LAMPIRAN A

KODE PROGRAM

- **Form Login**

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using MySql.Data.MySqlClient;

namespace RFIDRW
{
    public partial class Login : Form
    {
        public Login()
        {
            InitializeComponent();
        }

        private void btnLogin_Click(object sender, EventArgs e)
        {
            //MessageBox.Show(MyRoutine.DBRead());

            string query = "select * from user_id where username like '"
+ txtUser.Text.Trim() + "' and password like '" +
txtPassword.Text.Trim() + "';";
            String str =
@"server=localhost;database=nfcpayment;userid=root;password=;";
            MySqlConnection con = null;
            MySqlDataReader reader = null;

            try
            {
                con = new MySqlConnection(str);
                con.Open();
                String cmdText = query;
                MySqlCommand cmd = new MySqlCommand(cmdText, con);
                reader = cmd.ExecuteReader();
                /*The Read() method points to the next record It return
false if there are no more records else returns true.*/
```

```
                if (reader.Read())
                {
                    frmMain form = new frmMain();
                    form.Show();
                    this.Hide();
                }
                else lblError.Text = "User ID atau Password anda
salah..";
            }
            catch (MySqlException err)
            {
                //Console.WriteLine("Error: " + err.ToString());
                lblError.Text = "Error: " + err.ToString();
            }

