  BiaIuran,i,imax,jmlbln:integer;
  bln:array [0..11] of string;
  totbiaya,pulsa:integer;
begin
  if EditIuran.Text='' then
    exit;
  BiaIuran:=StrToInt(EditIuran.Text);
  jmlbln:=0;
  for i:=0 to 11 do
  if ListBulan.Checked[i]=true then
    begin
    jmlbln:=jmlbln+1;
    bln[i]:=ListBulan.Items[i];
    end;
  totbiaya:=BiaIuran*jmlbln;
//=========select card dulu=========
  retcode:= ACR120_Select(rHandle,@HaveTag,@tmpByte,@tmpArray);
  if retcode<0 then
  begin
    MessageDlg('Invalid Card', mtWarning, [mbOK], 0);
    Application.Terminate;
  end
  else
  begin
////////////////////////////
////BACA STATUS KARTU////////
////////////////////////////
//Baca status
        Sec:=$07;
        BLCK:=1;
        dPulsa:='';
        dPulsa:=PilihSektor(Sec,BLCK,rHandle);
        pulsa:=StrToInt(dPulsa);
```

```
        if totbiaya>pulsa then
        begin
         MessageDlg('Pulsa tidak cukup untuk melakukan
Transaksi', mtWarning, [mbOK], 0);
          exit;
        end;
        pulsa:=pulsa-totbiaya;
        dPulsa:=IntToStr(pulsa);
        Edit1.Text:=IntToStr(pulsa);
        imax:=length(Edit1.Text);
        for i:=0 to imax do
        begin
          dout[i]:=ord(Edit1.Text[i+1]);
        end;
        BLCK := Sec* 4 + BLCK;
        retcode:=ACR120_Write(rHandle,BLCK,@dout);
        if retcode<0 then
        begin
          MessageDlg('Transaski gagal', mtWarning, [mbOK], 0);
          exit;
        end;
//Baca NIS
        Sec:=$04;
        BLCK:=1;
        dNIS:='';
        dNIS:=PilihSektor(Sec,BLCK,rHandle);
//Baca Angkatan
        Sec:=$06;
        BLCK:=1;
        dAng:='';
        dAng:=PilihSektor(Sec,BLCK,rHandle);
//Buka Shell
        Addr_URL:='http://NOC-PC-
012/Administrasi/validIuran.php?ja='+bln[0]+'&fb='+bln[1]+'&mr='+
bln[2]+'&ap='+bln[3]+'&me='+bln[4]+'&jn='+bln[5]+'&jl='+bln[6]+'&
ag='+bln[7]+'&sp='+bln[8]+'&ok='+bln[9]+'&nv='+bln[10]+'&ds='+bln
[11]+'&pls='+dPulsa+'&ang='+dAng+'&nis='+dNIS;
        StrPCopy(A,Addr_URL);
        ShellExecute(0,'Open', A,'','',0);
        exit;
  end; //end else
end;

end.
```

A-3 Aplikasi Penulisan Data Smart Card

A-3.1 Listing Program Form Utama

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls, Forms,
  Dialogs,ACR120U,MyRoutines, StdCtrls, ExtCtrls,Unit2,Unit3;

type
  TForm1 = class(TForm)
    Panel1: TPanel;
    GroupBox1: TGroupBox;
    KrtSiswa: TButton;
    KrtStaff: TButton;
    Label1: TLabel;
```

```
    Label2: TLabel;
    ButtonExit: TButton;
    procedure KrtSiswaClick(Sender: TObject);
    procedure KrtStaffClick(Sender: TObject);
    procedure ButtonExitClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.KrtSiswaClick(Sender: TObject);
begin
FormSiswa.ShowModal;
end;

procedure TForm1.KrtStaffClick(Sender: TObject);
begin
FormStaff.ShowModal;
end;

procedure TForm1.ButtonExitClick(Sender: TObject);
begin
Application.Terminate;
end;

end.
```

A-3.2 Listing Program Form Penulisan Data Siswa

```
unit Unit2;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls, Forms,
  Dialogs,MyRoutines,ACR120U, StdCtrls,ShellAPI, ExtCtrls;

type
  TFormSiswa = class(TForm)
    GroupBox1: TGroupBox;
    Label1: TLabel;
    EditNama: TEdit;
    EditTempat: TEdit;
    EditLahir: TEdit;
    EditNIS: TEdit;
    EditKls: TEdit;
    EditAng: TEdit;
    EditStatus: TEdit;
    EditPass: TEdit;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    GroupBox2: TGroupBox;
    Label6: TLabel;
    Label7: TLabel;
```

```
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    GroupBox3: TGroupBox;
    ButtonSub: TButton;
    ButtonCancel: TButton;
    ButtonRst: TButton;
    Panel1: TPanel;
    EditAlamat: TEdit;
    procedure ButtonRstClick(Sender: TObject);
    procedure ButtonSubClick(Sender: TObject);
    procedure ButtonCancelClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FormSiswa: TFormSiswa;

implementation

{$R *.dfm}

procedure TFormSiswa.ButtonRstClick(Sender: TObject);
begin
EditNama.Text:='';
EditLahir.Text:='';
EditAlamat.Text:='';
EditTempat.Text:='';
EditAng.Text:='2008';
EditKls.Text:='1a';
EditNIS.Text:='0800';
EditPass.Text:='';
EditStatus.Text:='siswa';
end;

procedure TFormSiswa.ButtonSubClick(Sender: TObject);
var
Nama,Tempat,Lahir,Alamat,Kelas,Status,Pass,NIS,Ang:String;

//VARIABEL UNTUK SELECT KARTU
  HaveTag:array[0..49] of Byte;
  tmpArray:array[0..9] of Byte;
  tmpByte:Byte;

//variabel lokal
cek:boolean;

begin
//ambil data pada field2-nya
Nama:=EditNama.Text;
Tempat:=EditTempat.Text;
Lahir:=EditLahir.Text;
Alamat:=EditAlamat.Text;
Kelas:=EditKls.Text;
Status:=EditStatus.Text;
Pass:=EditPass.Text;
NIS:=EditNIS.Text;
Ang:=EditAng.Text;
if (Nama='') or (Tempat='') or (Lahir='') or (Alamat='') or
(Kelas='') or (Status='') or (Pass='') or (NIS='') or (Ang='')
then
```

```
begin
MessageDlg('Fill the Box', mtWarning, [mbOK], 0);
exit;
end;
//end ambil data pada field

//select kartu
 retcode:= ACR120_Select(rHandle,@HaveTag,@tmpByte,@tmpArray);
  if retcode<0 then
  begin
    MessageDlg('Invalid Card', mtWarning, [mbOK], 0);
    FormSiswa.Close;
    ACR120_Close(rHandle);
    Exit;
  end;
//end select kartu

//tulis nama ke kartu
Sec:=$00;
BLCK:=1;
cek:=TulisSektorBlok(Sec,BLCK,rHandle,Nama);
//end tulis nama

//tulis tanggal lahir dan tempat ke kartu
Sec:=$01;
BLCK:=1; //blok lahir
cek:=TulisSektorBLok(Sec,BLCK,rHandle,Lahir);
BLCK:=2; //blok tempat
cek:=TulisSektorBlok(Sec,BLCK,rHandle,Tempat);
//end tulis tempat

//tulis alamat
Sec:=$02;
BLCK:=1;
cek:=TulisSektorBlok(Sec,BLCK,rHandle,Alamat);
//end tulis alamat

//tulis NIS,kelas,angkatan,status,password
Sec:=$04;
BLCK:=1;
cek:=TulisSektorBlok(Sec,BLCK,rHandle,NIS);
BLCK:=2;
cek:=TulisSektorBlok(Sec,BLCK,rHandle,Kelas);
Sec:=$06;
BLCK:=1;
cek:=TulisSektorBlok(Sec,BLCK,rHandle,Ang);
BLCK:=2;
cek:=TulisSektorBlok(Sec,BLCK,rHandle,Status);
Sec:=$07;
BLCK:=2;
cek:=TulisSektorBlok(Sec,BLCK,rHandle,Pass);
//end tulis nis dll
//kirim ke server
Addr_URL:='http://NOC-PC-
012/Daftar/addSiswa.php?nam='+Nama+'&nis='+NIS+'&ang='+Ang+'&kls=
'+Kelas+'&pass='+Pass;
StrPCopy(A,Addr_URL);
ShellExecute(0,'Open', A,'','',0);
delay();
EditNama.Text:='';
EditLahir.Text:='';
EditAlamat.Text:='';
EditTempat.Text:='';
EditAng.Text:='2008';
EditKls.Text:='1a';
```

```
EditNIS.Text:='0800';
EditPass.Text:='';
EditStatus.Text:='siswa';
//end kirim ke server
end;


procedure TFormSiswa.ButtonCancelClick(Sender: TObject);
begin
ACR120_Close(rHandle);
FormSiswa.Close;
end;

procedure TFormSiswa.FormCreate(Sender: TObject);
var
//VARIABEL UNTUK PORT USB
  P_USB:integer;
  FirmVer:array[0..30] of Byte;
  Firm:String;
  infolen:Byte;
  temp:smallint;
  FirmVer1:array[0..19] of Byte;
  RState:tReaderStatus;
begin
 P_USB:=0;
    //'open port connection to ACR120 Reader
    rHandle := ACR120_Open(P_USB);
    //'Check if Handle is Valid
    If Int(rHandle) < 0 Then
      begin
        MessageDlg('INVALID PORT USB', mtWarning, [mbOK], 0);
        Application.Terminate;
        exit;
      end
    Else
      begin
////////////////////////////////
////////////CARI DLL version /////
////////////////////////////////
        retcode := ACR120_RequestDLLVersion(@infolen, @FirmVer);
        Firm := '';
        for temp := 0 to infolen - 1 do
          begin
            Firm := Firm + Chr(FirmVer[temp]);
          end;
         ///END CARI DLL/////

         ////////////////////////////
        //Get the firmware version////
        ////////////////////////////
        retcode := ACR120_Status(rHandle, @FirmVer1, @RState);

        Firm := '';
        for temp := 0 to infolen - 1 do
        begin
          if (FirmVer1[temp] <> 0) And (FirmVer1[temp] <> 255)
Then
          begin
            Firm := Firm + chr(FirmVer1[temp]);
          end;
        end;

        //END BACA FIRMWARE/////

      end;//END PROSEDUR BACA PORT USB//
```

```
      //MessageDlg('USB', mtWarning, [mbOK], 0);
end;


end.
```

A-3.3 Listing Program Form Penulisan Data Staff

```
unit Unit3;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls,MyRoutines,ACR120U,ShellAPI;

type
  TFormStaff = class(TForm)
    GroupBox1: TGroupBox;
    Edit_Nama: TEdit;
    Edit_Tempat: TEdit;
    Edit_Lahir: TEdit;
    Edit_Alamat: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    GroupBox2: TGroupBox;
    Edit_NIK: TEdit;
    Edit_Status: TEdit;
    Edit_Password: TEdit;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    ComboGolongan: TComboBox;
    GroupBox3: TGroupBox;
    Panel1: TPanel;
    Button_Sub: TButton;
    Button_Cancel: TButton;
    Button_Rst: TButton;
    procedure Button_CancelClick(Sender: TObject);
    procedure Button_SubClick(Sender: TObject);
    procedure Button_RstClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FormStaff: TFormStaff;

implementation

{$R *.dfm}

procedure TFormStaff.Button_CancelClick(Sender: TObject);
begin
ACR120_Close(rHandle);
FormStaff.Close;
end;
```

```
procedure TFormStaff.Button_SubClick(Sender: TObject);
var
Nama1,Tempat1,Lahir1,Alamat1,NIK1,Golongan1,Status1,Password1:str
ing;

//VARIABEL UNTUK SELECT KARTU
  HaveTag:array[0..49] of Byte;
  tmpArray:array[0..9] of Byte;
  tmpByte:Byte;

retkode:smallint;
cek:boolean;
LoginKey:Byte;
begin
//ambil data dari fieldnya
Nama1:=Edit_Nama.Text;
Tempat1:=Edit_Tempat.Text;
Lahir1:=Edit_Lahir.Text;
Alamat1:=Edit_Alamat.Text;
NIK1:=Edit_NIK.Text;
Status1:=Edit_Status.Text;
Password1:=Edit_Password.Text;
Golongan1:=ComboGolongan.Text;

if (Nama1='') or (Tempat1='') or (Lahir1='') or (Alamat1='') or
(NIK1='') or (Status1='') or (Password1='') or (Golongan1='')then
begin
MessageDlg('Fill the Box', mtWarning, [mbOK], 0);
exit;
end;

//end ambil data dari field

//select kartu
 retcode:= ACR120_Select(rHandle,@HaveTag,@tmpByte,@tmpArray);
  if retcode<0 then
  begin
    MessageDlg('Invalid Card', mtWarning, [mbOK], 0);
    FormStaff.Close;
    ACR120_Close(rHandle);
    Exit;
  end;
//end select kartu

//tulis nama ke kartu
Sec:=$00;
BLCK:=1;
cek:=TulisSektorBlok(Sec,BLCK,rHandle,Nama1);
//end tulis nama

//tulis tanggal lahir dan tempat ke kartu
Sec:=$01;
BLCK:=1; //blok lahir
cek:=TulisSektorBLok(Sec,BLCK,rHandle,Lahir1);
BLCK:=2; //blok tempat
cek:=TulisSektorBlok(Sec,BLCK,rHandle,Tempat1);
//end tulis tempat

//tulis alamat
Sec:=$02;
BLCK:=1;
cek:=TulisSektorBlok(Sec,BLCK,rHandle,Alamat1);
//end tulis alamat
```

```
//tulis golongan dan status
Sec:=$06;
BLCK:=1;
cek:=TulisSektorBlok(Sec,BLCK,rHandle,Golongan1);
BLCK:=2;
cek:=TulisSektorBlok(Sec,BLCK,rHandle,Status1);
//end tulis golongan

//tulis nik dan password
LoginKey:=ACR120_LOGIN_KEYTYPE_A;
Sec:=$04;
BLCK:=1;
cek:=TulisSektorKeyA(Sec,BLCK,rHandle,NIK1);
BLCK:=2;
cek:=TulisSektorKeyA(Sec,BLCK,rHandle,Password1);
//end tulis nik dan password

//kirim ke server
Addr_URL:='http://NOC-PC-
012/Daftar/addStaff.php?nam='+Nama1+'&nik='+NIK1+'&pass='+Passwor
d1;
StrPCopy(A,Addr_URL);
ShellExecute(0,'Open', A,'','',0);
delay();
Edit_Nama.Text:='';
Edit_Lahir.Text:='';
Edit_Alamat.Text:='';
Edit_Tempat.Text:='';
Edit_NIK.Text:='';
Edit_Password.Text:='';
//end kirim ke server
end;

procedure TFormStaff.Button_RstClick(Sender: TObject);
begin
Edit_Nama.Text:='';
Edit_Tempat.Text:='';
Edit_Lahir.Text:='';
Edit_Alamat.Text:='';
Edit_NIK.Text:='';
Edit_Status.Text:='staff';
Edit_Password.Text:='';
ComboGolongan.Text:='Golongan';
end;

procedure TFormStaff.FormCreate(Sender: TObject);
var
//VARIABEL UNTUK PORT USB
  P_USB:integer;
  FirmVer:array[0..30] of Byte;
  Firm:String;
  infolen:Byte;
  temp:smallint;
  FirmVer1:array[0..19] of Byte;
  RState:tReaderStatus;
begin
 P_USB:=0;
    //'open port connection to ACR120 Reader
    rHandle := ACR120_Open(P_USB);
    //'Check if Handle is Valid
    If Int(rHandle) < 0 Then
      begin
        MessageDlg('INVALID PORT USB', mtWarning, [mbOK], 0);
        Application.Terminate;
        exit;
```

```
      end
    Else
      begin
////////////////////////////////
/////////////CARI DLL version /////
////////////////////////////////
        retcode := ACR120_RequestDLLVersion(@infolen, @FirmVer);

        Firm := '';

        for temp := 0 to infolen - 1 do
          begin
            Firm := Firm + Chr(FirmVer[temp]);
          end;
         ///END CARI DLL/////

         ////////////////////////////
        //Get the firmware version////
        ////////////////////////////
        retcode := ACR120_Status(rHandle, @FirmVer1, @RState);
        Firm := '';
        for temp := 0 to infolen - 1 do
        begin
          if (FirmVer1[temp] <> 0) And (FirmVer1[temp] <> 255)
Then
          begin
            Firm := Firm + chr(FirmVer1[temp]);
          end;
        end;
        //END BACA FIRMWARE/////
      end;//END PROSEDUR BACA PORT USB//
      //MessageDlg('USB', mtWarning, [mbOK], 0);
end;

end.
```

