SM130
SM132-USB

Software Development Kit - SDK
C#.NET Mifare Class Library
USER MANUAL

Compatible with
.NET COMPACT FRAMEWORK 2.0

EVALUATION / DEVELOPMENT KIT

For a fast starting and a product just in days, we recommend you to buy one of our development/evaluation kits. SDK – Software Development Kit is included freely in Deluxe versions of the kits or it can be purchased separately.

For Online Shopping, you can visit:
http://www.sonmicro.com/shop/shop3.php

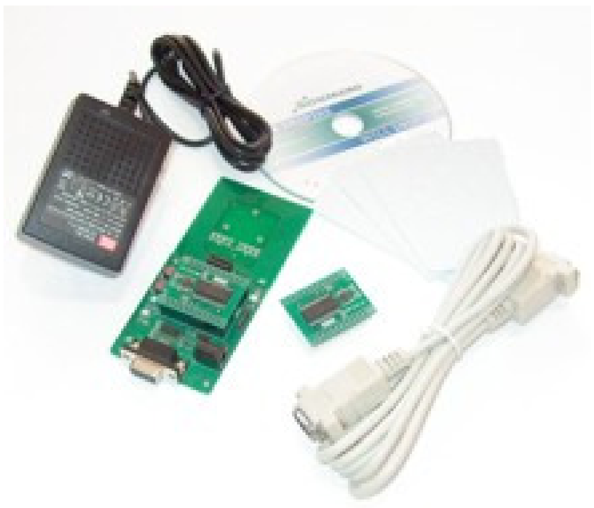

Figure 1 – SM1013 Evaluation Kit – Deluxe

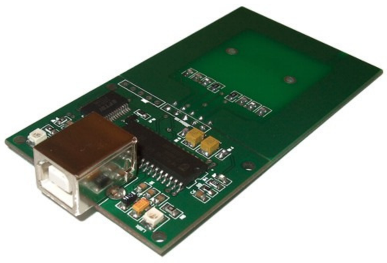

Figure 2 – SM132-USB – USB Mifare Reader

1. INTRODUCTION

This document explains usage of SonMicro Mifare class library written in C#(C Sharp) for .NET applications. Classes were written in .NET Compact Framework 2.0 environment so that the C# library can also be used in mobile applications that are running Windows CE, Windows Mobile operating systems.

For developer environments other than .NET (Such as Delphi, Visual Basic etc), SonMicro serves ActiveX Library that can be used in wide range of IDE. Please refer SM13X_SDK document for ActiveX library which can be found at our web site.

Users can quickly add Mifare functions to existing software or create new software for Mifare applications easily with the provided library.

Mifare library provides high level APIs to communicate with the supported devices (See Section 1.1 for the supported devices) and useful functions. Users never need to know about the communication protocol occurring between the device and the computer, Mifare library will handle with that. Communication channel is based on Com Port of the Computer/Mobile Device. Mifare library can also reliably be used with a virtual com port or the USB-Serial converters.

It is strongly recommended for users who are strange to Mifare, first read about Mifare basics. Brief information for Mifare and its application can be found in User Manual file at our web page. http://www.sonmicro.com/1356/d1356.php

Operating systems other than Microsoft Windows is not supported currently. For non-windows operating systems communication between the module and the PC/Controller can be implemented at low level by managing serial port with the protocol explained in device datasheet.

1.1 Supported Devices

Mifare SDK supports the following devices:

- SM130 Mifare Module
- SM132-USB Mifare Reader (Integrated with Serial-to-USB interface and PCB antenna)

Supported Development kits:

- SM1013 Eval. kit for SM130
- SM1013USB Eval. Kit for SM132-USB

Please note that SM132-USB module is connected over USB interface but the control of these modules is still done in "classic serial port" manner with the created virtual com port. Driver for SM132USB can be downloaded at our web site.

2. PROCEDURES AND FUNCTIONS

It is assumed that developer has learn the basics of Mifare and the SonMicro Mifare device. Please visit http://www.sonmicro.com/1356/d1356.php to see useful documents and software to get a fast understanding for Mifare and mifare applications. For SonMicro Mifare Readers details please reference the relevant product's User Manual documents.