            finally
            {
                if (reader != null)
                {
                    reader.Close();
                }
                if (con != null)
                {
                    con.Close(); //close the connection
                }
            }
        }
    }
}
```

Form Supervisor

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace RFIDRW
{
    public partial class frmMain : Form
    {
        public long retCode;
        int hContext, ReaderCount, Protocol, hCard;
        string sReaderList = string.Empty;
```

```
        byte[] sReaderGroup;
        string errString;

        ModWinsCard.SCARD_IO_REQUEST ioRequest;
        public int SendLen, RecvLen;
        public byte[] SendBuff = new byte[262];
        public byte[] RecvBuff = new byte[262];

        int UidLen;
        public byte[] Uid;
        int cardtype;
        string uiStr = string.Empty;

        public frmMain()
        {
            InitializeComponent();
            establish();
        }

        /// <summary>
        /// Set the controls to its default state.
        /// </summary>
        void InitMenu()
        {
            cmboReader.Items.Clear();
            lstLog.Items.Clear();

            btnEstablishContext.Enabled = true;
            btnQuit.Enabled = true;
            btnReaderlst.Enabled = false;
            btnConnect.Enabled = false;
            btnSetRetryTime.Enabled = false;
            btnDetectTag.Enabled = false;
            btnAuthentication.Enabled = false;
            btnReadBlock.Enabled = false;
            btnWriteBlock.Enabled = false;
            btnReset.Enabled = false;
        }

        private void establish() {
            sReaderList = string.Empty;
            ReaderCount = 255;
            cmboReader.Items.Clear();

            byte[] retData;

            string readerStr = string.Empty;
            string tmpStr = string.Empty;

            //Establish Context Part
            //Establish Context. Pls see SCardEstablishContext
definition from ModWinsCard for more info.
```

```
            retCode =
ModWinsCard.SCardEstablishContext(ModWinsCard.SCARD_SCOPE_USER, 0, 0,
ref hContext);
            lblError.Text = retCode.ToString();

            if (retCode != ModWinsCard.SCARD_S_SUCCESS)
            {
                FuncAddToLog(ModWinsCard.GetScardErrMsg(retCode));
                return;
            }

            FuncAddToLog("Establish context success.");
            btnReaderlst.Enabled = true;

            //Lis Reader Part
            retCode = ModWinsCard.SCardListReaders(hContext, null, null,
ref ReaderCount);

            if (retCode != ModWinsCard.SCARD_S_SUCCESS)
            {
                FuncAddToLog(ModWinsCard.GetScardErrMsg(retCode));
                return;
            }

            retData = new byte[ReaderCount];

            //Get the list of reader present again but this time add
sReaderGroup, retData as 2rd & 3rd parameter respectively.
            retCode = ModWinsCard.SCardListReaders(hContext,
sReaderGroup, retData, ref ReaderCount);

            if (retCode != ModWinsCard.SCARD_S_SUCCESS)
            {
                FuncAddToLog(ModWinsCard.GetScardErrMsg(retCode));
                return;
            }

            //Convert retData(Hexadecimal) value to String
            readerStr =
System.Text.ASCIIEncoding.ASCII.GetString(retData);

            //Add the readers from readerStr to combobox control
            ModWinsCard.LoadListControl(cmboReader, readerStr);

            //Select 1st item from the combobox if any
            if (cmboReader.Items.Count > 0)
                cmboReader.SelectedIndex = 0;

            //Enable Connect button
            if (cmboReader.Items.Count > 0)
                btnConnect.Enabled = true;
```

```
            //Connect Part
            retCode = ModWinsCard.SCardConnect(hContext,
cmboReader.SelectedItem.ToString(), ModWinsCard.SCARD_SHARE_EXCLUSIVE,
                                    ModWinsCard.SCARD_PROTOCOL_T0 |
ModWinsCard.SCARD_PROTOCOL_T1, ref hCard, ref Protocol);

            //Check if it connects successfully
            if (retCode != ModWinsCard.SCARD_S_SUCCESS)
            {
                FuncAddToLog(ModWinsCard.GetScardErrMsg(retCode));
                return;
            }
            else
            {
                FuncAddToLog("Connected to : " +
cmboReader.SelectedItem.ToString());
            }

            SendLen = 5;
            RecvLen = 10;

            RecvBuff = new byte[10];

            //Get Firmware Version Command
            SendBuff = new byte[5];
            SendBuff[0] = 0xFF;
            SendBuff[1] = 0x00;
            SendBuff[2] = 0x48;
            SendBuff[3] = 0x00;
            SendBuff[4] = 0x00;

            ioRequest.dwProtocol = Protocol;
            ioRequest.cbPciLength = 8;

            //The SCardTransmit function sends a service request to the
smart card and expects to receive data back from the card.
            //ms-
help://MS.VSCC.v80/MS.MSDN.v80/MS.WIN32COM.v10.en/secauthn/security/scar
dtransmit.htm
            retCode = ModWinsCard.SCardTransmit(hCard, ref ioRequest,
ref SendBuff[0], SendLen, ref ioRequest, ref RecvBuff[0], ref  RecvLen);

            if (retCode != ModWinsCard.SCARD_S_SUCCESS)
            {
                FuncAddToLog(ModWinsCard.GetScardErrMsg(retCode));
                return;
            }

            //Convert RecvBuff(Hexadecimal) to String
            tmpStr =
System.Text.ASCIIEncoding.ASCII.GetString(RecvBuff);
            FuncAddToLog("Firmware Version : " + tmpStr);
```

```
            //grpFunctions.Enabled = true;
            btnSetRetryTime.Enabled = true;
            btnReset.Enabled = true;

            //SetRetryTime Part
            SendLen = 11;
            RecvLen = 2;
            SendBuff = new byte[SendLen];
            RecvBuff = new byte[RecvLen];

            //Set Retry time Command
            SendBuff[0] = 0xFF;
            SendBuff[1] = 0x00;
            SendBuff[2] = 0x00;
            SendBuff[3] = 0x00;
            SendBuff[4] = 0x06;
            SendBuff[5] = 0xD4;
            SendBuff[6] = 0x32;
            SendBuff[7] = 0x05;
            SendBuff[8] = 0x00;
            SendBuff[9] = 0x00;
            SendBuff[10] = 0x00;

            retCode = Transmit();
            if (retCode != ModWinsCard.SCARD_S_SUCCESS)
            {
                FuncAddToLog(ModWinsCard.GetScardErrMsg(retCode));
                return;
            }

            btnDetectTag.Enabled = true;
            FuncAddToLog("Set Retry to One Success: " + retCode);
        }

        private void lstEstablishContext_Click(object sender, EventArgs
e)
        {
            /*sReaderList = string.Empty;
            ReaderCount = 255;

            //Establish Context. Pls see SCardEstablishContext
definition from ModWinsCard for more info.
            retCode =
ModWinsCard.SCardEstablishContext(ModWinsCard.SCARD_SCOPE_USER, 0, 0,
ref hContext);
            if (retCode != ModWinsCard.SCARD_S_SUCCESS)