A-4 Listing Program Routine

A-4.1 MyRoutines

```
unit MyRoutines;

interface

uses Windows, Messages, SysUtils, Classes, Graphics, Controls,
Forms, Dialogs, ACR120U,
     ExtCtrls, StdCtrls,Registry;

var
//sto: Longint;
LogType: Byte;
buttonOK: Boolean;
rHandle: smallint;
retcode: smallint;
//SID: Byte;
Sec:smallint;
dataRead: array[0..15] of Byte;
dPass:String;
dAng:String;
dNIS: String;
dNIK: String;
dStatus:String;
dPulsa:String;
dout:array[0..15] of Byte;
BLCK: Byte;
A:Array [0..300] of Char;
```

```
Addr_URL:String;
Button_Ok:Boolean;
FisikSector:Byte;
Sector:Byte;
ExitKode:Boolean;
LoginKey:Byte;
{ReadAsc: boolean;
WriteAsc: boolean;
PhysicalSector: smallint;}


function ErrDef(ErrorCode:smallint):string;
Function StrHEX_Dec(StrHex: char) : Integer;
Function Hex_Dec(val: TEdit): Byte;
function GetTagType1(GetTagTypeA: Byte):string;
function PilihSektor(SectorFisik:Byte;Blok:Byte;handle:smallint):
String;
Function
TulisSektorBlok(SectorFisik:Byte;Blok:Byte;handle:smallint;data:S
tring):boolean;
function
PilihSektorKeyA(SectorFisik:Byte;Blok:Byte;Login:Byte;handle:smal
lint): String;
Function
TulisSektorKeyA(SectorFisik:Byte;Blok:Byte;handle:smallint;data:s
tring):boolean;
function IsAppInRun(RunName: string): Boolean;
function delay():string;



implementation



//'=====================================================
//'  Routine for Defining Error Code
//'=====================================================

function ErrDef(ErrorCode:smallint):string;

begin

     case ErrorCode of
          -1000: ErrDef := '( X ) Unexpected Internal Library
Error : -1000';
          -2000: ErrDef := '( X ) Invalid Port : -2000';
          -2010: ErrDef := '( X ) Port Occupied by Another
Application : -2010';
          -2020: ErrDef := '( X ) Invalid Handle : -2020';
          -2030: ErrDef := '( X ) Incorrect Parameter : -2030';
          -3000: ErrDef := '( X ) No TAG Selected or in Reachable
Range : -3000';
          -3010: ErrDef := '( X ) Read Failed after Operation : -
3010';
          -3020: ErrDef := '( X ) Block does not contain value :
-3020';
          -3030: ErrDef := '( X ) Operation Failed : -3030';
          -3040: ErrDef := '( X ) Unknown Reader Error : -3040';
          -4010: ErrDef := '( X ) Invalid stored key format in
login process : -4010';
```

```
          -4020: ErrDef := '( X ) Reader cannot read after write
operation : -4020';
          -4030: ErrDef := '( X ) Decrement Failure (Empty) : -
4030';
     end;

end;




//'=========================================================
//'  Routine for converting Hex Significant Byte value
//'  to it's Decimal value.
//'=========================================================
Function StrHEX_Dec(StrHex: char) : Integer;
begin

      Case ord(StrHex) of
        ord('0'):  StrHEX_Dec := 0;
        ord('1'):  StrHEX_Dec := 1;
        ord('2'):  StrHEX_Dec := 2;
        ord('3'):  StrHEX_Dec := 3;
        ord('4'):  StrHEX_Dec := 4;
        ord('5'):  StrHEX_Dec := 5;
        ord('6'):  StrHEX_Dec := 6;
        ord('7'):  StrHEX_Dec := 7;
        ord('8'):  StrHEX_Dec := 8;
        ord('9'):  StrHEX_Dec := 9;
        ord('A'):  StrHEX_Dec := 10;
        ord('B'):  StrHEX_Dec := 11;
        ord('C'):  StrHEX_Dec := 12;
        ord('D'):  StrHEX_Dec := 13;
        ord('E'):  StrHEX_Dec := 14;
        ord('F'):  StrHEX_Dec := 15;
      end;

End;



Function GetTagType1(GetTagTypeA: Byte) : string;

begin

      //Function that explains the value of the TAGTYPE of the
Card.
      Case GetTagTypeA of
         1: GetTagType1 := 'Mifare Light';
         2: GetTagType1 := 'Mifare 1K';
         3: GetTagType1 := 'Mifare 4K';
         4: GetTagType1 := 'Mifare DESFire';
         5: GetTagType1 := 'Mifare Ultralight';
         6: GetTagType1 := 'JCOP30';
         7: GetTagType1 := 'Shanghai Transport';
         8: GetTagType1 := 'MPCOS Combi';
         128: GetTagType1 := 'ISO Type B, Calypso';

    end;

End;
```

```
//'=====================================================
//'  Routine for converting Hex to Decimal value.
//'=====================================================

Function Hex_Dec(val: TEdit): Byte;
var
MSB: Byte;
LSB: Byte;
Fbyte: Byte;
begin
              Fbyte := 0;
              MSB := 0;
              LSB := 0;

              MSB := StrHEX_Dec(val.Text[1]);
              LSB := StrHEX_Dec(val.Text[2]);
              Fbyte := (MSB * 16) + LSB;

Hex_Dec := Fbyte;

End;

//'=====================================================
//' Routine untuk ambil data dismart card
//'=====================================================

Function
PilihSektor(SectorFisik:Byte;Blok:Byte;handle:smallint):String;
var
Login :Byte;
Stored:Byte;
retkode:smallint;
NULL:byte;
ctr:integer;
dIsi:String;
rHandle: smallint;
retcode: smallint;
dataRead: array[0..15] of Byte;
begin
      NULL:=0;
      Login:=$FD;//LOGINnya default F
      Stored:=0;//stored number
      rHandle:=handle;

retkode:=ACR120_Login(rHandle,SectorFisik,Login,Stored,@NULL);
      Blok := SectorFisik * 4 + Blok;
      retcode := ACR120_Read(rHandle, Blok, @dataRead);
      if retcode<0 then
      begin
      MessageDlg(ErrDef(rHandle), mtInformation, [mbOK], 0);
      PilihSektor:='';
      end
      else
      begin
      dIsi := '';
        For ctr := 0 To 15 do //ambil karakter
        begin
          if dataRead[ctr]=$00 then
            begin
            end
          else
            begin
              dIsi := dIsi + chr(dataRead[ctr]);
            end;
```

```
        end;
      end;
    //MessageDlg(dIsi+'113123', mtInformation, [mbOK], 0);
    PilihSektor:=dIsi;
end;

//=========================================
//' Routine Tulis data pada kartu
//=========================================
Function
TulisSektorBlok(SectorFisik:Byte;Blok:Byte;handle:smallint;data:S
tring):boolean;
var
Login:byte;
Stored:Byte;
retkode:smallint;
NULL:byte;
rHandle: smallint;
retcode: smallint;
x,y,z:integer;
Sec:Byte;
begin
  NULL:=0;
  Sec:=SectorFisik;
  Login:=$FD;
  Stored:=0;
  rHandle:=handle;
  retkode:=ACR120_Login(rHandle,SectorFisik,Login,Stored,@NULL);
  Blok := SectorFisik * 4 + Blok;
  x:=length(data);
  if x<=16 then
  begin
  for y:=0 to 15 do
  begin
    dout[y]:=$00;
  end;
  for y:=0 to x do
    begin
       dout[y]:=ord(data[y+1]);
    end;
      retcode:=ACR120_Write(rHandle,Blok,@dout);
       if retkode<0 then
        begin
          MessageDlg(ErrDef(retcode), mtWarning, [mbOK], 0);
        end;
  end

  else if (x>16) and (x<=32) then
    begin
    for y:=0 to 15 do
    begin
      dout[y]:=$00;
    end;
    for y:=0 to 15 do
      begin
        dout[y]:=ord(data[y+1]);
      end;
    retcode:=ACR120_Write(rHandle,BLok,@dout);

    x:=x-16;
    for y:=0 to 15 do
    begin
      dout[y]:=$00;
    end;
    for y:=0 to x do
```

```
  begin
      dout[y]:=ord(data[y+1+16]);
  end;
  Blok:=Blok+1;
  retcode:=ACR120_Write(rHandle,BLok,@dout);
  end

else if (x>32) and (x<=48) then
begin
  for y:=0 to 15 do
  begin
    dout[y]:=$00;
  end;
  for y:=0 to 15 do
    begin
      dout[y]:=ord(data[y+1]);
    end;
  retcode:=ACR120_Write(rHandle,BLok,@dout);

  x:=x-16;
  for y:=0 to 15 do
    begin
      dout[y]:=ord(data[y+1+16]);
    end;
  Blok:=Blok+1;
  retcode:=ACR120_Write(rHandle,BLok,@dout);

  Sec:=Sec+1;
  retkode:=ACR120_Login(rHandle,Sec,Login,Stored,@NULL);

  x:=x-16;
  for y:=0 to x do
  begin
    dout[y]:=$00;
  end;
  for y:=0 to x do
  begin
    dout[y]:=ord(data[y+1+16+16]);
  end;
  Blok:=1;
  Blok := Sec* 4 + Blok;
  retcode:=ACR120_Write(rHandle,Blok,@dout);
end

  else if (x>48) and (x<=64) then
  begin
  for y:=0 to 15 do
  begin
    dout[y]:=$00;
  end;
  for y:=0 to 15 do
    begin
      dout[y]:=ord(data[y+1]);
    end;
  retcode:=ACR120_Write(rHandle,BLok,@dout);

  x:=x-16;
  for y:=0 to 15 do
    begin
      dout[y]:=ord(data[y+1+16]);
    end;
  Blok:=Blok+1;
  retcode:=ACR120_Write(rHandle,BLok,@dout);

  Sec:=Sec+1;
```

```
    retkode:=ACR120_Login(rHandle,Sec,Login,Stored,@NULL);

    x:=x-16;
    for y:=0 to 15 do
    begin
      dout[y]:=ord(data[y+1+16+16]);
    end;
    Blok:=1;
    Blok := Sec* 4 + Blok;
    retcode:=ACR120_Write(rHandle,Blok,@dout);

    x:=x-16;
    Blok:=Blok+1;
    for y:=0 to 15 do
    begin
      dout[y]:=$00;
    end;
    for y:=0 to x do
    begin
      dout[y]:=ord(data[y+1+16+16+16]);
    end;
    retcode:=ACR120_Write(rHandle,Blok,@dout);
    end;
end;

//=========================================
// Routine Tulis sektor login key A
//=========================================
Function
TulisSektorKeyA(SectorFisik:Byte;Blok:Byte;handle:smallint;data:s
tring):boolean;
var
Stored:Byte;
retkode:smallint;
pKey: array[0..5] of Byte;
rHandle: smallint;
retcode: smallint;
dataRead: array[0..15] of Byte;
NULL:byte;
Login:byte;
x,y,z:integer;
Sec:Byte;
begin
//login key A
      pKey[0] :=106; //6Ah
      pKey[1] :=101; //65h
      pKey[2] :=115; //73h
      pKey[3] :=117; //75h
      pKey[4] :=115; //73h
      pKey[5] := 00; //00
      Login:=$AA;//LOGINnya key A
      Stored:=0;//stored number
      rHandle:=handle;

retkode:=ACR120_Login(rHandle,SectorFisik,Login,Stored,@pKey);
      Sec:=SectorFisik;
      Blok:=Sec*4+Blok;
      x:=length(data);
  if x<=16 then
  begin
  for y:=0 to 15 do
  begin
    dout[y]:=$00;
  end;
  for y:=0 to x do
```

```
    begin
       dout[y]:=ord(data[y+1]);
    end;
      retkode:=ACR120_Write(rHandle,Blok,@dout);
       if retkode<0 then
        begin
          MessageDlg(ErrDef(retkode), mtWarning, [mbOK], 0);
        end;
  end;


end;
//=========================================
//' Routine pilih sektor dan login key A
//=========================================
Function
PilihSektorKeyA(SectorFisik:Byte;Blok:Byte;Login:Byte;handle:smal
lint):String;
var
Stored:Byte;
retkode:smallint;
pKey: array[0..5] of Byte;
ctr:integer;
dIsi:String;
rHandle: smallint;
retcode: smallint;
dataRead: array[0..15] of Byte;
begin
/////pkey dirubah ke bentuk decimal
      pKey[0] :=106; //6Ah
      pKey[1] :=101; //65h
      pKey[2] :=115; //73h
      pKey[3] :=117; //75h
      pKey[4] :=115; //73h
      pKey[5] := 00; //00
      Login:=$AA;//LOGINnya key A
      Stored:=0;//stored number
      rHandle:=handle;

retkode:=ACR120_Login(rHandle,SectorFisik,Login,Stored,@pKey);
      Blok := SectorFisik * 4 + Blok;
      retcode := ACR120_Read(rHandle, Blok, @dataRead);
      if retcode<0 then
      begin
      MessageDlg('Not Selected Card', mtInformation, [mbOK], 0);
      PilihSektorKeyA:='';
      exit;
      end
      else
      begin
      dIsi := '';
        For ctr := 0 To 15 do //ambil karakter dari NIS
        begin
          if dataRead[ctr]=$00 then
            begin
            end
          else
            begin
              dIsi := dIsi + chr(dataRead[ctr]);
            end;
        end;
      end;
    //MessageDlg(dIsi+'113123', mtInformation, [mbOK], 0);
    PilihSektorKeyA:=dIsi;
end;
```

```
//'=========================================================
//' Routine delay
//'=========================================================
function delay():string;
var
n1,n2:integer;
begin
for n1:=0 to 10000 do
  begin
    for n2:=0 to 9000 do
    begin
    end;
  end;
end;

//'=========================================================
//' Routine untuk cek aplikasi apakah ada diregister
//'=========================================================
function IsAppInRun(RunName: string): Boolean;
var
  Reg: TRegistry;
begin
  Reg := TRegistry.Create;
  with Reg do
  begin
    RootKey := HKEY_LOCAL_MACHINE;
    OpenKey('Software\Microsoft\Windows\CurrentVersion\Run',
False);
    Result := ValueExists(RunName);
    CloseKey;
    Free;
  end;
//'=========================================================
//' Routine untuk cek aplikasi apakah ada diregister
//'=========================================================
end;
end.
```