All commands sent from host (PC) have immediate response from the Mifare device. However, there are two response type that Mifare device can send to host while host not requesting. These are;

1- When module resets, or power-on it sends Firmware Version information to the host.
2- When "Seek For Tag" command is used, module will send Tag data as soon as a Tag enters in to the field.

For both of these situations, host needs to check if there is any unread data in serial port input buffer. In the example C# project, this is done by using a **timer**. **Timer1** checks every 500ms if there is unread data at input buffer of the serial port. User can develop or modify the code found in Timer Event according to his/her needs and take necessary actions according to one of two response type.

2.1 OpenPort

This function will open selected port. Port needs to be opened before communicating with the Mifare device.

`bool OpenPort(string PortName,int BaudRate)`	
Arguments	**PortName:** Port number to be opened **BaudRate:** Baud rate setting of the port.
Returns	True : If port is open False : If port could not be opened

Example

```
sm_mifare_lib.mifare sm132 = new sm_mifare_lib.mifare();

  if (sm132.OpenPort(comboBox1.Text, 19200))
            {
                textBox1.Text = "Port is initialized and opened";
                timer1.Enabled = true;
            }
            else
                 textBox1.Text = "Port could not be opened";
```

2.2 ClosePort

This function will close the serial port.

`void ClosePort()`	
Arguments	None
Returns	none

Example

```
sm_mifare_lib.mifare sm132 = new sm_mifare_lib.mifare();

      sm132.ClosePort();
```

2.3 CMD_ResetDevice

This function sends reset command to the mifare device. Device will reset itself and sends the firmware version information to the PC in a second.

`bool CMD_ResetDevice(out string Firmware)`	
Arguments	**Firmware:** Firmware version information will be passed to this variable
Returns	True : Reset operation is successful False : Reset operation is not successful

Example:

```
sm_mifare_lib.mifare sm132 = new sm_mifare_lib.mifare();
string Firmware;

      if (sm132.CMD_ResetDevice(out Firmware))
            textBox1.Text = Firmware;
      else
            textBox1.Text = "Error. Expecting Firmware Version";
```

2.4 CMD_SelectTag

This function sends Select Tag command to the mifare device. Serial number and type of the tag will return if the tag is in the RF field at that time.

Application Hint

Before performing any read/write/value operations on a Mifare Tag, the tag should be **selected** first. For a tag to be selected it should be in the RF Field. For Mifare 1K and Mifare 4K tags, authentication process should follow the select operation in order to read/write/value operations on the tag's blocks.

Select operation can be done in two ways with CMD_SelectTag and CMD_SeekForTag. CMD_SelectTag will return the result immediately. If there is a tag in the field the details of the tag will return with the serial number. If there is no tag in the field, relevant error code will be sent.

In the customer application, CMD_SeekForTag can be used to auto-detect Mifare Tags or, CMD_SelectTag command should be send periodically. Response to CMD_SelectTag can be processed immediately in the calling function. If CMD_SeekForTag is used, then the process is done under Timer Event

```
bool CMD_SelectTag(out byte TagType,out byte[] TagSerial,out byte ReturnCode)
```

Arguments	**TagType:** Indicates the type of the tag (i.e Mifare 1K, Mifare 4K, Mifare UL) **TagSerial:** Includes 4 BYTE Serial number of the Mifare Tag at TagSerial[0],TagSerial[1],TagSerial[2],TagSerial[3] **ReturnCode:** If the select operation is not successful, CMD_Select tag will return "false". When CMD_SelectTag returns false, the error code will be passed to this variable. <u>Possible Return Code:</u> 0x00 - Communication Error 0x4E 'N' - No Tag present. 0x55 'U' - Access failed due to RF Field is OFF
Returns	True : Select operation is successful False : Select operation is not successful

Example:

```
sm_mifare_lib.mifare sm132 = new sm_mifare_lib.mifare();

byte TagType;
byte[] TagSerial = new byte[4];
byte ReturnCode = 0;
byte i;


 if (sm132.CMD_SelectTag(out TagType, out TagSerial, out ReturnCode))
 {
    textBox1.Text = "Tag Type:" + TagType.ToString() + " ";
    textBox1.Text += " Tag Serial: ";


     for (i = 0; i < 4; i++)
     textBox1.Text += TagSerial[i].ToString("X2");

  }
  else //Select not successful
  {
    if (ReturnCode == 0x55) //'N'
      textBox1.Text = "RF Field is off. Error Code:" + ReturnCode.ToString("X2");
    else if (ReturnCode==0)
      textBox1.Text = "Communication Error. Error Code:" + ReturnCode.ToString("X2");
    else
      textBox1.Text = "Unknown Error. Error Code:" + ReturnCode.ToString("X2");
  }
```