            {
                FuncAddToLog(ModWinsCard.GetScardErrMsg(retCode));
                return;
            }
```

```
            FuncAddToLog("Establish context success.");
            btnReaderlst.Enabled = true;*/

        }

        /// <summary>
        /// Add item to log(listbox). Use this if your message is in a string form.
        /// </summary>
        public void FuncAddToLog(string str)
        {
            lstLog.Items.Add(str);
            lstLog.SelectedIndex = lstLog.Items.Count - 1;
        }

        /// <summary>
        /// Add item to log(listbox). Use this if your message is in a hexadecimal form.
        /// </summary>
        public void FuncAddToLog(string prefixStr, byte[] buff, string postfixStr, int buffLen)
        {
            string tmpStr = string.Empty;

            //Convert each byte from buff to its string value.
            for (int i = 0; i < buffLen; i++)
                tmpStr += string.Format("{0:X2}", buff[i]) + " ";

            //Add item to log (listBox)
            lstLog.Items.Add(prefixStr + tmpStr + postfixStr);

            //Select the last item from the listbox.
            lstLog.SelectedIndex = lstLog.Items.Count - 1;
        }

        /// <summary>
        /// Get Response from Direct Transmit
        /// Issue Get Response to fetch the actual response of the card
        /// </summary>
        public bool GetResponse()
        {
            //Issue Get Response for Tag found
            SendLen = 5;
            RecvLen = RecvBuff[1];
            RecvBuff = new byte[RecvLen];
            SendBuff = new byte[SendLen];

            //Get Response Command
            SendBuff[0] = 0xFF;
            SendBuff[1] = 0xC0;
            SendBuff[2] = 0x00;
            SendBuff[3] = 0x00;
```

```
            SendBuff[4] = (byte)RecvLen;

            retCode = Transmit();
            if (retCode != ModWinsCard.SCARD_S_SUCCESS)
                return false;

            return true;

        }

        /// <summary>
        /// Transmit any command from SendBuff variable and store recieved data to RecvBuff
        /// </summary>
        public long Transmit()
        {
            string tmpStr = string.Empty;

            ioRequest.dwProtocol = Protocol;
            ioRequest.cbPciLength = 8;

            RecvLen = 262;

            FuncAddToLog("<<", SendBuff, "", SendLen);

            //The SCardTransmit function sends a service request to the smart card and expects to receive data back from the card.
            //ms-help://MS.VSCC.v80/MS.MSDN.v80/MS.WIN32COM.v10.en/secauthn/security/scardtransmit.htm
            retCode = ModWinsCard.SCardTransmit(hCard, ref ioRequest, ref SendBuff[0], SendLen, ref ioRequest, ref RecvBuff[0], ref RecvLen);

            if (retCode != ModWinsCard.SCARD_S_SUCCESS)
            {
                FuncAddToLog(ModWinsCard.GetScardErrMsg(retCode));
                return retCode;
            }

            FuncAddToLog(">>", RecvBuff, "", RecvLen);

            return retCode;
        }

        /// <summary>
        /// Reinitialize RecvBuff and  Sendbuff variable
        /// </summary>
        public void ClearBuffers()
        {
            RecvBuff = new byte[262];
            SendBuff = new byte[262];
        }
```

```
•
•           private void btnQuit_Click(object sender, EventArgs e)
•           {
•               //Release Context. Pls see SCardReleaseContext definition
    from ModWinsCard for more info.
•               retCode = ModWinsCard.SCardReleaseContext(hContext);
•               this.Close();
•               Application.Exit();
•           }
•
•           private void btnReaderlst_Click(object sender, EventArgs e)
•           {
•               /*cmboReader.Items.Clear();
•               byte[] retData;
•               string readerStr = string.Empty;
•
•               //The SCardListReaders function provides the list of readers
    within a set of named reader groups, eliminating duplicates.
•               //ms-
    help://MS.VSCC.v80/MS.MSDN.v80/MS.WIN32COM.v10.en/secauthn/security/scar
    dlistreaders.htm
•               //Get the number of reader found and store it to ReaderCount
•               retCode = ModWinsCard.SCardListReaders(hContext, null, null,
    ref ReaderCount);
•
•               if (retCode != ModWinsCard.SCARD_S_SUCCESS)
•               {
•                   FuncAddToLog(ModWinsCard.GetScardErrMsg(retCode));
•                   return;
•               }
•
•               retData = new byte[ReaderCount];
•
•               //Get the list of reader present again but this time add
    sReaderGroup, retData as 2rd & 3rd parameter respectively.
•               retCode = ModWinsCard.SCardListReaders(hContext,
    sReaderGroup, retData, ref ReaderCount);
•
•               if (retCode != ModWinsCard.SCARD_S_SUCCESS)
•               {
•                   FuncAddToLog(ModWinsCard.GetScardErrMsg(retCode));
•                   return;
•               }
•
•               //Convert retData(Hexadecimal) value to String
•               readerStr =
    System.Text.ASCIIEncoding.ASCII.GetString(retData);
•
•               //Add the readers from readerStr to combobox control
•               ModWinsCard.LoadListControl(cmboReader, readerStr);
•
•               //Select 1st item from the combobox if any
```

```
            if (cmboReader.Items.Count > 0)
                cmboReader.SelectedIndex = 0;

            //Enable Connect button
            if (cmboReader.Items.Count > 0)
                btnConnect.Enabled = true;*/
        }

        private void btnConnect_Click(object sender, EventArgs e)
        {
            /*string tmpStr = string.Empty;

            //establishes a connection (using a specific resource
manager context)
            //between the calling application and a smart card contained
by a specific reader.
            //If no card exists in the specified reader, an error is
returned.
            //For more info:
            //ms-
help://MS.VSCC.v80/MS.MSDN.v80/MS.WIN32COM.v10.en/secauthn/security/scar
dconnect.htm if you have MSDN 2005 installed
            retCode = ModWinsCard.SCardConnect(hContext,
cmboReader.SelectedItem.ToString(), ModWinsCard.SCARD_SHARE_EXCLUSIVE,
                                    ModWinsCard.SCARD_PROTOCOL_T0 |
ModWinsCard.SCARD_PROTOCOL_T1, ref hCard, ref Protocol);

            //Check if it connects successfully
            if (retCode != ModWinsCard.SCARD_S_SUCCESS)
            {
                FuncAddToLog(ModWinsCard.GetScardErrMsg(retCode));
                return;
            }
            else
            {
                FuncAddToLog("Connected to : " +
cmboReader.SelectedItem.ToString());
            }

            SendLen = 5;
            RecvLen = 10;

            RecvBuff = new byte[10];

            //Get Firmware Version Command
            SendBuff = new byte[5];
            SendBuff[0] = 0xFF;
            SendBuff[1] = 0x00;
            SendBuff[2] = 0x48;
            SendBuff[3] = 0x00;
            SendBuff[4] = 0x00;