A-4.2 ACR120U

```
unit ACR120U;

interface
uses windows;

Const
//================================ Error Code
================================
ERR_READER_NO_RESPONSE = -5000;
ERR_ACR120_INTERNAL_UNEXPECTED = -1000;
ERR_ACR120_PORT_INVALID = -2000;
ERR_ACR120_PORT_OCCUPIED = -2010;
ERR_ACR120_HANDLE_INVALID = -2020;
ERR_ACR120_INCORRECT_PARAM = -2030;
ERR_ACR120_READER_NO_TAG = -3000;
ERR_ACR120_READER_READ_FAIL_AFTER_OP = -3010;
ERR_ACR120_READER_NO_VALUE_BLOCK = -3020;
ERR_ACR120_READER_OP_FAILURE = -3030;
ERR_ACR120_READER_UNKNOWN = -3040;
ERR_ACR120_READER_LOGIN_INVALID_STORED_KEY_FORMAT = -4010;
ERR_ACR120_READER_WRITE_READ_AFTER_WRITE_ERROR = -4020;
ERR_ACR120_READER_DEC_FAILURE_EMPTY = -4030;
ERR_READER_VALUE_INC_OVERFLOW = -4031;
ERR_READER_VALUE_OP_FAILURE = -4032;
```

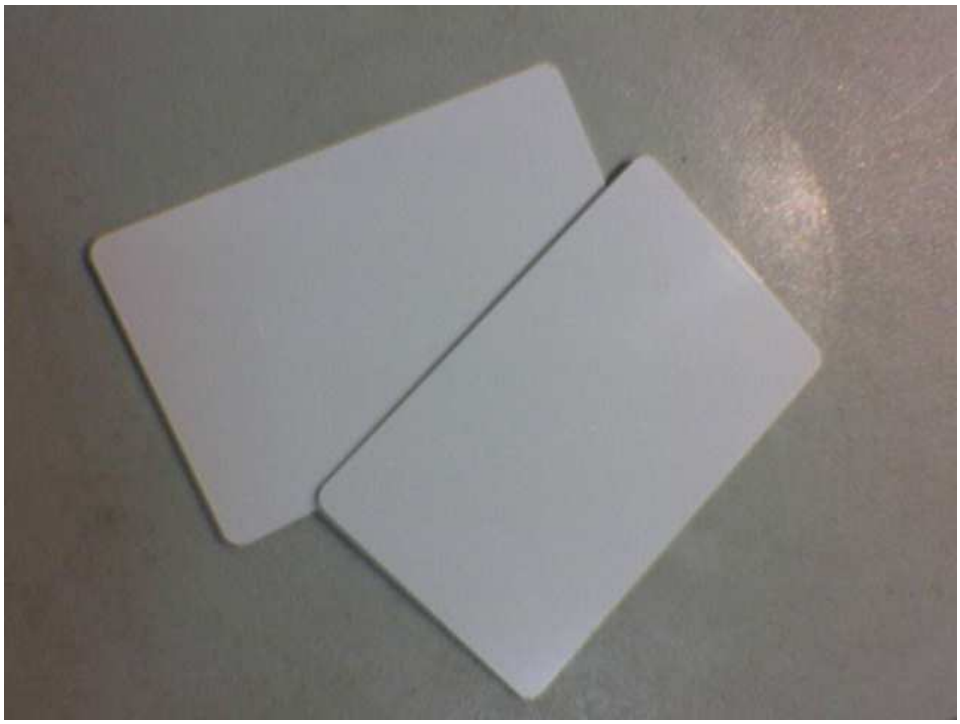
```
ERR_READER_VALUE_INVALID_BLOCK = -4033;
ERR_READER_VALUE_ACCESS_FAILURE = -4034 ;

//======================== Reader Port for AC_Open
==========================

ACR120_USB1 = 0;
ACR120_USB2 = 1;
ACR120_USB3 = 2;
ACR120_USB4 = 3;
ACR120_USB5 = 4;
ACR120_USB6 = 5;
ACR120_USB7 = 6;
ACR120_USB8 = 7;

//*======================== Key Type for ACR120_Login
========================

ACR120_LOGIN_KEYTYPE_A = $AA;
ACR120_LOGIN_KEYTYPE_B = $BB;
ACR120_LOGIN_KEYTYPE_DEFAULT_A = $AD;
ACR120_LOGIN_KEYTYPE_DEFAULT_B = $BD;
ACR120_LOGIN_KEYTYPE_DEFAULT_F = $FD;
ACR120_LOGIN_KEYTYPE_STORED_A = $AF;
ACR120_LOGIN_KEYTYPE_STORED_B = $BF;

type tReaderStatus = record

    // 0x01 = Type A; 0x02 = Type B; 0x03 = Type A + Type B
    MifareInterfaceType: Byte;
    // Bit 0 = Mifare Light;
    // Bit 1 = Mifare1K;
    // Bit 2 = Mifare 4K;
    // Bit 3 = Mifare DESFire
    // Bit 4 = Mifare UltraLight;
    // Bit 5 = JCOP30;
    // Bit 6 = Shanghai Transport;
    // Bit 7 = MPCOS Combi;
    //Bit 8 = ISO type B, Calypso
    // Bit 9 - Bit 31 = To be defined

    CardsSupported:array[0..3] of BYTE;
    CardOpMode: byte;
    // 0x00 = Type A; 0x01 = Type B TAG is being processed
    // 0xFF = No TAG is being processed.
    FWI: byte; // the current FWI value (time out value)
    RFU: byte; // To be defined
    RFU2: Integer; // to be defined
end;
    LPSCARD_tReaderStatus = ^tReaderStatus;

//-----------------------------------------------------------------
----------------
//Prototype section
//-----------------------------------------------------------------
----------------

type mpSN = array [0..3] of Byte;
type pLTSN = array [0..199] of Byte;
type mpKey = array [0..5] of Byte;

type mpBlockData = array [0..15] of Byte;
type mVersionInfo = array [0..13] of Byte;
```

```
type mpFirmwareVersion = byte;
     mRegData = byte;
     mpData = byte;
     mpResponseDataLength = byte;
     mpResponseData = byte;
     mpReceivedDataLength = byte;
     mpReceivedData = byte;
     mpVersionInfoLen = byte;
     mpVersionInfo = byte;
     mpUserPortState = byte;
     mpResultTagType = byte;
     mpResultTagLength = byte;
     mpResultSN = byte;
     mpValueData = LongInt;
     mpNewValue = LongInt;
     mpNumTagFound = byte;
     mpTagType = byte;
     mpTagLength = byte;
     mSN = byte;
     mpSendData = byte;
     mpEEPROMData = byte;


     TpFirmwareVersion = ^mpFirmwareVersion;
     TRegData = ^mRegData;
     TpData = ^mpData;
     TpResponseDataLength = ^mpResponseDataLength;
     TpResponseData = ^mpResponseData;
     TpReceivedDataLength = ^mpReceivedDataLength;
     TpReceivedData = ^mpReceivedData;
     TpVersionInfoLen = ^mpVersionInfoLen;
     TpVersionInfo = ^mpVersionInfo;
     TpEEPROMData = ^mpEEPROMData;
     TpUserPortState = ^mpUserPortState;
     TpResultTagType =  ^mpResultTagType;
     TpResultTagLength = ^mpResultTagLength;
     TpResultSN = ^mpResultSN;
     TpKey = ^mpKey;
     TpBlockData = ^mpBlockData;
     TpValueData = ^mpValueData;
     TpNewValue = ^mpNewValue;
     TpNumTagFound = ^mpNumTagFound;
     TpTagType = ^mpTagType;
     TpTagLength = ^mpTagLength;
     TpSN = ^mpSN;
     TSN = ^mSN;
     TpSendData = ^mpSendData;



  ////////////////////////////
 //   USB   READER COMMANDS  //
////////////////////////////

function ACR120_Open (ReaderPort:byte):smallint; stdcall;
external 'ACR120U.dll';
function ACR120_Close(hReader:smallint):smallint; stdcall;
external 'ACR120U.dll';

function ACR120_Reset(hReader:smallint):smallint; stdcall;
external 'ACR120U.dll';
function ACR120_Status(hReader:smallint; pFirmwareVersion:
TpFirmwareVersion; pReaderStatus
```

```
:LPSCARD_tReaderStatus):smallint; stdcall; external
'ACR120U.dll';

function ACR120_ReadRC500Reg(hReader:smallint; RegNo:Byte;
pRegData: TRegData):smallint; stdcall; external 'ACR120U.dll';
function ACR120_WriteRC500Reg(hReader:smallint; RegNo:Byte;
TRegData:Byte):smallint; stdcall; external 'ACR120U.dll';

function ACR120_ReadRC531Reg(hReader:smallint; RegNo:Byte;
pRegData: TRegData):smallint; stdcall;
function ACR120_WriteRC531Reg(hReader:smallint; RegNo:Byte;
TRegData:Byte):smallint; stdcall;

function ACR120_DirectSend(hReader:smallint; DataLength:Byte;
pData: TpData; pResponseDataLength: TpResponseDataLength;
pResponseData: TpResponseData; TimedOut:smallint):smallint;
stdcall; external 'ACR120U.dll';
function ACR120_DirectReceive(hReader:smallint;
RespectedDataLength:Byte; pReceivedDataLength:
TpReceivedDataLength; pReceivedData: TpReceivedData;
TimedOut:smallint):smallint; stdcall; external 'ACR120U.dll';

function ACR120_RequestDLLVersion(pVersionInfoLen:
TpVersionInfoLen; pVersionInfo: TpVersionInfo):smallint; stdcall;
external 'ACR120U.dll';
function ACR120_ReadEEPROM(hReader:smallint; RegNo:Byte;
pEEPROMData: TpEEPROMData):smallint; stdcall; external
'ACR120U.dll';

function ACR120_WriteEEPROM(hReader:smallint; RegNo:Byte;
eePROMData:Byte):smallint; stdcall; external 'ACR120U.dll';
function ACR120_ReadUserPort(hReader:smallint; pUserPortState:
TpUserPortState):smallint; stdcall; external 'ACR120U.dll';

function ACR120_WriteUserPort(hReader:smallint;
userPortState:Byte):smallint; stdcall; external 'ACR120U.dll';

//=================================================================
======================
// CARD COMMANDS
//=================================================================
======================

function ACR120_Select(hReader:smallint; pResultTagType:
TpResultTagType; pResultTagLength: TpResultTagLength; pResultSN:
TpResultSN):smallint; stdcall; external 'ACR120U.dll';
function ACR120_Login(hReader:smallint; Sector:Byte;
keyType:Byte; StoredNo:Byte; pKey: TpKey):smallint; stdcall;
external 'ACR120U.dll';

function ACR120_Read(hReader:smallint; Block:Byte; pBlockData:
TpBlockData):smallint; stdcall; external 'ACR120U.dll';
function ACR120_ReadValue(hReader:smallint; Block:Byte;
pValueData: TpValueData):smallint; stdcall; external
'ACR120U.dll';

function ACR120_Write(hReader:smallint; Block:Byte; pBlockData:
TpBlockData):smallint; stdcall; external 'ACR120U.dll';
function ACR120_WriteValue(hReader:smallint; Block:Byte;
ValueData:LongInt):smallint; stdcall; external 'ACR120U.dll';

function ACR120_WriteMasterKey(hReader:smallint; KeyNo:Byte;
pKey: TpKey):smallint; stdcall; external 'ACR120U.dll';
function ACR120_Inc(hReader:smallint; Block:Byte; value:LongInt;
pNewValue: TpNewValue):smallint; stdcall; external 'ACR120U.dll';
```

```
function ACR120_Dec(hReader:smallint; Block:Byte; value:LongInt;
pNewValue: TpNewValue):smallint; stdcall; external 'ACR120U.dll';
function ACR120_Copy(hReader:smallint; srcBlock:Byte;
desBlock:Byte; pNewValue: TpNewValue):smallint; stdcall; external
'ACR120U.dll';

function ACR120_Power(hReader:smallint; State: Byte):smallint;
stdcall; external 'ACR120U.dll';
function ACR120_ListTags(hReader:smallint; pNumTagFound:
TpNumTagFound; pTagType: TpTagType; pTagLength: TpTagLength; pSN:
TpSN):smallint; stdcall; external 'ACR120U.dll';

function ACR120_MultiTagSelect(hReader:smallint; TagLength:Byte;
SN: TSN; pResultTagType: TpResultTagType; pResultTagLength:
TpResultTagLength; pResultSN: TpResultSN):smallint; stdcall;
external 'ACR120U.dll';
function ACR120_TxDataTelegram(hReader:smallint;
SendDataLength:Byte; pSendData: TpSendData; pReceivedDataLength:
TpReceivedDataLength; pReceivedData: TpReceivedData):smallint;
stdcall; external 'ACR120U.dll';



implementation
function ACR120_ReadRC531Reg(hReader:smallint; RegNo:Byte;
pRegData: TRegData):smallint;
begin
    ACR120_ReadRC500Reg(hReader, RegNo, pRegData);
end;

function ACR120_WriteRC531Reg(hReader:smallint; RegNo:Byte;
TRegData:Byte):smallint;
begin
    ACR120_WriteRC500Reg(hReader, RegNo, TRegData);
end;

end.
```