2.5 CMD_Authenticate

This function sends Authentication command to the mifare device. This command needs to be sent after Mifare Tag is **selected** with CMD_SelectTag or CMD_SeekForTag command. CMD_Authenticate will return "true" if Authentication process is successful, otherwise returns "false" and the error code can be read from ReturnCode parameter.

Application Hint

Authentication is necessary to access Mifare Tag blocks. To access Mifare blocks, tag needs to be selected first. Then authentication should follow the select operation. After authentication is successful, host can read and write Mifare Tag blocks of the authenticated sector. Note that every sector can have different type of authentication conditions and keys.

There are three types of Authentication source.

1 – Mifare Default. If this option is selected then the Authentication will be done with KeyA and the key 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF. Mifare tags coming from the factory can be accessed by this way.

2 – Provided Key. If this option is selected, then user can determine the Key Type. (KeyA or KeyB) then determine the 6 byte key.

3 – E2prom. SM13X have internal memory to keep 16 sets of Key. Each set(memory block) have KeyA and KeyB keys. User can select one of these keys and determine the key type to be used. Once key is burned to SM13X internal memory they cannot be read again, so it is useful to store special keys without revealing them.

```
bool CMD_Authenticate(byte AuthSource, byte[] Key, byte BlockNo,out
byte ReturnCode)
```

Arguments	**AuthSource:** Determines Authentication source and type 0xFF Mifare Default (KeyA, FF FF FF FF FF FF) 0xAA Provided Key - KeyA (Key is in Key[] array below) 0xBB Provided Key - KeyB (Key is in Key[] array below) 0x10 E2PROM - KeyA E2prom Internal Block 0 0x11 E2PROM - KeyA E2prom Internal Block 1 .. 0x1F E2PROM - KeyA E2prom Internal Block 15 0x20 E2PROM - KeyB E2prom Internal Block 0 0x21 E2PROM - KeyB E2prom Internal Block 1 .. 0x2F E2PROM - KeyB E2prom Internal Block 15 **Key:** 6 Byte Key if AuthSource is 0xAA or 0xBB **BlockNo:** Block number of Mifare Tag to be authenticated. **ReturnCode:** If the Authentication operation is not successful, function will return "false" and the error code will be passed to this variable. Possible Return Code: 0x00 - Communication Error 0x55 'U' - Authentication Failed 0x45 'E' - Invalid Key Format or Authentication Failed 0x4E 'N' - No Tag or Authentication Failed
Returns	True : Authentication is successful False : Authentication is not successful. See ReturnCode

Example:

```
sm_mifare_lib.mifare sm132 = new sm_mifare_lib.mifare();

byte AuthSource = 0xFF;
byte BlockNo;
byte ReturnCode;
byte[] Key = new byte[6];

BlockNo = 14; // 14th Block will be authenticated (Sector 3)

//The following Key will be used only if Provided Key option is selected.

Key[0] = 0xCC;
Key[1] = 0xCC;
Key[2] = 0xCC;
Key[3] = 0xCC;
Key[4] = 0xCC;
Key[5] = 0xCC;

AuthSource = (byte)sm_mifare_lib.ASource.KeyTypeA;
// AuthSource = (byte)sm_mifare_lib.Asource.KeyTypeB;


//The KeyA, FF FF FF FF FF FF will be used if Mifare Default option is selected
AuthSource = (byte)sm_mifare_lib.ASource.KeyMifareDefault; // 0xFF


//Example to Use E2PROM internal keys->
AuthSource = sm_mifare_lib.mifare.E2promKeyA[cmb_E2promBlockNo.SelectedIndex];
//AuthSource = sm_mifare_lib.mifare.E2promKeyA[0]; //Use E2prom block0 KeyA
//AuthSource = sm_mifare_lib.mifare.E2promKeyA[15]; //Use E2prom block15 KeyA
//AuthSource = sm_mifare_lib.mifare.E2promKeyB[15]; //Use E2prom block15 KeyB



//Authenticate
 if (sm132.CMD_Authenticate(AuthSource, Key, BlockNo, out ReturnCode))
    textBox1.Text = " Authentication Success ";
 else
 {
   if (ReturnCode == 0x4E) //'N'
   textBox1.Text = "No Tag or Login failed. Error Code:" + ReturnCode.ToString("X2");
   else if (ReturnCode == 0x55)
   textBox1.Text = "Login failed. Error Code:" + ReturnCode.ToString("X2");
   else if (ReturnCode == 0x45)
   textBox1.Text = "Invalid Key Format. Error Code:" + ReturnCode.ToString("X2");
   else if (ReturnCode == 0)
   textBox1.Text = "Communication Error. Error Code:" + ReturnCode.ToString("X2");
   else
   textBox1.Text = "Unknown Error. Error Code:" + ReturnCode.ToString("X2");
 }
```