```

```
            ioRequest.dwProtocol = Protocol;
            ioRequest.cbPciLength = 8;

            //The SCardTransmit function sends a service request to the smart card and expects to receive data back from the card.
            //ms-help://MS.VSCC.v80/MS.MSDN.v80/MS.WIN32COM.v10.en/secauthn/security/scardtransmit.htm
            retCode = ModWinsCard.SCardTransmit(hCard, ref ioRequest, ref SendBuff[0], SendLen, ref ioRequest, ref RecvBuff[0], ref  RecvLen);

            if (retCode != ModWinsCard.SCARD_S_SUCCESS)
            {
                FuncAddToLog(ModWinsCard.GetScardErrMsg(retCode));
                return;
            }

            //Convert RecvBuff(Hexadecimal) to String
            tmpStr = System.Text.ASCIIEncoding.ASCII.GetString(RecvBuff);
            FuncAddToLog("Firmware Version : " + tmpStr);

            //grpFunctions.Enabled = true;
            btnSetRetryTime.Enabled = true;
            btnReset.Enabled = true;*/
        }

        private void btnSetRetryTime_Click(object sender, EventArgs e)
        {
            /*SendLen = 11;
            RecvLen = 2;

            SendBuff = new byte[SendLen];
            RecvBuff = new byte[RecvLen];

            //Set Retry time Command
            SendBuff[0] = 0xFF;
            SendBuff[1] = 0x00;
            SendBuff[2] = 0x00;
            SendBuff[3] = 0x00;
            SendBuff[4] = 0x06;
            SendBuff[5] = 0xD4;
            SendBuff[6] = 0x32;
            SendBuff[7] = 0x05;
            SendBuff[8] = 0x00;
            SendBuff[9] = 0x00;
            SendBuff[10] = 0x00;

            retCode = Transmit();
            if (retCode != ModWinsCard.SCARD_S_SUCCESS)
            {
                FuncAddToLog(ModWinsCard.GetScardErrMsg(retCode));
```

```
                return;
            }

            btnDetectTag.Enabled = true;
            FuncAddToLog("Set Retry to One Success: " + retCode);*/
        }

        private void btnDetectTag_Click(object sender, EventArgs e)
        {
            if (!CheckCard())
            {
                FuncAddToLog("No valid card within detection range.");
                return;
            }

            btnAuthentication.Enabled = true;
            FuncAddToLog(errString);

            //authenticate automatically
            FrmAuthenticate formAuthenticate = new FrmAuthenticate();
            DialogResult dr = formAuthenticate.ShowDialog(this);

            //Check if Authentication is successful
            if (dr == DialogResult.OK)
            {
                btnReadBlock.Enabled = true;
                btnWriteBlock.Enabled = true;
            }
        }

        /// <summary>
        /// Check if recognized card is within detection range.
        /// </summary>
        bool CheckCard()
        {
            if (!isMifareType())
                cardtype = 255;

            //Set the string to be displayed in the StatusStrip
            switch (cardtype)
            {
                case 1: errString = "Mifare UltraLight UID:" + uiStr;
break;
                case 2: errString = "Mifare 1K UID:" + uiStr; break;
                case 3: errString = "Mifare Mini UID:" + uiStr; break;
                case 4: errString = "Mifare 4K UID:" + uiStr; break;
                case 5: errString = "Mifare DESFire UID:" + uiStr;
break;
                case 6: errString = "ISO14443-4 Type A UID:" + uiStr;
break;
                case 7: errString = "Gemplus MPCOS UID:" + uiStr; break;
                case 8: errString = "Topaz UID:" + uiStr; break;
```

```
                case 9: errString = "FeliCa, 212 kbps UID:" + uiStr;
break;
                case 10: errString = "FeliCa, 424 kbps UID:" + uiStr;
break;
                case 11: errString = "ISO14443-4 Type B, 106 kbps UID:"
+ uiStr; break;
                default: return false;
            }
            return true;
        }


        /// <summary>
        /// Check if Mifare card type is within detection range.
        /// Returns True if Mifare or False if not.
        /// </summary>
        bool isMifareType()
        {
            ClearBuffers();
            byte[] uidValue;

            //Mifare Polling Command
            SendBuff[0] = 0xFF;
            SendBuff[1] = 0x00;
            SendBuff[2] = 0x00;
            SendBuff[3] = 0x00;
            SendBuff[4] = 0x04;
            SendBuff[5] = 0xD4;
            SendBuff[6] = 0x4A;
            SendBuff[7] = 0x01;
            SendBuff[8] = 0x00;

            SendLen = 9;
            RecvLen = 2;

            if (Transmit() != ModWinsCard.SCARD_S_SUCCESS)
            {
                FuncAddToLog(ModWinsCard.GetScardErrMsg(retCode));
                return false;
            }

            if (RecvBuff[0] == 0x61 && RecvBuff[1] == 0x05)
                return false;

            //Get Response Command
            SendBuff[0] = 0xFF;
            SendBuff[1] = 0xC0;
            SendBuff[2] = 0x00;
            SendBuff[3] = 0x00;
            SendBuff[4] = RecvBuff[1];

            SendLen = 5;
```

```
            RecvLen = RecvBuff[1];

            retCode = Transmit();

            if (retCode != ModWinsCard.SCARD_S_SUCCESS)
            {
                FuncAddToLog("Mifare Type Error: " +
ModWinsCard.GetScardErrMsg(retCode));
                return false;
            }

            switch (RecvBuff[6])
            {
                case 0: cardtype = 1; break;    // Mifare UltraLight
                case 8: cardtype = 2; break;    // Mifare 1K
                case 9: cardtype = 3; break;    // Mifare Mini
                case 24: cardtype = 4; break;   // Mifare 4K
                case 32: cardtype = 5; break;   // Mifare DESFire
                case 40: cardtype = 6; break;   // ISO14443-4 Type A
                case 152: cardtype = 7; break;  //Gemplus MPCOS
                default: cardtype = 255; break; //Unknown card
            }

            FuncAddToLog("Detected Tag Success: " + retCode);

            UidLen = RecvBuff[7];
            uidValue = new byte[UidLen];

            Array.Copy(RecvBuff, 8, uidValue, 0, UidLen);

            Uid = uidValue;
            uiStr = MyRoutine.FuncByteAsString(uidValue);

            return true;

        }

        private void btnAuthentication_Click(object sender, EventArgs e)
        {
            //Show Authentication Form
            FrmAuthenticate formAuthenticate = new FrmAuthenticate();
            DialogResult dr = formAuthenticate.ShowDialog(this);

            //Check if Authentication is successful
            if (dr == DialogResult.OK)
            {
                btnReadBlock.Enabled = true;
                btnWriteBlock.Enabled = true;
            }

        }