LAMPIRAN B
FOTO ALAT



Gambar B-1 Smart Card Mifare 1K

Gambar B-2 Reader/Writer ACR120U

LAMPIRAN C
DATA SHEET DAN REFERENSI MANUAL

1 FEATURES

1.1 MIFARE® RF Interface (ISO/IEC 14443 A)

- Contactless transmission of data and supply energy (no battery needed)
- Operating distance: Up to 100mm (depending on antenna geometry)
- Operating frequency: 13.56 MHz
- Fast data transfer: 106 kbit/s
- High data integrity: 16 Bit CRC, parity, bit coding, bit counting
- True anticollision
- Typical ticketing transaction: < 100 ms (including backup management)

1.2 EEPROM

- 1 Kbyte, organized in 16 sectors with 4 blocks of 16 bytes each (one block consists of 16 byte)
- User definable access conditions for each memory block
- Data retention of 10 years.
- Write endurance 100.000 cycles

1.3 Security

- Mutual three pass authentication (ISO/IEC DIS9798-2)
- Data encryption on RF-channel with replay attack protection
- Individual set of two keys per sector (per application) to support multi-application with key hierarchy
- Unique serial number for each device
- Transport key protects access to EEPROM on chip delivery

2 GENERAL DESCRIPTION

Philips has developed the MIFARE® MF1 IC S50 to be used in contactless smart cards according to ISO/IEC 14443A. The communication layer (MIFARE® RF Interface) complies to parts 2 and 3 of the ISO/IEC 14443A standard. The security layer sports the field-proven CRYPTO 1 stream cipher for secure data exchange of the MIFARE® Classic family.

2.1 Contactless Energy and Data Transfer

In the MIFARE® system, the MF1 IC S50 is connected to a coil with a few turns and then embedded in plastic to form the passive contactless smart card. No battery is needed. When the card is positioned in the proximity of the Read Write Device (R/WD) antenna, the high speed RF communication interface allows to transmit data with 106 kBit/s.

2.2 Anticollision

An intelligent anticollision function allows to operate more than one card in the field simultaneously. The anticollision algorithm selects each card individually and ensures that the execution of a transaction with a selected card is performed correctly without data corruption resulting from other cards in the field.

2.3 User Convenience

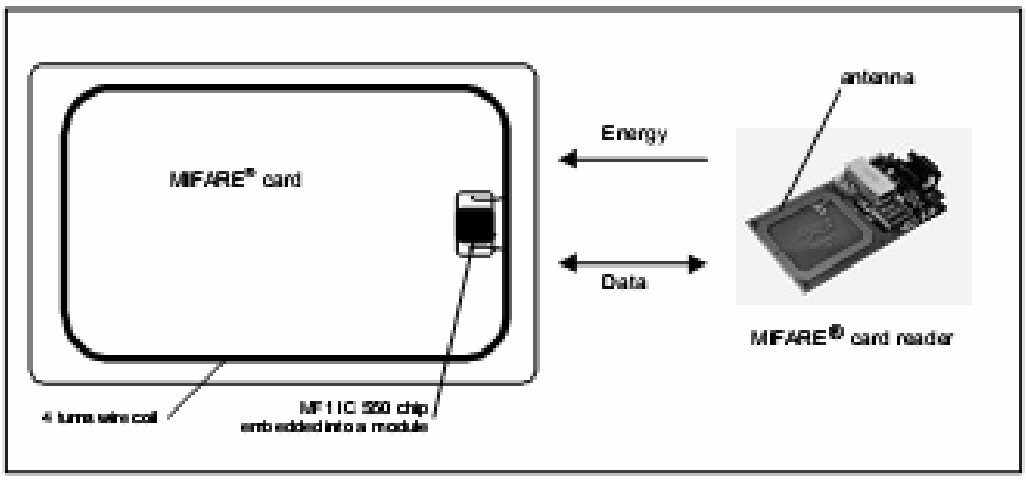
The MIFARE® system is designed for optimal user convenience. The high data transmission rate for example allows complete ticketing transactions to be handled in less than 100 ms. Thus, the MIFARE® card user is not forced to stop at the R/WD antenna leading to a high throughput at gates and reduced boarding times onto busses. The MIFARE® card may also remain in the wallet during the transaction, even if there are coins in it.

2.4 Security

Special emphasis has been placed on security against fraud. Mutual challenge and response authentication, data ciphering and message authentication checks protect the system from any kind of tampering and thus make it attractive for ticketing applications. Serial numbers, which can not be altered, guarantee the uniqueness of each card.

2.5 Multi-application Functionality

The MIFARE® system offers real multi-application functionality comparable to the features of a processor card. Two different keys for each sector support systems using key hierarchies.



2.6 Delivery Options

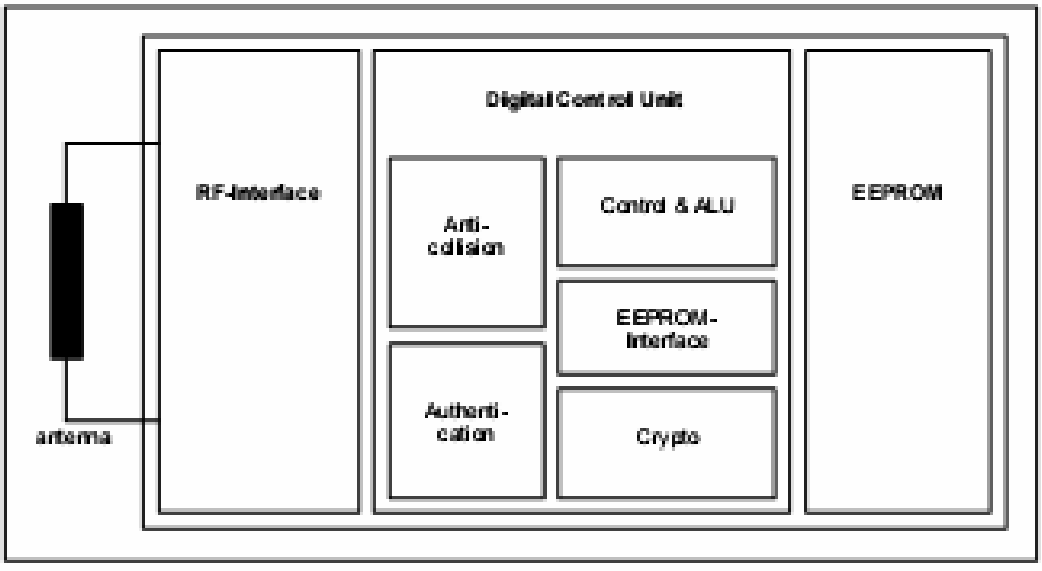
- Die on wafer
- Bumped die on wafer
- Chip Card Module

3 FUNCTIONAL DESCRIPTION

3.1 Block Description

The MF1 IC S50 chip consists of the 1 Kbyte EEPROM, the RF-Interface and the Digital Control Unit. Energy and data are transferred via an antenna, which consists of a coil with a few turns directly connected to the MF1 IC S50. No further external components are necessary. (For details on antenna design please refer to the document *MIFARE® Card IC Coil Design Guide*.)

- RF-Interface:
  – Modulator/Demodulator
  – Rectifier
  – Clock Regenerator
  – Power On Reset
  – Voltage Regulator
- Anticollision: Several cards in the field may be selected and operated in sequence
- Authentication: Preceding any memory operation the authentication procedure ensures that access to a block is only possible via the two keys specified for each block
- Control & Arithmetic Logic Unit: Values are stored in a special redundant format and can be incremented and decremented
- EEPROM-Interface
- Crypto unit: The field-proven CRYPTO1 stream cipher of the MIFARE® Classic family ensures a secure data exchange
- EEPROM: 1 Kbyte are organized in 16 sectors with 4 blocks each. A block contains 16 bytes. The last block of each sector is called "trailer", which contains two secret keys and programmable access conditions for each block in this sector.



3.2 Communication Principle

The commands are initiated by the R/W/D and controlled by the Digital Control Unit of the MF1 IC S50 according to the access conditions valid for the corresponding sector.

3.2.1 REQUEST STANDARD / ALL

After Power On Reset (POR) of a card it can answer to a request command - sent by the R/W/D to all cards in the antenna field - by sending the answer to request code (ATQA according to ISO/IEC 14443A).

3.2.2 ANTICOLLISION LOOP

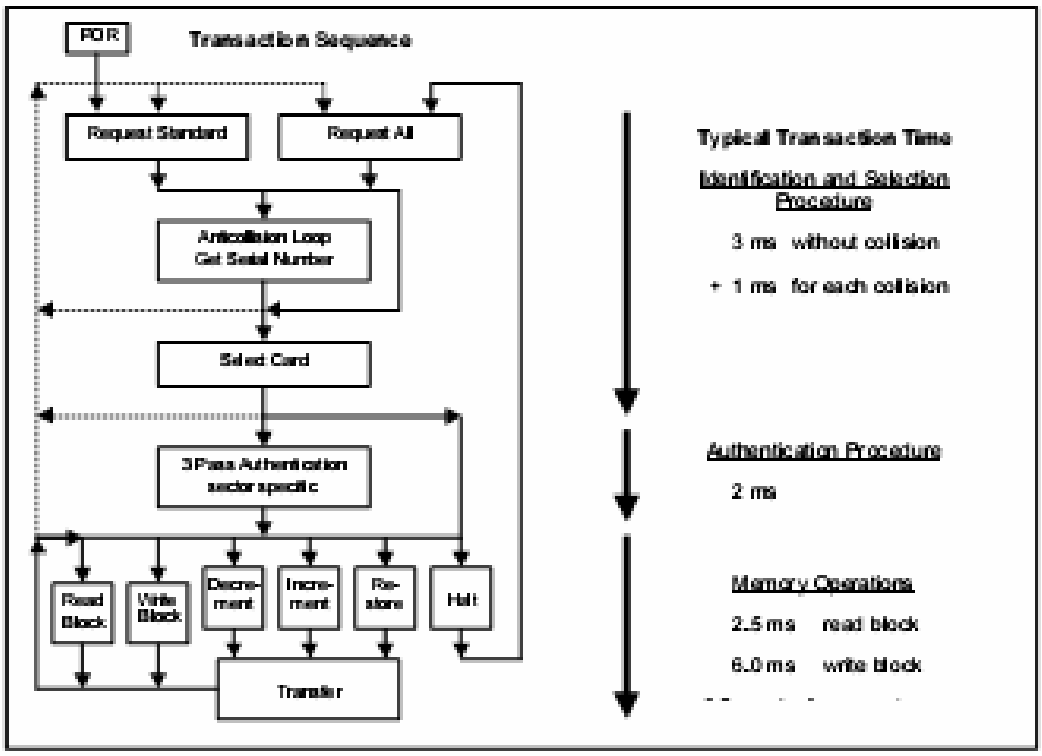
In the anticollision loop the serial number of a card is read. If there are several cards in the operating range of the R/W/D, they can be distinguished by their unique serial numbers and one can be selected (select card) for further transactions. The unselected cards return to the standby mode and wait for a new request command.

3.2.3 SELECT CARD

With the select card command the R/W/D selects one individual card for authentication and memory related operations. The card returns the Answer To Select(ATS) code (= 08h), which determines the type of the selected card. Please refer to the document *MIFARE® Standardised Card Type Identification Procedure* for further details.

3.2.4 3 PASS AUTHENTICATION

After selection of a card the R/W/D specifies the memory location of the following memory access and uses the corresponding key for the 3 pass authentication procedure. After a successful authentication all memory operations are encrypted.



3.2.5 MEMORY OPERATIONS

After authentication any of the following operations may be performed:

- Read block
- Write block
- Decrement:Decrements the contents of a block and stores the result in a temporary internal data-register
- Increment: Increments the contents of a block and stores the result in the data-register
- Restore: Moves the contents of a block into the data-register
- Transfer: Writes the contents of the temporary internal data-register to a value block

3.3 Data Integrity

Following mechanisms are implemented in the contactless communication link between RWD and card to ensure very reliable data transmission:

- 16 bits CRC per block
- Parity bits for each byte
- Bit count checking
- Bit coding to distinguish between "1", "0", and no information
- Channel monitoring (protocol sequence and bit stream analysis)

3.4 Security

To provide a very high security level a three pass authentication according to ISO 9798-2 is used.

3.4.1 THREE PASS AUTHENTICATION SEQUENCE

a) The RWD specifies the sector to be accessed and chooses key A or B.
b) The card reads the secret key and the access conditions from the sector trailer. Then the card sends a random number as the challenge to the RWD (pass one).
c) The RWD calculates the response using the secret key and additional input. The response, together with a random challenge from the RWD, is then transmitted to the card (pass two).
d) The card verifies the response of the RWD by comparing it with its own challenge and then it calculates the response to the challenge and transmits it (pass three).
e) The RWD verifies the response of the card by comparing it to its own challenge.

After transmission of the first random challenge the communication between card and RWD is encrypted.

3.5 RF Interface

The RF-interface is according to the standard for contactless smart cards ISO/IEC 14443A.