2.6 CMD_Halt

This function executes a Halt command on Mifare Tag. If the Mifare Tag is selected then, after a halt command, all Select, Authentication commands needs to be repeated for new operations.

bool CMD_Halt(out byte ReturnCode)	
Arguments	**ReturnCode:** If the Halt operation is not successful, CMD_Halt will return "false" and the error code can be investigate by this parameter. Possible Return Code: 0x00 – Communication Error 0x55 'U' – Halt failed due to RF Field is OFF
Returns	True : Halt command is successful False : Halt command is not successful. See ReturnCode

Example:

```
sm_mifare_lib.mifare sm132 = new sm_mifare_lib.mifare();

  byte ReturnCode = 0;

if (sm132.CMD_Halt(out ReturnCode))
textBox1.Text = "Halt successfull";
else //Halt not successful
{

 if (ReturnCode == 0x55) //'U'
 textBox1.Text = "Halt not successfull. Error Code:" + ReturnCode.ToString("X2");
 else
 textBox1.Text = "Communication Error. Error Code:" + ReturnCode.ToString("X2");

 }
```

2.7 CMD_ReadBlock

This function sends Read Block Command to the mifare device. To Read a block successfully, prior Select Tag and Authentication commands needs to be successful. Read Block command will read 16 bytes from the specified block. Read will be successful only if the given block is authenticated successfully.

Application Hint

Before performing any read/write/value operations on a Mifare Tag, the tag should be **selected** first. Then the given block needs to be authenticated.

Mifare Classic 1K and 4K blocks consist of 16 bytes data. There are 64 blocks (0-63) for Mifare 1K, and 256 blocks (0-255) for Mifare 4K tags.

```
bool CMD_ReadBlock(byte BlockNo, out byte[] BlockData, out byte
ReturnCode)
```

Arguments	**BlockNo:** Block number to be read. **BlockData:** Includes 16 BYTE data of Mifare Block BlockData[0]….BlockData[15] **ReturnCode:** If the Read Block command is not successful then the error code will be passed to this parameter. Possible Return Code: 0x00 – Communication Error 0x4E ‘N’ – No Tag present. 0x46 ‘F’ – Read Failed
Returns	True : Read Block operation is successful False : Read Block operation is not successful

***Example*:**

```
sm_mifare_lib.mifare sm132 = new sm_mifare_lib.mifare();

byte BlockNo = 0;
byte[] BlockData = new byte[16];
byte ReturnCode = 0;

 BlockNo = byte.Parse(cmb_RWBlockNo.Text);


 if (sm132.CMD_ReadBlock(BlockNo,out BlockData,out ReturnCode))
   {
       textBox1.Text = "Read Successfull";
       textBox_S.Text = textBox1.Text;

       t_B1.Text = BlockData[0].ToString("X2");
       t_B2.Text = BlockData[1].ToString("X2");
       t_B3.Text = BlockData[2].ToString("X2");
       t_B4.Text = BlockData[3].ToString("X2");
       t_B5.Text = BlockData[4].ToString("X2");
       t_B6.Text = BlockData[5].ToString("X2");
       t_B7.Text = BlockData[6].ToString("X2");
       t_B8.Text = BlockData[7].ToString("X2");
       t_B9.Text = BlockData[8].ToString("X2");
       t_B10.Text = BlockData[9].ToString("X2");
       t_B11.Text = BlockData[10].ToString("X2");
       t_B12.Text = BlockData[11].ToString("X2");
       t_B13.Text = BlockData[12].ToString("X2");
       t_B14.Text = BlockData[13].ToString("X2");
       t_B15.Text = BlockData[14].ToString("X2");
       t_B16.Text = BlockData[15].ToString("X2");


    }
  else //Read not successful
  {

if (ReturnCode == 0x4E) //'N'
   textBox1.Text = "No Tag found. Error Code:" + ReturnCode.ToString("X2");
 else if (ReturnCode == 0x46) //'F'
   textBox1.Text = "Read Failed:" + ReturnCode.ToString("X2");
 else
   textBox1.Text = "Communication Error. Error Code:" + ReturnCode.ToString("X2");

   textBox_S.Text = textBox1.Text;

   }
```