```

```
        private void btnReset_Click(object sender, EventArgs e)
        {
            //Relase Context. Pls. see the summary/description
            //from the definition int the ModWinsCard class for more
info.
            retCode = ModWinsCard.SCardReleaseContext(hContext);
            InitMenu();
            establish();
        }

        private void btnWriteBlock_Click(object sender, EventArgs e)
        {
            //Show the form for writting value to block.
            FrmWriteBlock formWrite = new FrmWriteBlock();
            formWrite.ShowDialog(this);
        }

        private void btnReadBlock_Click(object sender, EventArgs e)
        {
            //Show the form for reading from block
            FrmReadBlock formRead = new FrmReadBlock();
            formRead.ShowDialog(this);
        }

        private void frmMain_Load(object sender, EventArgs e)
        {

        }

        private void custBtt_Click(object sender, EventArgs e)
        {
            DB dbForm = new DB();
            dbForm.ShowDialog(this);

        }

    }
}
```

Form Registrasi

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
```

```
using System.Globalization;
using MySql.Data.MySqlClient;

namespace RFIDRW
{
    public partial class FrmWriteBlock : Form
    {
        frmMain ownerForm;
        byte BlockNum;
        int flag;

        Person _temporaryPerson;

        public Person TemporaryPerson
        {
            get { return _temporaryPerson; }
        }


        public FrmWriteBlock()
        {
            InitializeComponent();

            this.Text = "Add New Person";
            _temporaryPerson = new Person();
        }

        public FrmWriteBlock(Person person)
        {
            InitializeComponent();
            this.Text = "Edit Person Data";
            _temporaryPerson = person;
            txtNama.Text = TemporaryPerson.Name;
            txtSaldo.Text = TemporaryPerson.Saldo.ToString();
            txtAddress.Text = TemporaryPerson.Alamat;
        }

        public FrmWriteBlock(frmMain owner, int flag)
        {
            InitializeComponent();
            ownerForm = owner;
            this.flag = flag;
        }

        private void FrmWriteBlock_Load(object sender, EventArgs e)
        {
            //Initialize the ownerForm to get handle to main form so it
can access the public property/method of
            //main form like the SendBuff variable, FuncAddToLog methodm
etc.
            if (flag != 1) ownerForm = (frmMain)this.Owner;

```

```
            //AddressbookDBDataContext db = new
AddressbookDBDataContext();
            //var query = (from p in db.Persons
            //             where p.PersonID ==
db.Persons.Max(x=>x.PersonID)
            //             select p).Single();

            //var y = query.PersonID+1;
            //db.SubmitChanges();

            string query = "select Max(ID) from customer";
            String str =
@"server=localhost;database=nfcpayment;userid=root;password=;";
            MySqlConnection con = null;
            MySqlDataReader reader = null;
            int inc = 0;

            try
            {
                con = new MySqlConnection(str);
                con.Open();
                String cmdText = query;
                MySqlCommand cmd = new MySqlCommand(cmdText, con);
                reader = cmd.ExecuteReader();
                /*The Read() method points to the next record It return
false if there are no more records else returns true.*/

                if (reader.Read())
                {
                    string temp = reader.GetValue(0).ToString();
                    inc = int.Parse(temp) + 1;
                }
            }
            catch (MySqlException err)
            {
                //Console.WriteLine("Error: " + err.ToString());
                MessageBox.Show("Error: " + err.ToString());
            }

            finally
            {
                if (reader != null)
                {
                    reader.Close();
                }
                if (con != null)
                {
                    con.Close(); //close the connection
                }
            }

            txtId.Text = inc.ToString();
```

```
        }

        private void btnCancel_Click(object sender, EventArgs e)
        {
            ownerForm.FuncAddToLog("Write Block Cancelled....");
            this.Close();
        }

        private void btnInput_Click(object sender, EventArgs e)
        {
            byte[] data;
            string temp;

            //Write NAME to block 01
            //
            //Check if Block Number specified is in valid format
            if (!BlockNumCheck("1"))
            {
                return;
            }

            //Get the data (ID) to be written
            temp = txtId.Text.Trim();
            if (temp.Length < 16)
            {
                //add trailing 'whitespace' if the length of string is below 16
                temp = temp.PadRight(16, ' ');
            }

            data = new byte[16];
            data = ASCIIEncoding.ASCII.GetBytes(temp);
            if (data == null)
            {
                MessageBox.Show("No Valid Data, please try again");
                txtNama.Focus();
                return;
            }

            //Check if it trying to write on to trailer block
            if (!BlockNumTrailerWriteCheck())
            {
                return;
            }

            write(data);


            //Write SALDO to Block 02
            //
            //Check the Block Number first
            if (!BlockNumCheck("2"))
```

```
            {
                return;
            }

            //Get the data (saldo) to be written
            temp = txtSaldo.Text.Trim();
            if (temp.Length < 16)
            {
                //add trailing '*' if the length of string is below 16
                temp = temp.PadRight(16, '*');
            }

            data = new byte[16];
            data = ASCIIEncoding.ASCII.GetBytes(temp);
            if (data == null)
            {
                MessageBox.Show("The amount to be written are not a
valid value.");
                txtSaldo.Focus();
                return;
            }

            //Check if it trying to write on to trailer block
            if (!BlockNumTrailerWriteCheck())
            {
                return;
            }

            write(data);

            /*//Write SALDO to Block 04
            //
            //Check the Block Number first
            if (!BlockNumCheck("4"))
            {
                return;
            }

            //Get the data (serial) to be written
            temp = txtPass.Text.Trim();
            if (temp.Length < 16)
            {
                //add trailing '#' if the length of string is below 16
                temp = temp.PadRight(16, '#');
            }

            data = new byte[16];
            data = ASCIIEncoding.ASCII.GetBytes(temp);

            //Check if it trying to write on to trailer block
            if (!BlockNumTrailerWriteCheck())
            {
```

```
                return;
            }

            write(data);*/

        }

        /// <summary>
        /// Check if Block Number specified is in valid format
        /// </summary>
        /// <param name="block">block number to be checked</param>
        /// <returns>true for valid blocknum, false for invalid blocknum</returns>
        public bool BlockNumCheck(string block)
        {
            //Check and then store the converted block number to BlockNum
            if (!byte.TryParse(block, NumberStyles.HexNumber, null, out BlockNum))
            {
                MessageBox.Show("Writing error - Block Number was incorrect, contact your administrator");
                return false;
            }
            return true;
        }