The carrier field from the RWD is always present (with short pauses when transmitting), because it is used for the power supply of the card.

For both directions of data communication there is only one start bit at the beginning of each frame. Each byte is transmitted with a parity bit (odd parity) at the end. The LSB of the byte with the lowest address of the selected block is transmitted first. The maximum frame length is 163 bits (16 data bytes + 2 CRC bytes = 16 * 9 + 2 * 9 + 1 start bit).

3.8 Memory Organisation

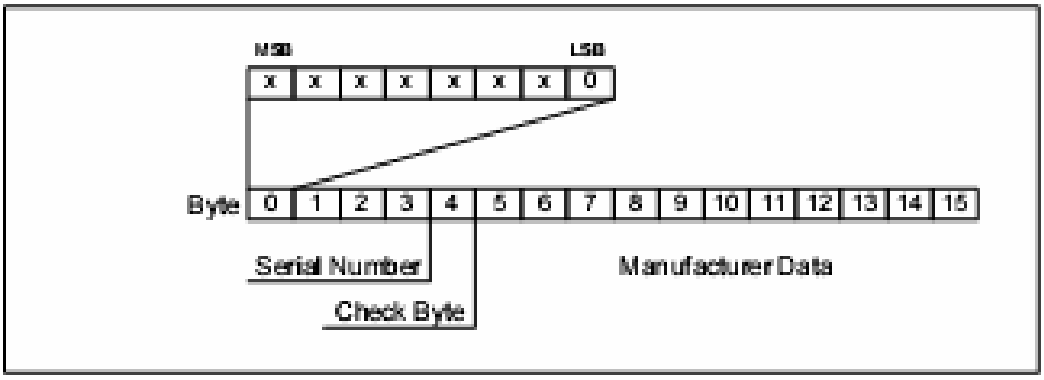
The 1024 x 8 bit EEPROM memory is organized in 16 sectors with 4 blocks of 16 bytes each.

In the erased state the EEPROM cells are read as a logical "0", in the written state as a logical "1".

Sector	Block	Byte Number within a Block 0 1 2 3 4 5	6 7 8 9	10 11 12 13 14 15	Description
15	3	Key A	Access Bits	Key B	Sector Trailer 15
	2				Data
	1				Data
	0				Data
14	3	Key A	Access Bits	Key B	Sector Trailer 14
	2				Data
	1				Data
	0				Data
:	:				
:	:				
:	:				
1	3	Key A	Access Bits	Key B	Sector Trailer 1
	2				Data
	1				Data
	0				Data
0	3	Key A	Access Bits	Key B	Sector Trailer 0
	2				Data
	1				Data
	0				Manufacturer Block

3.6.1 MANUFACTURER BLOCK

This is the first data block (block 0) of the first sector (sector 0). It contains the IC manufacturer data. Due to security and system requirements this block is write protected after having been programmed by the IC manufacturer at production.



3.6.2 DATA BLOCKS

All sectors contain 3 blocks of 16 bytes for storing data (Sector 0 contains only two data blocks and the read-only manufacturer block).

The data blocks can be configured by the access bits as

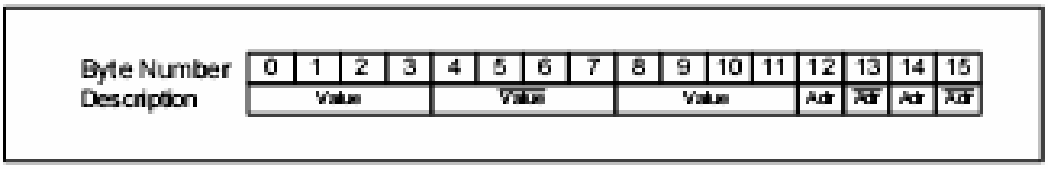
- read/write blocks for e.g. contactless access control or
- value blocks for e.g. electronic purse applications, where additional commands like increment and decrement for direct control of the stored value are provided.

An authentication command has to be carried out before any memory operation in order to allow further commands.

3.6.2.1 *Value Blocks*

The value blocks allow to perform electronic purse functions (valid commands: *read*, *write*, *increment*, *decrement*, *restore*, *transfer*).
The value blocks have a fixed data format which permits error detection and correction and a backup management.
A value block can only be generated through a write operation in the value block format:

- Value: Signifies a signed 4-byte value. The lowest significant byte of a value is stored in the lowest address byte. Negative values are stored in standard 2's complement format. For reasons of data integrity and security, a value is stored three times, twice non-inverted and once inverted.
- Adr: Signifies a 1-byte address, which can be used to save the storage address of a block, when implementing a powerful backup management. The address byte is stored four times, twice inverted and non-inverted. During *increment*, *decrement*, *restore* and *transfer* operations the address remains unchanged. It can only be altered via a write command.



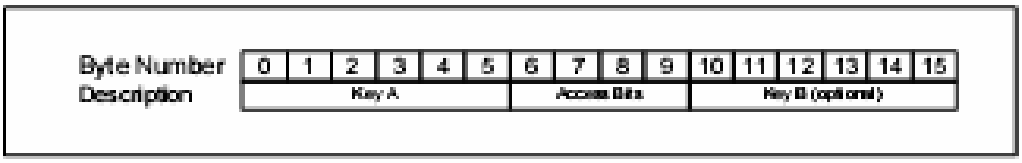
3.6.3 SECTOR TRAILER (BLOCK 3)

Each sector has a sector trailer containing the

- secret keys A and B(optional), which return logical "0"s when read and
- the access conditions for the four blocks of that sector, which are stored in bytes 6...9. The access bits also specify the type (read/write or value) of the data blocks.

If key B is not needed, the last 6 bytes of block 3 can be used as data bytes.

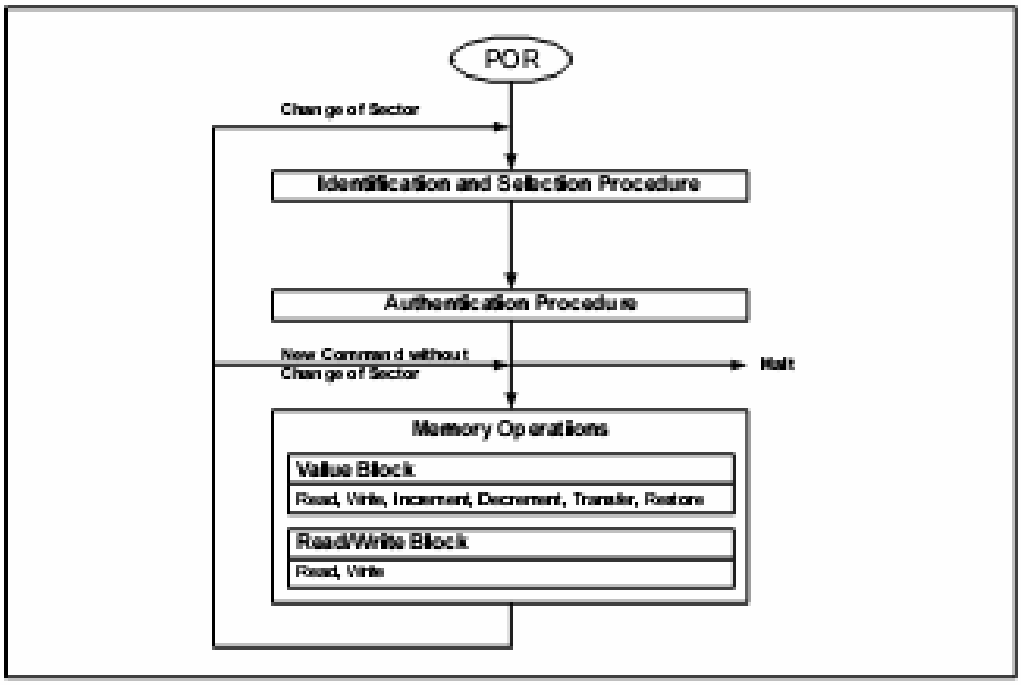
Byte 9 of the sector trailer is available for user data. For this byte apply the same access rights as for byte 6, 7 and 8.



3.7 Memory Access

Before any memory operation can be carried out, the card has to be selected and authenticated as described previously.

The possible memory operations for an addressed block depend on the key used and the access conditions stored in the associated sector trailer.



Memory Operations		
Operation	Description	Valid for Block Type
Read	reads one memory block	read/write, value and sector trailer
Write	writes one memory block	read/write, value and sector trailer
Increment	increments the contents of a block and stores the result in the internal data register	value
Decrement	decrements the contents of a block and stores the result in the internal data register	value
Transfer	writes the contents of the internal data register to a block	value
Restore	reads the contents of a block into the internal data register	value

3.7.1 ACCESS CONDITIONS

The access conditions for every data block and sector trailer are defined by 3 bits, which are stored non-inverted and inverted in the sector trailer of the specified sector.

The access bits control the rights of memory access using the secret keys A and B. The access conditions may be altered, provided one knows the relevant key and the current access condition allows this operation.

Note: In the following description the access bits are mentioned in the non-inverted mode only.

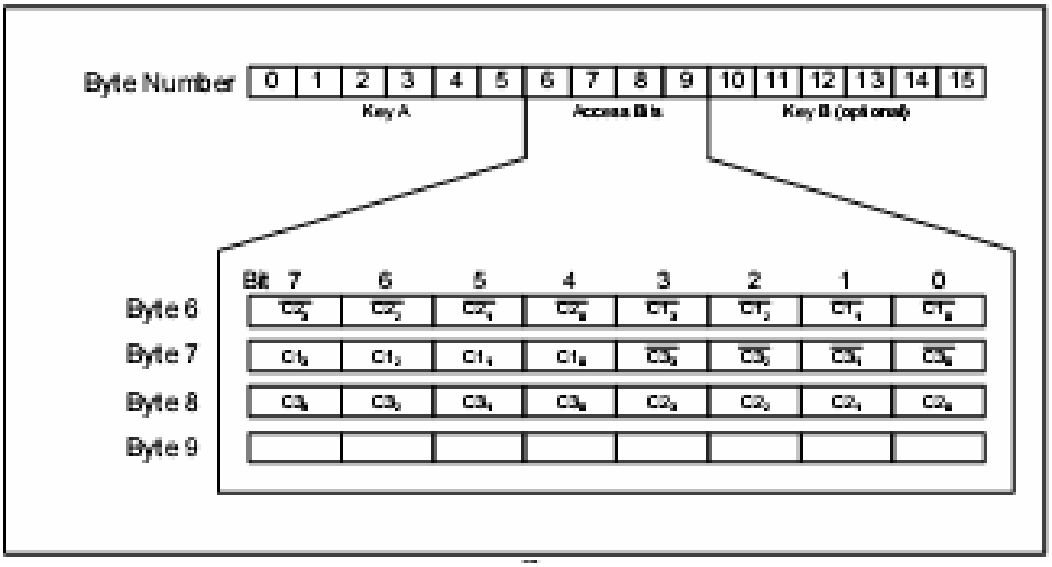
The internal logic of the MF1 IC S50 ensures that the commands are executed only after an authentication procedure or never.

Access Bits	Valid Commands		Block	Description
$C1_3 C2_3 C3_3$	read, write	→	3	sector trailer
$C1_2 C2_2 C3_2$	read, write, increment, decrement, transfer, restore	→	2	data block
$C1_1 C2_1 C3_1$	read, write, increment, decrement, transfer, restore	→	1	data block
$C1_0 C2_0 C3_0$	read, write, increment, decrement, transfer, restore	→	0	data block

Note: With each memory access the internal logic verifies the format of the access conditions. If it detects a format violation the whole sector is irreversible blocked.

3.7.2 ACCESS CONDITIONS FOR THE SECTOR TRAILER

Depending on the access bits for the sector trailer (block 3) the read/write access to the keys and the



access bits is specified as 'never', 'key A', 'key B' or key A|B' (key A or key B).

On chip delivery the access conditions for the sector trailers and key A are predefined as transport configuration. Since key B may be read in transport configuration, new cards must be authenticated with key A.

Since the access bits themselves can also be blocked, special care should be taken during personalization of cards.

Note: the grey marked lines are access conditions where key B is readable and may be used for data.

Access bits			Access condition for						Remark
			KEYA		Access bits		KEYB		
C1	C2	C3	read	write	read	write	read	write	
0	0	0	never	key A	key A	never	key A	key A	Key B may be read
0	1	0	never	never	key A	never	key A	never	Key B may be read
1	0	0	never	key B	key A|B	never	never	key B	
1	1	0	never	never	key A|B	never	never	never	
0	0	1	never	key A	key A	key A	key A	key A	Key B may be read, transport configuration
0	1	1	never	key B	key A|B	key B	never	key B	
1	0	1	never	never	key A|B	key B	never	never	
1	1	1	never	never	key A|B	never	never	never	

3.7.3 ACCESS CONDITIONS FOR DATA BLOCKS

Depending on the access bits for data blocks (blocks 0...2) the read/write access is specified as 'never', 'key A', 'key B' or 'key A|B' (key A or key B). The setting of the relevant access bits defines the application and the corresponding applicable commands.

- Read/write block: The operations read and write are allowed.
- Value block: Allows the additional value operations *increment*, *decrement*, *transfer* and *restore*. In one case ('001') only *read* and *decrement* are possible for a non-rechargeable card. In the other case ('110') recharching is possible by using key B.
- Manufacturer block: The read-only condition is not affected by the access bits setting!!
- Key management: In transport configuration key A must be used for authentication[1] .

Access bits			Access condition for				Application
C1	C2	C3	read	write	increment	decrement, transfer, restore	
0	0	0	key A|B[1]	key A|B[1]	key A|B[1]	key A|B[1]	transport configuration
0	1	0	key A|B[1]	never	never	never	read/write block
1	0	0	key A|B[1]	key B[1]	never	never	read/write block
1	1	0	key A|B[1]	key B[1]	key B[1]	key A|B[1]	value block
0	0	1	key A|B[1]	never	never	key A|B[1]	value block
0	1	1	key B[1]	key B[1]	never	never	read/write block
1	0	1	key B[1]	never	never	never	read/write block
1	1	1	never	never	never	never	read/write block

[1] If Key B may be read in the corresponding Sector Trailer it cannot serve for authentication (all grey marked lines in previous table). Con sequence: If the RWD tries to authenticate any block of a sector with key B using grey marked access conditions, the card will refuse any subsequent memory access after authentication.