2.8 CMD_WriteBlock

This function sends Write Block command to the mifare device. To write a block successfully, prior Select Tag and Authentication commands needs to be successful. Write Block command will write 16 bytes to the specified block. Write will be successful only if the given block is authenticated successfully. Writing to sector trailer blocks (last block at each sector) is not allowed with this function to prevent locking the Tag mistakenly that can happen if user does not have basic information for Mifare applications.

Application Hint

Before performing any read/write/value operations on a Mifare Tag, the tag should be **selected** first. Then the given block needs to be authenticated.

Mifare Classic 1K and 4K blocks consist of 16 bytes data. There are 64 blocks (0-63) for Mifare 1K, and 256 blocks (0-255) for Mifare 4K tags.

```
bool CMD_WriteBlock(byte BlockNo,byte[] BlockData, out byte
ReturnCode)
```

Arguments	**BlockNo:** Block number to be written. **BlockData:** 16 BYTE data to be written to given block BlockData[0]….BlockData[15] **ReturnCode:** If the Write Block command is not successful then the error code will be passed to this parameter. Possible Return Code: 0x00 – Communication Error 0x01 – The given block is Sector Trailer. Write failed 0x4E ‘N’ – No Tag present. Write Failed 0x46 ‘F’ – Write Failed 0x55 ‘U’ – Read after write failed 0x58 ‘X’ – Unable to read after write
Returns	True : Write Block operation is successful False : Write Block operation is not successful

Example:

```
sm_mifare_lib.mifare sm132 = new sm_mifare_lib.mifare();

byte BlockNo = 0;
byte[] BlockData = new byte[16];
byte ReturnCode = 0;

 BlockNo = byte.Parse(cmb_RWBlockNo.Text);
 BlockData[0] = byte.Parse(t_B1.Text, System.Globalization.NumberStyles.HexNumber);
 BlockData[1] = byte.Parse(t_B2.Text, System.Globalization.NumberStyles.HexNumber);
 BlockData[2] = byte.Parse(t_B3.Text, System.Globalization.NumberStyles.HexNumber);
 BlockData[3] = byte.Parse(t_B4.Text, System.Globalization.NumberStyles.HexNumber);
 BlockData[4] = byte.Parse(t_B5.Text, System.Globalization.NumberStyles.HexNumber);
 BlockData[5] = byte.Parse(t_B6.Text, System.Globalization.NumberStyles.HexNumber);
 BlockData[6] = byte.Parse(t_B7.Text, System.Globalization.NumberStyles.HexNumber);
 BlockData[7] = byte.Parse(t_B8.Text, System.Globalization.NumberStyles.HexNumber);
 BlockData[8] = byte.Parse(t_B9.Text, System.Globalization.NumberStyles.HexNumber);
 BlockData[9] = byte.Parse(t_B10.Text, System.Globalization.NumberStyles.HexNumber);
 BlockData[10] = byte.Parse(t_B11.Text, System.Globalization.NumberStyles.HexNumber);
 BlockData[11] = byte.Parse(t_B12.Text, System.Globalization.NumberStyles.HexNumber);
 BlockData[12] = byte.Parse(t_B13.Text, System.Globalization.NumberStyles.HexNumber);
 BlockData[13] = byte.Parse(t_B14.Text, System.Globalization.NumberStyles.HexNumber);
 BlockData[14] = byte.Parse(t_B15.Text, System.Globalization.NumberStyles.HexNumber);
 BlockData[15] = byte.Parse(t_B16.Text, System.Globalization.NumberStyles.HexNumber);

if (sm132.CMD_WriteBlock(BlockNo,BlockData, out ReturnCode))
 {
    textBox1.Text = "Write is Successfull";
    textBox_S.Text = textBox1.Text;
 }
  else //Write not successful
  {

    if (ReturnCode == 0x4E) //'N'
    textBox1.Text = "No Tag found. Error Code:" + ReturnCode.ToString("X2");
    else if (ReturnCode == 0x46) //'F'
    textBox1.Text = "Write Failed:" + ReturnCode.ToString("X2");
    else if (ReturnCode == 0x55) //'U'
    textBox1.Text = "Read after write Failed:" + ReturnCode.ToString("X2");
    else if (ReturnCode == 0x58) //'X'
    textBox1.Text = "Unable to read after write" + ReturnCode.ToString("X2");
    else if (ReturnCode == 0x01) //
    textBox1.Text = "Writing to Sector Trailer is not allowed with this function" +
ReturnCode.ToString("X2");
    else
    textBox1.Text = "Communication Error. Error Code:" + ReturnCode.ToString("X2");

    textBox_S.Text = textBox1.Text;

   }
```