        /// <summary>
        /// Check if it trying to write on to trailer block
        /// </summary>
        /// <param name=""></param>
        /// <returns></returns>
        public bool BlockNumTrailerWriteCheck()
        {
            DialogResult dr;

            if (BlockNum < 0x80)
            {
                if ((BlockNum % 4) == 3)
                {
                    dr = MessageBox.Show("This is for Trapping Accidental write to Trailer Block, if you're not sure on how to setup Access Bit..\r\nCancel the operation, otherwise Click OK.",
                                         "Critical Block Chosen to write", MessageBoxButtons.OKCancel, MessageBoxIcon.Warning);
                    if (dr == DialogResult.Cancel)
                        return false;
                }
            }
            else
            {
                if (((BlockNum - 128) % 4) == 3)
```

```
                {
                    dr = MessageBox.Show("This is for Trapping
Accidental.",
                                         "Critical Block Chosen to
write", MessageBoxButtons.OKCancel, MessageBoxIcon.Warning);
                    if (dr == DialogResult.Cancel)
                        return false;
                }
            }

            return true;
        }

        /// <summary>
        /// Preapare Write command and write it!
        /// </summary>
        /// <param name="data">The data to be written</param>
        /// <returns></returns>
        public void write(byte[] data)
        {
            ownerForm.SendLen = 26;
            ownerForm.RecvLen = 2;

            ownerForm.SendBuff = new byte[ownerForm.SendLen];
            ownerForm.RecvBuff = new byte[ownerForm.RecvLen];

            //Update/Write Command
            ownerForm.SendBuff[0] = 0xFF;
            ownerForm.SendBuff[1] = 0x00;
            ownerForm.SendBuff[2] = 0x00;
            ownerForm.SendBuff[3] = 0x00;
            ownerForm.SendBuff[4] = 0x15;
            ownerForm.SendBuff[5] = 0xD4;
            ownerForm.SendBuff[6] = 0x40;
            ownerForm.SendBuff[7] = 0x01;
            ownerForm.SendBuff[8] = 0xA0;
            ownerForm.SendBuff[9] = BlockNum;

            Array.Copy(data, 0, ownerForm.SendBuff, 10, 16);

            //Transmit the Update or Write Command
            ownerForm.retCode = ownerForm.Transmit();
            if (ownerForm.retCode != ModWinsCard.SCARD_S_SUCCESS)
                return;


            //Issue Get Response to fetch the actual response of the
card
            ownerForm.GetResponse();

            if (ownerForm.RecvBuff[2] != 0x00)
            {
```

```
ownerForm.FuncAddToLog(MyRoutine.GetContactLessCardErrMsg(ownerForm.Recv
Buff[2]));
            }
            else
                ownerForm.FuncAddToLog("Write to Block " +
BlockNum.ToString() + " Success: " + ownerForm.retCode);
        }

        private void btnInput2_Click(object sender, EventArgs e)
        {
            //AddressbookDBDataContext db = new
AddressbookDBDataContext();

            ////Write to DB
            //_temporaryPerson.Name = txtNama.Text;
            //_temporaryPerson.Saldo = int.Parse(txtSaldo.Text);
            //_temporaryPerson.Alamat = txtAddress.Text ;

            //db.Persons.InsertOnSubmit(_temporaryPerson);
            //db.SubmitChanges();

            string query = "INSERT INTO customer (NAMA,ALAMAT,SALDO)
VALUES('"+txtNama.Text+"','"+txtAddress.Text+"',"+int.Parse(txtSaldo.Tex
t)+")";
            String strcon =
@"server=localhost;database=nfcpayment;userid=root;password=;";
            MySqlConnection con = null;
            try
            {
                con = new MySqlConnection(strcon);
                con.Open();
                String cmdText = query;
                MySqlCommand cmd = new MySqlCommand(cmdText, con);
                //cmd.Prepare();
                //cmd.Parameters.AddWithValue("@name", "your value
here");
                cmd.ExecuteNonQuery();
            }
            catch (MySqlException err)
            {
                MessageBox.Show("Error: " + err.ToString());
            }

            finally
            {
                if (con != null)
                {
                    con.Close();
                }
            }
        }
```