4 DEFINITIONS

Data sheet status	
Objective specification	This data sheet contains target or goal specifications for product development.
Preliminary specification	This data sheet contains preliminary data; supplementary data may be published later.
Product specification	This data sheet contains final product specifications.
Limiting values	
Limiting values given are in accordance with the Absolute Maximum Rating System (IEC 134). Stress above one or more of the limiting values may cause permanent damage to the device. These are stress ratings only and operation of the device at these or at any other conditions above those given in the Characteristics section of the specification is not implied. Exposure to limiting values for extended periods may affect device reliability.	
Application information	
Where application information is given, it is advisory and does not form part of the specification.	

5 LIFE SUPPORT APPLICATIONS

These products are not designed for use in life support appliances, devices, or systems where malfunction of these products can reasonably be expected to result in personal injury. Philips customers using or selling these products for use in such applications do so on their own risk and agree to fully indemnify Philips for any damages resulting from such improper use or sale.

6 REVISION HISTORY

Table 1 Functional Specification MF1 IC S50 Revision History

REVISION	DATE	CPCN	PAGE	DESCRIPTION
5.1	0501	2001 05013	9, 10	New Coding Manufacturer Block
5.0	1199			New Layout. Revised. Includes MF1 IC S50 05 silicon.
1.0		-		Initial version.

Philips Semiconductors - a worldwide company

Argentina: see South America
Australia: 34 Waterloo Road, NORTH RYDE, NSW 2113,
Tel. +612 9805 4455, Fax. +612 9805 4466
Austria: Computerstraße 6, A-1101 WIEN, P.O.Box 213,
Tel. +431 60 101, Fax. +431 60 101 1210
Belarus: Hotel Minsk Business Center, Bld. 3, r.1211, Volodarski Str. 6,
220050 MINSK, Tel. +375 172 200 733, Fax. +375 172 200 773
Belgium: see The Netherlands
Brazil: see South America
Bulgaria: Philips Bulgaria Ltd., Energoproject, 15th floor,
51 James Bourchier Blvd., 1407 SOFIA
Tel. +359 2 689 211, Fax. +359 2 689 102
Canada: Philips Semiconductors/Components,
Tel. +1 800 234 7381
China/Hong Kong: 501 Hong Kong Industrial Technology Centre,
72 Tai Chee Avenue, Kowloon Tong, HONG KONG,
Tel. +852 2319 7888, Fax. +852 2319 7700
Colombia: see South America
Czech Republic: see Austria
Denmark: Prags Boulevard 80, PB 1919, DK-2300 COPENHAGEN S,
Tel. +45 33 29 3333, Fax. +45 31 57 1949
Finland: Sinikalliontie 3, FIN-02630 ESPOO,
Tel. +358 9 615 800, Fax. +358 9 61 580 920
France: 4 Rue du Port-aux-Vins, BP 317, 92156 SURESNES Cedex,
Tel. +331 40 99 6161, Fax. +331 40 99 6427
Germany: Hammerbrookstraße 69, D-20097 HAMBURG,
Tel. +49 40 23 53 60, Fax. +49 40 23 536 300
Greece: No. 15, 25th March Street, GR 17778 TAVROS/ATHENS,
Tel. +301 4894 339/239, Fax. +301 4814 240
Hungary: see Austria
India: Philips INDIA Ltd., Shivsagar Estate, A Block, Dr. Annie Besant Rd.
Worli, MUMBAI 400 018, Tel. +91 22 4938 541, Fax. +91 22 4938 722
Indonesia: see Singapore
Ireland: Newstead, Clonskeagh, DUBLIN 14,
Tel. +353 1 764 5000, Fax. +353 1 764 5099
Israel: RAPAC Electronics, 7 Kehilat Saloniki St., TEL AVIV 61180
Tel. +972 3 645 0444, Fax. +972 3 649 1007
Italy: Philips Semiconductors, Piazza IV Novembre 3,
20124 MILANO, Tel. +39 2 6752 2531, Fax. +39 2 6752 2557
Japan: Philips Bldg. 13-37, Kohnan 2-chome, Minato-ku, TOKYO 108,
Tel. +81 3 3740 5130, Fax. +81 3 3740 5077
Korea: Philips House, 260-199, Itaewon-dong, Yongsan-ku, SEOUL,
Tel. +822 709 1412, Fax. +822 709 1415
Malaysia: No. 76 Jalan Universiti, 46300 PETALING JAYA, Selangor,
Tel. +603 750 5214, Fax. +603 757 4880
Mexico: 5900 Gateway East, Suite 200, EL PASO, Texas 79905,
Tel. +9 800 234 7381
Middle East: see Italy
Netherlands: Postbus 90050, 5600 PB EINDHOVEN, Bldg. VB,
Tel. +31 40 27 82785, Fax. +31 40 27 88399
New Zealand: 2 Wagener Place, C.P.O. Box 1041, AUCKLAND,
Tel. +64 9 849 4160, Fax. +64 9 849 7811
Norway: Box 1, Manglerud 0612, OSLO,
Tel. +47 22 74 8000, Fax. +47 22 74 8341
Philippines: Philips Semiconductors Philippines Inc.,
106 Valero St. Salcedo Village, P.O. Box 2108 MCC, MAKATI,
Metro MANILA, Tel. +632 816 6380, Fax. +632 817 3474
Poland: Ul. Lukiska 10, PL 04-123 WARSZAWA,
Tel. +4822 612 2831, Fax. +4822 612 2327
Portugal: see Spain
Romania: see Italy
Russia: Philips Russia, Ul. Usatcheva 35A, 119048 MOSCOW,
Tel. +7095 247 9145, Fax. +7095 247 9144
Singapore: Lorong 1, Toa Payoh, SINGAPORE 1232,
Tel. +65 350 2538, Fax. +65 251 6500
Slovakia: see Austria
Slovenia: see Italy
South Africa: S.A. Philips Pty Ltd., 195-215 Main Road Martindale,
2092 JOHANNESBURG, P.O.Box 7430 Johannesburg 2000,
Tel. +2711 470 5911, Fax. +2711 470 5494
South America: Rua do Rocio 220, 5th floor, Suite 51,
04552-903 São Paulo, SÃO PAULO - SP, Brazil,
Tel. +9811 821 2333, Fax. +9811 829 1849
Spain: Balmes 22, 08007 BARCELONA,
Tel. +343 301 6312, Fax. +343 301 4107
Sweden: Kottbygatan 7, Akalla, S-164 85 STOCKHOLM,
Tel. +468 632 2000, Fax. +468 632 2745
Switzerland: Allmendstraße 140, CH-8027 ZÜRICH,
Tel. +411 488 2686, Fax. +411 481 7730
Taiwan: Philips Taiwan Ltd., 23/30F, 66,
Chung Hsiao West Road, Sec. 1, P.O.Box 22978,
TAIPEI 100, Tel. +8862 382 4443, Fax. +8862 382 4444
Thailand: Philips Electronics (Thailand) Ltd.,
209/2 Sanpavuth-Bangna Road Prakanong, BANGKOK 10260,
Tel. +662 745 4090, Fax. +662 398 0793
Turkey: Talapasa Cad. No. 6, 80640 GÜLTEPE/ISTANBUL,
Tel. +9012 279 2770, Fax. +9012 282 6707
Ukraine: Philips Ukraine, 4 Patrice Lumumba Str., Building B, Floor 7,
252042 KIEV, Tel. +380 44 264 2776, Fax. +380 44 268 0461
United Kingdom: Philips Semiconductors Ltd., 276 Bath Road, Hayes,
MIDDLESEX UB3 5BX, Tel. +44 181 730 5000, Fax. +44 181 754 8421
United States: 811 Argues Avenue, SUNNYVALE, CA 94088-3409,
Tel. +1 800 234 7381
Uruguay: see South America
Vietnam: see Singapore
Yugoslavia: Philips, Trg N. Pasica 5/v, 11000 BEOGRAD,
Tel. +381 11 625 344, Fax. +381 11 635 777

Published by:
Philips Semiconductors Gratkorn GmbH, Mikron-Weg 1, A-8101 Gratkorn, Austria Fax: +43 3124 299 - 270

For all other countries apply to: Philips Semiconductors, Marketing & Sales Communications, Internet: http://www.semiconductors.philips.com
Building BE-p, P.O.Box 218, 5600 MD EINDHOVEN, The Netherlands, Fax: +31 40 27 31811

Philips
Semiconductors

PHILIPS

Advanced Card Systems Ltd.

ACR120 Contactless Reader/Writer

GUIDELINES IN PORTING APPLICATIONS FROM ACR120S TO ACR120U AND VICE-VERSA

Version 1.2 11-2006

Unit 1008, 10th Floor, Hongkong International Trade and Exhibition Centre
1 Trademart Drive, Kowloon Bay, Hong Kong
Tel: +852 2796 7873 Fax: +852 2796 1286 Email: info@acs.com.hk Website: www.acs.com.hk

Contents

Advanced Card Systems Ltd.
Unit 1008, 10th Floor
Hong Kong International Trade and Exhibition Centre
1 Trademart Drive, Kowloon Bay, Hong Kong

Tel: +852 2796 7873
Website: www.acs.com.hk

Fax: +852 2796 1286
Email: info@acs.com.hk

I. Overview

This document serves as a guide to properly port your ACR120S applications to the new ACR120U device and vice-versa.

Note:

1. Make sure you have the right ACR120 Device.
2. Make sure you have installed the right ACR120 driver. The ACR120 Drivers of both interface can be downloaded from our website (www.acs.com.hk).

Language	ACR120S (ACR120.dll)	ACR120U (ACR120U.dll)
VC6	ACR120.h ACR120.lib	ACR120U.h ACR120U.lib
VB6	ACR120.bas	ACR120U.bas
Delphi7	ACR120.pas	ACR120U.pas

II. Explanation and Differences

Serial API	USB API	Remarks
ACR120_Open(Port, Baudrate)	ACR120_Open(Port)	ACR120S parameters: *Port *Baudrate ACR120U parameter: * Port
ACR120_Close (rHandle)	ACR120_Close (hReader)	No change.
ACR120_Reset(rHandle, stationID)	ACR120_Reset(hReader)	ACR120S parameters: * rHandle *station ID ACR120U parameter: * hReader
ACR120_Select(rHandle, stationID, pHaveTag, pTAG, pSN[10]);	ACR120_Select(hReader, pResultTagType, pResultTagLength, pResultSN[10]);	ACR120S parameters: * rHandle *stationID *pHaveTag (Indicate whether the TAG Type Identification is returned) *pTAG (Contains the TAG Type Identification if returned) *pSN(card serial number) ACR120U parameters: *hReader *pResultTagType(Contain the selected Tag Type) *pResultTagLength(Contain the Length of the selected TAG.) *pResultSN[10](Card Serial Number)

ACR120_Login(hHandle, stationID, sector, keyType, storedNo, pKey[6]);	ACR120_Login(hReader, Sector, KeyType, StoredNo, pKey[6]);	ACR1205 parameters: *rHandle , *station ID *sector , *keyType -ACR120_LOGIN_KEYTYPE_AA -ACR120_LOGIN_KEYTYPE_BB -ACR120_LOGIN_KEYTYPE_FF -ACR120_LOGIN_KEYTYPE_STORED_A -ACR120_LOGIN_KEYTYPE_STORED_B *storedNo , *pKey[6] ACR120U parameters: *hReader , *Sector *KeyType -AC_MIFARE_LOGIN_KEYTYPE_A = 0xAA (KeyType2 = 0xAA) -AC_MIFARE_LOGIN_KEYTYPE_B = 0xBB (KeyType2 = 0xBB) - AC_MIFARE_LOGIN_KEYTYPE_DEFAULT_A = 0xAD (KeyType2 = 0xAA) - AC_MIFARE_LOGIN_KEYTYPE_DEFAULT_B = 0xBD (KeyType2 = 0xBB) - AC_MIFARE_LOGIN_KEYTYPE_DEFAULT_F = 0xFD (KeyType2 = 0xFF) - AC_MIFARE_LOGIN_KEYTYPE_STORED_A = 0xAF (KeyType2 = 0xAA) - AC_MIFARE_LOGIN_KEYTYPE_STORED_B = 0xBF (KeyType2 = 0xBB) *StoredNo ,*pKey[6]
ACR120_Read(hHandle, stationID, block, pBlockData[16]);	ACR120_Read(hReader, Block, pBlockData[16]);	ACR1205 parameters: *rHandle *station ID *block *pBlockData[16] ACR120U parameters: *hReader *Block *pBlockData[16]

ACR120_ReadValue(rHandle, stationID, block, pValueData);	ACR120_ReadValue(hReader, Block, pValueData);	ACR120S parameters: *rHandle *stationID *block *pValueData ACR120U parameters: *hReader *Block *pValueData
ACR120_ReadEEPROM(rHandle, stationID, reg, pEEPROMData);	ACR120_ReadEEPROM(hReader, RegNo, pEEPROMData);	ACR120S parameters: *rHandle *stationID *reg (register number) *pEEPROMData ACR120U parameters: *hReader *RegNo (register number) *pEEPROMData
ACR120_ReadLowLevelR egister(rHandle, stationID, reg, pRegData);	ACR120_ReadRC531Reg(hReader, RegNo, pValue);	In ACR120S ACR120_ReadLowLevelRegister was used. Parameters: * rHandle * stationID * reg (register number) * pRegData (Contains the register's value.) In ACR120U ACR120_ReadRC531Reg was used. Parameters: * hReader * RegNo (register number) * pValue (Contains the register's value.)