```
        private void txtId_TextChanged(object sender, EventArgs e)
        {

        }

    }
}
```

- **Form Transaksi**

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using MySql.Data.MySqlClient;

namespace RFIDRW
{
    public partial class FrmReadBlock : Form
    {
        frmMain ownerForm;
        Person _temporaryPerson;
        int flag;
        string billing;

        public Person TemporaryPerson
        {
            get { return _temporaryPerson; }
        }

        public FrmReadBlock()
        {
            InitializeComponent();
        }

        private void btnCancel_Click(object sender, EventArgs e)
        {
            ownerForm.FuncAddToLog("Read Block Cancelled....");
            this.Close();
        }

        private void FrmReadBlock_Load(object sender, EventArgs e)
        {
```

```
            //Initialize the ownerForm to get handle to main form so it can
access the public property/method of
            //main form like the SendBuff variable, FuncAddToLog methodm etc.
            ownerForm = (frmMain)this.Owner;
            MessageBox.Show("Letakkan Kartu pada Reader");
        }

        private void btnBaca_Click(object sender, EventArgs e)
        {
            lblID.Text = read("1"); //Read ID from block number 01
            lblSaldo2.Text = read("2").TrimEnd('*'); //Read Saldo from block
number 02

            string query = "select NAMA from customer where ID = " +
lblID.Text.Trim() + ";";
            String str =
@"server=localhost;database=nfcpayment;userid=root;password=;";
            MySqlConnection con = null;
            MySqlDataReader reader = null;

            try
            {
                con = new MySqlConnection(str);
                con.Open();
                String cmdText = query;
                MySqlCommand cmd = new MySqlCommand(cmdText, con);
                reader = cmd.ExecuteReader();
                /*The Read() method points to the next record It return false
if there are no more records else returns true.*/

                if (reader.Read())
                {
                    lblNama2.Text = reader.GetValue(0).ToString();
                }
            }
            catch (MySqlException err)
            {
                //Console.WriteLine("Error: " + err.ToString());
                MessageBox.Show("Error: " + err.ToString());
            }

            finally
            {
                if (reader != null)
                {
                    reader.Close();
                }
                if (con != null)
                {
                    con.Close(); //close the connection
                }
            }

            btnBuy.Enabled = true;
            btnTopup.Enabled = true;
```

```
}

private void btnBuy_Click(object sender, EventArgs e)
{
    flag = 0;
    itemBox.Visible = true;
    txtAmount.ReadOnly = true;
}

private void btnTopup_Click(object sender, EventArgs e)
{
    flag = 1;
    itemBox.Visible = false;
    txtAmount.ReadOnly = false;
}

/// <summary>
/// Write the data (Saldo) to Card
/// </summary>
/// <param name="curAmount">Current Amount/Saldo</param>
/// <returns></returns>
public void writeAmount(int curAmount) {
    //Prepare Data to Write
    byte[] data;
    string temp;

    FrmWriteBlock formWrite = new FrmWriteBlock(this.ownerForm, 1);
    //Check the Block Number first
    if (!formWrite.BlockNumCheck("2"))
    {
        return;
    }

    //Get the data (saldo) to be written
    temp = curAmount.ToString();
    if (temp.Length < 16)
    {
        //add trailing '*' if the length of string is below 16
        temp = temp.PadRight(16, '*');
    }

    data = new byte[16];
    data = ASCIIEncoding.ASCII.GetBytes(temp);
    if (data == null)
    {
        MessageBox.Show("Calculation Error, Please try again");
        return;
    }

    //Check if it trying to write on to trailer block
    if (!formWrite.BlockNumTrailerWriteCheck())
    {
        return;
    }
```

```
            formWrite.write(data);
        }

        /// <summary>
        /// Returns the Data Read from Block Number as string
        /// </summary>
        /// <param name="block">Block number to be read</param>
        /// <returns></returns>
        public string read(string block)
        {
            byte BlockNum;
            byte[] dataRead;

            //Check if block number supplied is valid
            if (!byte.TryParse(block.Trim(),
System.Globalization.NumberStyles.HexNumber, null, out BlockNum))
            {
                MessageBox.Show("Block Number invalid, please Contact your
administrator.");
                //error: please specify hexadecimal value for Block number
                return null;
            }

            ownerForm.SendLen = 10;
            ownerForm.RecvLen = 2;

            ownerForm.SendBuff = new byte[ownerForm.SendLen];
            ownerForm.RecvBuff = new byte[ownerForm.RecvLen];

            //Read Command
            ownerForm.SendBuff[0] = 0xFF;
            ownerForm.SendBuff[1] = 0x00;
            ownerForm.SendBuff[2] = 0x00;
            ownerForm.SendBuff[3] = 0x00;
            ownerForm.SendBuff[4] = 0x05;
            ownerForm.SendBuff[5] = 0xD4;
            ownerForm.SendBuff[6] = 0x40;
            ownerForm.SendBuff[7] = 0x01;
            ownerForm.SendBuff[8] = 0x30;
            ownerForm.SendBuff[9] = BlockNum;

            //Transmit Read Command
            ownerForm.retCode = ownerForm.Transmit();
            if (ownerForm.retCode != ModWinsCard.SCARD_S_SUCCESS)
                return null;

            //Issue Get Response
            ownerForm.GetResponse();

            //Check if the 3rd element of RecvBuff array is equals to 0x00
            //If not, Error occurs.
            if (ownerForm.RecvBuff[2] != 0x00)
            {
```

```
ownerForm.FuncAddToLog(MyRoutine.GetContactLessCardErrMsg(ownerForm.RecvBuff[2]
));
                return null;
            }
            else
            {
                dataRead = new byte[16];
                Array.Copy(ownerForm.RecvBuff, 3, dataRead, 0, 16);

                ownerForm.FuncAddToLog("Data readed at block " +
BlockNum.ToString() + " ASCII mode: " +
ASCIIEncoding.ASCII.GetString(dataRead));
                return ASCIIEncoding.ASCII.GetString(dataRead);
            }
        }

        private void btnExe_Click(object sender, EventArgs e)
        {
            //Calculate Amount
            int curAmount = int.Parse(lblSaldo2.Text);
            int buyAmount = int.Parse(txtAmount.Text);

            if (flag == 0)
            {
                if (buyAmount > curAmount)
                {
                    string str = "Saldo Anda Kurang " + Math.Abs(curAmount -
buyAmount);
                    MessageBox.Show(str);
                    txtAmount.Focus();
                    return;
                }
                else curAmount -= buyAmount;
            }
            else
            {
                curAmount += buyAmount;
                writeAmount(curAmount);
            }
            writeAmount(curAmount);

            //Update DB
            //AddressbookDBDataContext db = new AddressbookDBDataContext();
            //var query = (from p in db.Persons
            //             where p.Name == lblNama2.Text
            //             select p).Single();

            //query.Saldo = curAmount;
            //db.SubmitChanges();

            DateTime dt = DateTime.Now;
            string datetimemysql = dt.ToString("yyyy-MM-dd HH:mm:ss");
            string query = "INSERT INTO transactions
(ID,NAMA,SALDO_AWAL,SALDO_AKHIR,WAKTU) VALUES(" + int.Parse(lblID.Text)+ ",'" +
```

```
lblNama2.Text + "'," + int.Parse(lblSaldo2.Text) + ","+ curAmount +",'"+
datetimemysql+"');";
            query += "UPDATE customer SET SALDO = "+curAmount+" WHERE ID =
"+int.Parse(lblID.Text)+";";

            String strcon =
@"server=localhost;database=nfcpayment;userid=root;password=;";
            MySqlConnection con = null;
            try
            {
                con = new MySqlConnection(strcon);
                con.Open();
                String cmdText = query;
                MySqlCommand cmd = new MySqlCommand(cmdText, con);
                //cmd.Prepare();
                //cmd.Parameters.AddWithValue("@name", "your value here");
                cmd.ExecuteNonQuery();
                billing = "Nama: " + lblNama2.Text + "\nSaldo Awal: " +
lblSaldo2.Text + "\nSaldo Akhir: " + curAmount.ToString() + "\nWaktu: " +
datetimemysql;
            }
            catch (MySqlException err)
            {
                MessageBox.Show("Error: " + err.ToString());
            }

            finally
            {
                if (con != null)
                {
                    con.Close();
                }
            }

            lblSaldo2.Text = curAmount.ToString();
            btnBill.Enabled = true;
        }

        private void itemBox_SelectedIndexChanged(object sender, EventArgs e)
        {
            if (itemBox.SelectedIndex == 0)
            {
                txtAmount.Text = "12000";
            }
            else
            {
                txtAmount.Text = "1000000";
            }
        }

        private void btnBill_Click(object sender, EventArgs e)
        {
            string fileName = @"C:\Bill.txt";

            try
```

```
            {
                // Check if file already exists. If yes, delete it.
                //if (File.Exists(fileName))
                //{
                //    File.Delete(fileName);
                //}

                // Create a new file
                using (StreamWriter sw = File.CreateText(fileName))
                {
                    sw.WriteLine("New Bill created: {0}",
DateTime.Now.ToString());
                    sw.WriteLine("Author: NFC");
                    sw.WriteLine(billing);
                }

                // Open the stream and read it back.
                //using (StreamReader sr = File.OpenText(fileName))
                //{
                //    string s = "";
                //    while ((s = sr.ReadLine()) != null)
                //    {
                //        Console.WriteLine(s);
                //    }
                //}
            }
            catch (Exception Ex)
            {
                Console.WriteLine(Ex.ToString());
            }

            btnBill.Enabled = false;
        }

    }
}
```