ACR120_Write(rHandle, stationID, block, pBlockData[16]);	ACR120_Write(hReader, Block, pBlockData[16]);	ACR120S parameters: *rHandle *stationID *block *pBlockData[16] ACR120U parameters: *hReader *Block *pBlockData[16]
ACR120_WriteValue(rHandle, stationID, block, valueData);	ACR120_WriteValue(hReader, Block, ValueData);	ACR120S parameters: *rHandle *stationID *block *valueData ACR120U parameters: *hReader *Block *ValueData
ACR120_WriteEEPROM(rHandle, stationID, reg, eepromData);	ACR120_WriteEEPROM(hReader, RegNo, EEPROMData);	ACR120S parameters: *rHandle *stationID *reg (register number) *eepromData ACR120U parameters: *hReader *RegNo (register number) *EEPROMData

ACR120_WriteLowLevelRegister(rHandle, stationID, reg, registerData);	ACR120_WriteRC531Reg(hReader, RegNo, Value);	In ACR120S ACR120_WriteLowLevelRegister was used. Parameters: * rHandle * stationID * reg (register number) * registerData (Contains the register value to write) In ACR120U ACR120_WriteRC531Reg was used. Parameters: * hReader * RegNo (register number) * Value (Contains the register value to write.)
ACR120_WriteMasterKey(rHandle, stationID, keyNo, pKey[6]);	ACR120_WriteMasterKey(hReader, KeyNo, pKey[6]);	ACR120S parameters: *rHandle *stationID *keyNo *pKey[6] ACR120U parameters: *hReader *KeyNo *pKey[6]
ACR120_Inc(rHandle, stationID, block, value, pNewValue);	ACR120_Inc(hReader, Block, Value, pNewValue);	ACR120S parameters: *rHandle *stationID *Block *value *pNewValue ACR120U parameters: *hReader *Block *Value *pNewValue

ACR120_Dec(rHandle, stationID, block, value, pNewValue);	ACR120_Dec(hReader, Block, Value, pNewValue);	ACR120S parameters: *rHandle *stationID *block *value *pNewValue ACR120U parameters: *hReader *Block *Value *pNewValue
ACR120_Copy(rHandle, stationID, srcBlock, desBlock, pNewValue);	ACR120_Copy(hReader, srcBlock, desBlock, pNewValue);	ACR120S parameters: *rHandle *stationID *srcBlock *desBlock *pNewValue ACR120U parameters: *hReader *srcBlock *desBlock *pNewValue
ACR120_Power(rHandle, stationID, bOn);	ACR120_Power(hReader, State);	ACR120S parameters: *rHandle *stationID *bOn (Boolean | Turn on (TRUE) or off (FALSE).) ACR120U parameters: *hReader *State (INT8 | Turn OFF (0) or ON (1).)
ACR120_ReadUserPort(rHandle, stationID, pUserPortState);	ACR120_ReadUserPort(hReader, pUserPortState);	ACR120S parameters: *rHandle *stationID *pUserPortState Contains the port state (only LSB is used). ACR120U parameters: *hReader *pUserPortState Contain the port state (only Bit 2 & Bit 6 are used).

ACR120_WriteUserPort(rHandle, stationID, userPortState);	ACR120_WriteUserPort(hReader, UserPortState);	ACR120S parameters: *rHandle *stationID *userPortState (Clear the port pin if userPortState = 0. Otherwise it's set.) ACR120U parameters: *hReader *UserPortState Contain the port state to write (only Bit 2 & Bit6 are used).
ACR120_GetIDs(rHandle, pNumID, pStationID);	N/A	Function not applicable for ACR120U
ACR120_ListTag(rHandle, stationID, pNumTagFound, pHaveTag, pTAG, pSN);	ACR120_ListTags(hReader, pNumTagFound, pTagType[4], pTagLength[4], pSN[4][10]);	ACR120S parameters: *rHandle *stationID *pNumTagFound *pHaveTag (Whether the TAG Type Identification is listed.) *pTAG (The list of TAG Type Identification. If pHaveTag is false, this is an array of serial number length of the cards detected. If pHaveTag is true, this is an array of Tag type. The corresponding serial number length could then be determined from the Tag type.) *pSN(The flat array of serial numbers. All serial numbers are concatenated with length of either 4, 7 or 10 numbers. The lengths are indicated in pTag field) ACR120U parameters: *hReader *pNumTagFound * pTagType[4] (Contains the TAG Type) * pTagLength[4] (Contains the length of the serial number) * pSN[4][10] (The flat array of serial numbers. All serial numbers are concatenated with fixed length – 10 bytes.)

ACR120_MultiTagSelect(rHandle, stationID, pSN[10], pHaveTag, pTAG, pResultSN[10]);	ACR120_MultiTagSelect(hReader, TagLength, SN[10], pResultTagType, pResultTagLength, pResultSN);	ACR1205 parameters: *rHandle *stationID *pSN[10] (Contains the serial number of the TAG to be selected) *pHaveTag (//whether the TAG Type Identification of selected tag is returned...) *pTAG (The list of TAG Type Identification. If pHaveTag is false, this is an array of serial number length of the cards detected. If pHaveTag is true, this is an array of Tag type. The corresponding serial number length could then be determined from the Tag type..) *pResultSN(The serial number of selected TAG) ACR120U parameters: *hReader * TagLength (Contains the length of the serial number of the TAG to be selected.) * SN[10] (Contain the serial number of the TAG to be selected) * pResultTagType (Contain the selected Tag Type) * pResultTagLength (Contain the length of the serial number of the selected TAG) * pResultSN (The serial number of the selected TAG.)

ACR120_TxDataTelegram (rHandle, stationID, length, bParity, bOddParity, bCRCGen, bCRCCheck, bCryptoInactive, bitFrame, data, pRecvLen, recvData);	ACR120_TxDataTelegram(hReader, SendDataLength, pSendData pReceivedDataLength, pReceivedData);	ACR120S parameters: * rHandle * stationID * length (The length of user specific data frame) * bParity (TRUE if parity generation is enabled) * bOddParity (TRUE if parity is odd. Otherwise it's even) * bCRCGen (TRUE if CRC generation for transmission is enabled) * bCRCCheck (TRUE if CRC checking for receiving is enabled) * bCryptoInactive (TRUE if Crypto unit is deactivated before transmission start) * bitFrame (number of bits from last byte transmitted) * data (contains the user specific data frame) * pRecvLen (it returns the length of response data received) * recvData (contains the response data receive) ACR120U parameters: * hReader * SendDataLength (the length of data to be sent) * pSendData (the data to be sent) * pReceivedDataLength (the length of received data) * pReceivedData (the received data)
ACR120_RequestDLLVersion(pVersionInfoLen, pVersionInfo);	ACR120_RequestDLLVersion(pVersionInfoLength, pVersionInfo);	ACR120S parameters: * pVersionInfoLen (It returns the length of the DLL Version string.) * pVersionInfo ACR120U parameters: * pVersionInfoLength (It returns the length of the DLL Version string.) * pVersionInfo

N/A	ACR120_Status(hReader, pFirmwareVersion[20], pReaderStatus);	New Function for ACR120U It Returns the firmware version and the Reader status. Parameters: * hReader * pFirmwareVersion[20] (The firmware version will be returned (20 bytes)) * pReaderStatus (The Reader status.) *See ACR120U API Documentation for more details*
N/A	ACR120_DirectSend (hReader, DataLength pData, pResponseDataLength, pResponseData, TimeOut);	New Function for ACR120U To send data to the Mifare Chip directly. Parameters: * hReader * DataLength (The Data Length (maximum 66 bytes)) * pData (The Data to be sent) * pResponseDataLength (The Response Data Length) * pResponseData (The Response Data) * TimeOut (The Time Out for waiting the response data in m-sec) *See ACR120U API Documentation for more details*
N/A	ACR120_DirectReceive(hReader, RespectedDataLength, pReceivedDataLength, pReceivedData, TimeOut);	New Function for ACR120U To receive data from the Mifare Chip directly. Parameters: * hReader * RespectedDataLength (The Respected Data Length to be received (maximum 64 bytes)) * pReceivedDataLength (The Data Length of the received data) * pReceivedData (The Received Data) * TimeOut (The Time Out for waiting the received data in m-sec) *See ACR120U API Documentation for more details*

III. Error Codes

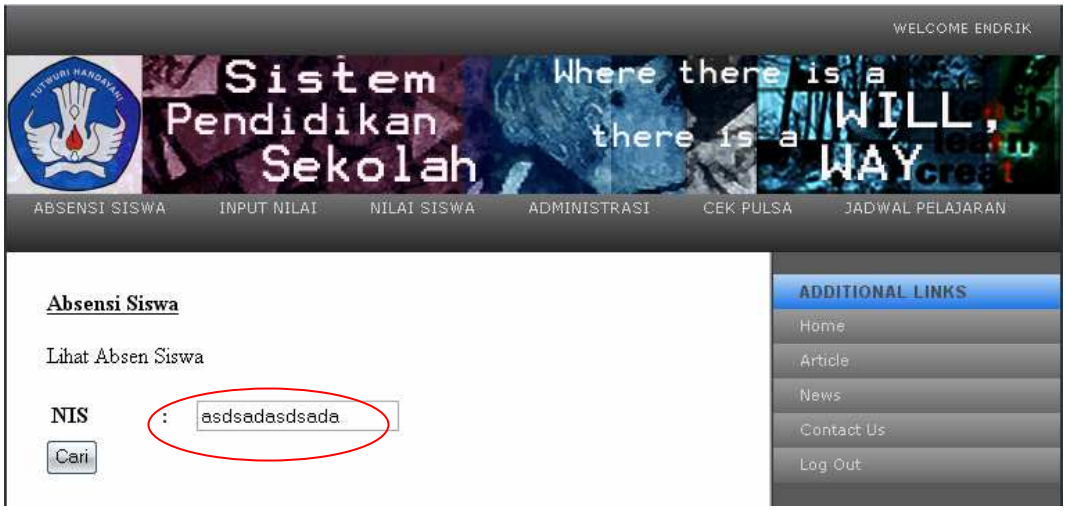
ACR120 (Serial Interface)	
Error Codes	Description
ERR_ACR120_INTERNAL_UNEXPECTED(-1000)	Library internal unexpected error.
ERR_ACR120_PORT_INVALID(-2000)	The port is invalid.
ERR_ACR120_PORT_OCCUPIED(-2010)	The port is occupied by another application.
ERR_ACR120_HANDLE_INVALID(-2020)	The handle is invalid.
ERR_ACR120_INCORRECT_PARAM(-2030)	Incorrect Parameter.
ERR_ACR120_READER_NO_TAG(-3000)	No TAG in reachable range / selected.
ERR_ACR120_READER_READ_FAIL_AFTER_OP(-3010)	Read fail after operation.
ERR_ACR120_READER_NO_VALUE_BLOCK(-3020)	Block doesn't contain value.
ERR_ACR120_READER_OP_FAILURE(-3030)	Operation failed.
ERR_ACR120_READER_UNKNOWN(-3040)	Reader unknown error.
ERR_ACR120_READER_LOGIN_INVALID_STORED_KEY_FORMAT(-4010)	Invalid stored key format in login process.
ERR_ACR120_READER_WRITE_READ_AFTER_WRITE_ERROR(-4020)	Reader can't read after write operation.
ERR_ACR120_READER_DEC_FAILURE_EMPTY(-4030)	Decrement failure (empty).

ACR120 (USB Interface)	
Error Codes	**Description**
ERR_INTERNAL_UNEXPECTED(-1000)	Library internal unexpected error. *#Handled by the DLL*
ERR_PORT_INVALID(-2000)	The port is invalid. *#Handled by the DLL*
ERR_PORT_OCCUPIED(-2010)	The port is occupied by another application. *#Handled by the DLL*
ERR_HANDLE_INVALID(-2020)	The handle is invalid. *#Handled by the DLL*
ERR_INCORRECT_PARAM(-2030)	Incorrect Parameter. *#Handled by the DLL.*
ERR_READER_NO_TAG(-3000, or 0xF448)	No TAG in reachable range / selected. *#Corresponding to the << Response Status 'N' >>.*
ERR_READER_OP_FAILURE(-3030, or 0xF42A)	Operation failed. *#Corresponding to the << Response Status 'F' >>.*
ERR_READER_UNKNOWN(-3040, or 0xF420)	Reader unknown error. *#Corresponding to the << Response Status 'C', 'D', 'X' & '?' >>.*
ERR_READER_LOGIN_INVALID_STORED_KEY_FORMAT(-4010, or 0xF056)	Invalid stored key format in login process. *#Handled by the DLL.*
ERR_READER_LOGIN_FAIL(-4011, or 0xF055)	Login failed. *#Corresponding to the << Response Status 'I' >>.*
ERR_READER_OP_AUTH_FAIL(-4012, or 0xF054)	The operation or access is not authorized. *#Corresponding to the << Response Status 'I' >>.*
ERR_READER_VALUE_DEC_EMPTY(-4030, or 0xF042)	Decrement failure (empty). *#Corresponding to the << Response Status 'E' >>.*
ERR_READER_VALUE_INC_OVERFLOW(-4031, or 0xF041)	Increment Overflow. *#Corresponding to the << Response Status 'E' >>.*
ERR_READER_VALUE_OP_FAILURE (-4032, 0xF040)	Value Operations failure. E.g. Value Increment *#Corresponding to the << Response Status 'I' >>.*

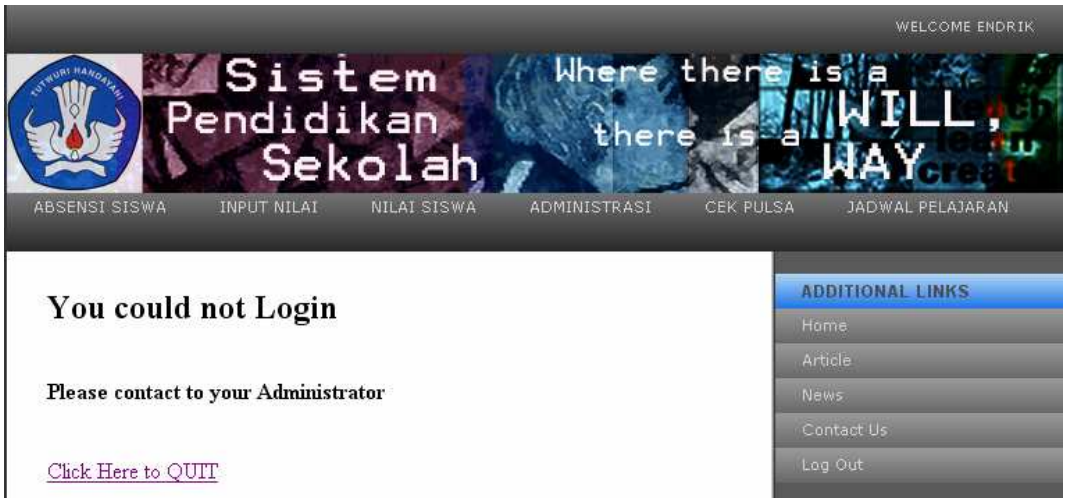
ERR_READER_VALUE_INVALID_BLOCK(-4033, 0xF03F)	Block doesn't contain value. #Corresponding to the << Response Status 'F' >>.
ERR_READER_VALUE_ACCESS_FAILURE (-4034, 0xF03E)	Value Access failure. #Corresponding to the << Response Status 'U' >>.

LAMPIRAN D
TAMPILAN KESALAHAN WEB ADMIN

Fungsi Melihat Absensi Siswa

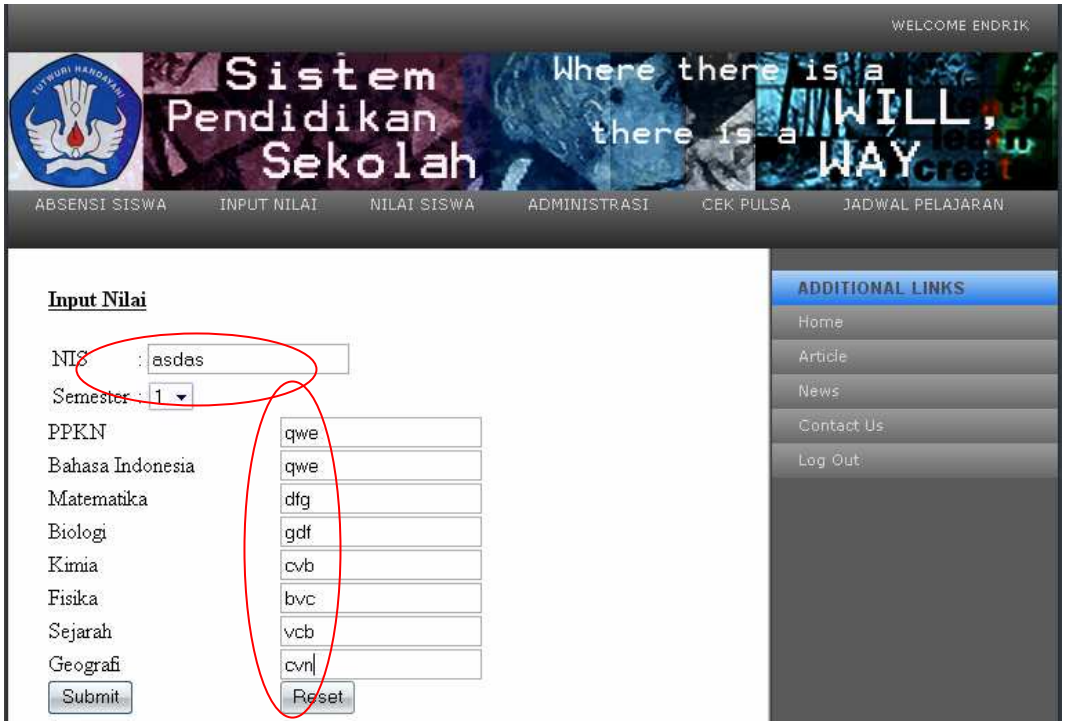


Diberikan input tidak sesuai pada kolom NIS

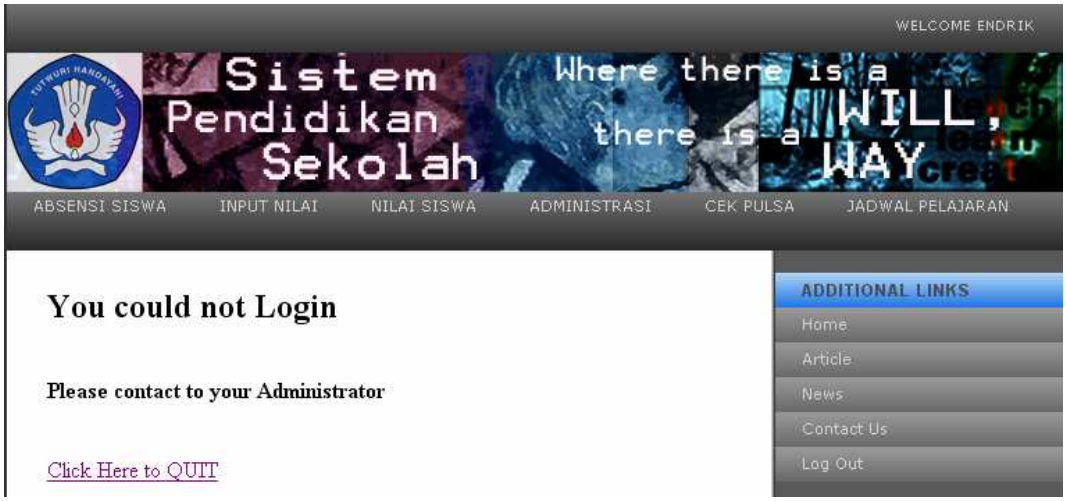


Hasil tampilan dari input yang tidak sesuai

Fungsi Menginputkan Nilai Siswa

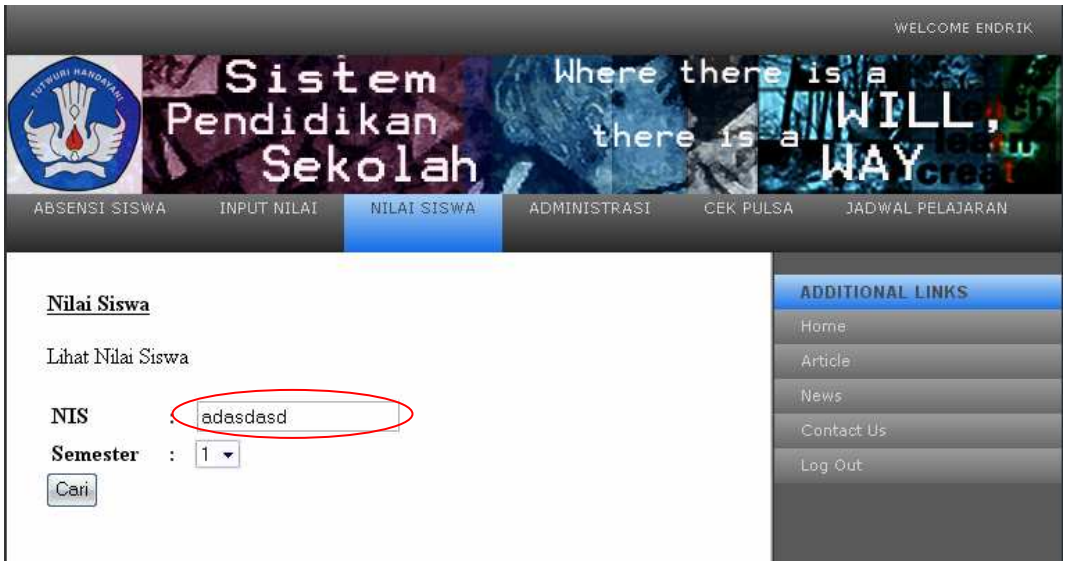


Diberikan input tidak sesuai pada semua kolom



Hasil tampilan dari input yang tidak sesuai

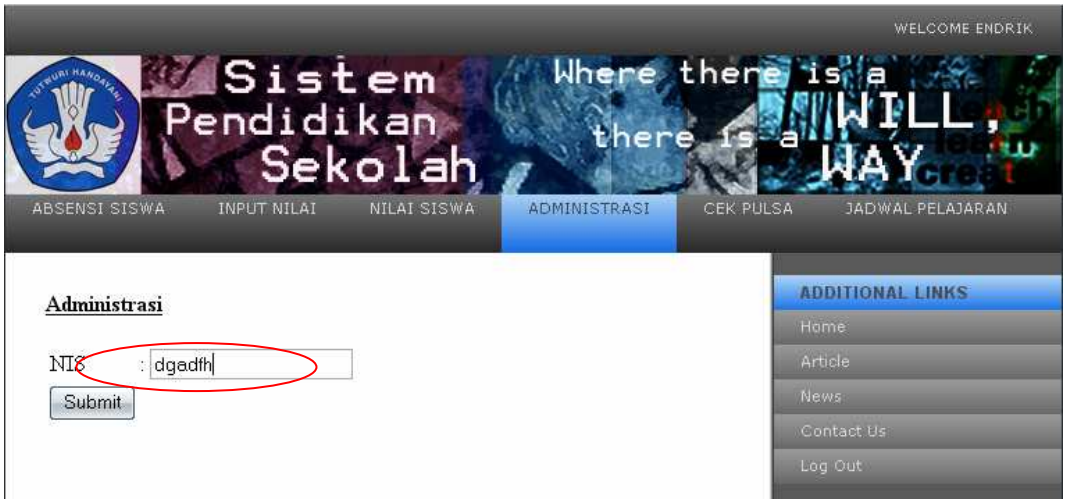
Fungsi Melihat Nilai Siswa



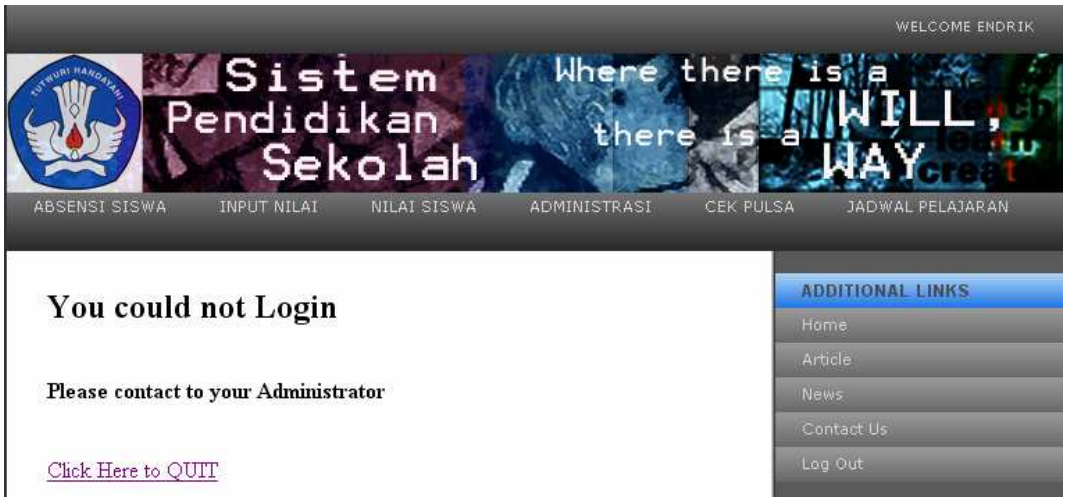
Diberikan input tidak sesuai pada kolom NIS

Hasil tampilan dari input yang tidak sesuai

Fungsi Melihat Administrasi Siswa

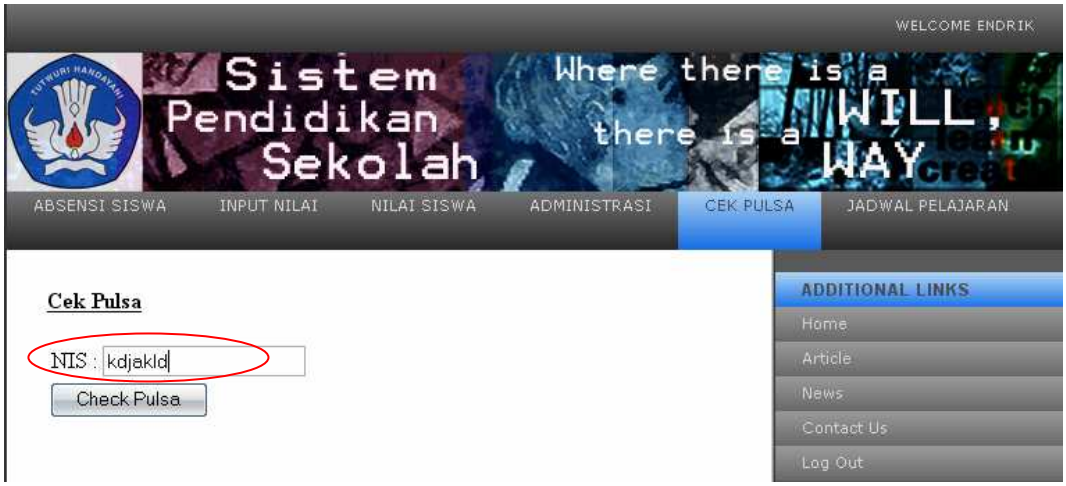


Diberikan input yang tidak sesuai pada kolom NIS

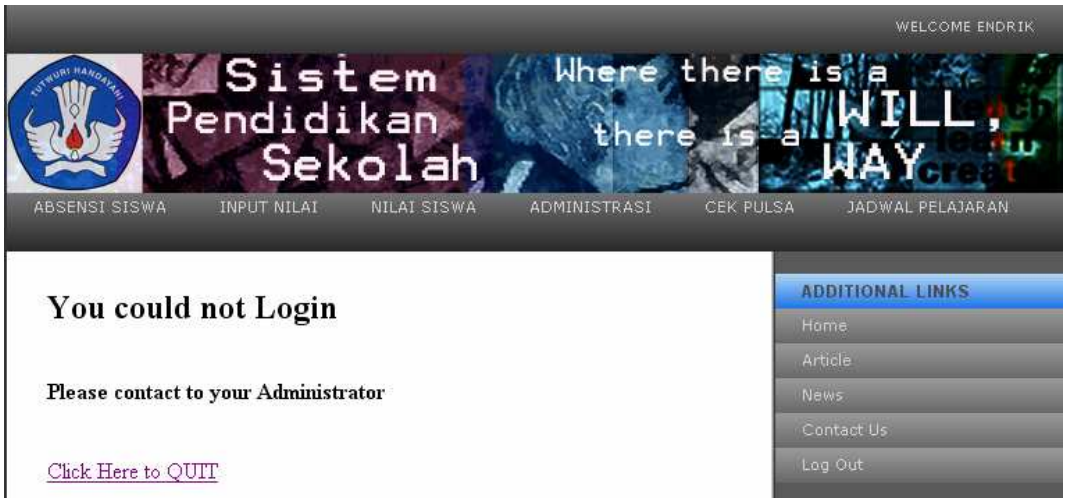


Hasil tampilan dari input yang tidak sesuai

Fungsi Melihat Saldo Pulsa Siswa



Diberikan input yang tidak sesuai pada kolom siswa



Hasil tampilan dari input yang tidak sesuai

Fungsi Melihat Jadwal Pelajaran Siswa

Tampilan Web Admin untuk melihat jadwal pelajaran