LAMPIRAN B

STANDAR ISO/IEC 18092 dan ISO 14443 A dan B

ISO/IEC 14443

From Wikipedia, the free encyclopedia

Jump to: navigation, search

ISO/IEC 14443 *Identification cards -- Contactless integrated circuit cards -- Proximity cards* is an international standard that defines proximity cards used for identification, and the transmission protocols for communicating with it.[1][2][3][4]

Contents

Standard

The standard was developed by the *Working Group 8* of *Subcommittee 17* in *ISO/IEC Joint Technical Committee 1*.

Parts

- ISO/IEC 14443-1:2008 Part 1: Physical characteristics[1]
- ISO/IEC 14443-2:2010 Part 2: Radio frequency power and signal interface[2]
- ISO/IEC 14443-3:2011 Part 3: Initialization and anticollision[3]
- ISO/IEC 14443-4:2008 Part 4: Transmission protocol[4]

Types

Cards may be Type A and Type B, both of which communicate via radio at 13.56 MHz. The main differences between these types concern modulation methods, coding schemes (Part 2) and protocol initialization procedures (Part 3). Both Type A and Type B cards use the same transmission protocol described in Part 4. The transmission protocol specifies data block exchange and related mechanisms:

1. data block chaining
2. waiting time extension
3. multi-activation

ISO/IEC 14443 uses following terms for components:

- PCD: proximity coupling device (the card reader)
- PICC: proximity integrated circuit card

Physical size

Part 1 of the standard specifies that the card shall be compliant with ISO/IEC 7810 or ISO/IEC 15457-1, or "an object of any other dimension".[1]

Notable implementations

- MIFARE cards
- Calypso (electronic ticketing system)
- iCLASS cards and tags by HID Global[*citation needed*]
- Biometric passports
- EMV payment cards (PayPass, payWave, ExpressPay)
- German identity card
- Near Field Communication is based on in part, and is compatible with, ISO/IEC 14443

ISO/IEC 18092

1 Scope

This International Standard defines communication modes for Near Field Communication Interface and Protocol (NFCIP-1) using inductive coupled devices operating at the centre frequency of 13,56 MHz for interconnection of computer peripherals. It also defines both the Active and the Passive communication modes of Near Field Communication Interface and Protocol (NFCIP-1) to realize a communication network using Near Field Communication devices for networked products and also for consumer equipment. This International Standard specifies, in particular, modulation schemes, codings, transfer speeds, and frame format of the RF interface, as well as initialisation schemes and conditions required for data collision control during initialisation. Furthermore, this International Standard defines a transport protocol including protocol activation and data exchange methods.

Information interchange between systems also requires, at a minimum, agreement between the interchange parties upon the interchange codes and the data structure.

2 Conformance

A system implementing the Active and the Passive communication mode shall be in conformance with this International Standard if it meets all the mandatory requirements specified herein.

It may also implement the NFC-SEC Option as specified in ISO/IEC 13157-1.

3 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ITU-T V.41:1988, *Code-independent error-control system*

ISO/IEC 13157-1:2010, *Information technology — Telecommunications and information exchange between systems — NFC Security — Part 1: NFC-SEC NFCIP-1 security services and protocol*

ISO/IEC 14443-2:2010, *Identification cards — Contactless integrated circuit cards — Proximity cards — Part 2: Radio frequency power and signal interface*

ISO/IEC 14443-3:2011, *Identification cards — Contactless integrated circuit cards — Proximity cards — Part 3: Initialization and anticollision*

ISO/IEC 14443-4:2008, *Identification cards — Contactless integrated circuit cards — Proximity cards — Part 4: Transmission protocol*