2.9 CMD_ReadValue

This function sends Read Value command to the mifare device. To Read a value block successfully, prior Select Tag and Authentication commands needs to be successful and the format of the block should be value block, in other words, the block should have been written value with CMD_WriteValue previously.

Application Hint

Before performing any read/write/value operations on a Mifare Tag, the tag should be **selected** first. Then the given block needs to be authenticated.

Value block is a special formatted block and one of Mifare's most important features. User can write and read 4 byte signed number to these blocks. There are also Increment and Decrement Value commands. These commands can add to or subtract from the value block with the determined amount with just one command. Mifare tag will handle addition and subtraction routines.

Every sector, have a special block called Sector Trailer. Sector trailer block includes the access conditions and "key" (password) information of the blocks that belongs to that sector.

Block 0 of the Mifare Tag where Mifare Serial number is stored and the Sector Trailer Blocks could not be used as Value blocks. Any other blocks can be used as Value blocks.

To format a block for Value operation, CMD_WriteValue should have been used at first.

```
bool CMD_ReadValue(byte BlockNo, out long Value, out byte ReturnCode)
```

Arguments	**BlockNo:** Block number formatted as value block **Value:** 4 Byte signed integer value written in the block **ReturnCode:** If the Read Value command is not successful then the error code will be passed to this parameter. Possible Return Code: 0x00 - Communication Error 0x4E 'N' - No Tag present. 0x46 'F' - Read Failed 0x49 'I' - Invalid Value Block.
Returns	True : Read Value operation is successful False : Read Value operation is not successful

Example:

```
sm_mifare_lib.mifare sm132 = new sm_mifare_lib.mifare();

byte BlockNo = 0;
long Value = 0;
byte ReturnCode = 0;

BlockNo = byte.Parse(cmb_RWBlockNo.Text);

 if (sm132.CMD_ReadValue(BlockNo, out Value, out ReturnCode))
 {
     textBox1.Text = "Value Read Successfull";
     textBox_S.Text = textBox1.Text;
     t_ReadValue.Text = Value.ToString();
 }
else //ReadValue not successful
 {
     if (ReturnCode == 0x4E) //'N'
     textBox1.Text = "No Tag found. Error Code:" + ReturnCode.ToString("X2");
     else if (ReturnCode == 0x46) //'F'
     textBox1.Text = "Read Failed. Error Code:" + ReturnCode.ToString("X2");
     else if (ReturnCode == 0x49) //'I'
     textBox1.Text = "Invalid Value Block. Error Code:" + ReturnCode.ToString("X2");
     else
     textBox1.Text = "Communication Error. Error Code:" + ReturnCode.ToString("X2");

     textBox_S.Text = textBox1.Text;

}
```

2.10 CMD_WriteValue

This function sends Write Value command to the mifare device. To write a value block successfully, prior Select Tag and Authentication commands needs to be successful. CMD_WriteValue command will format the determined block for value operation with the given Value.

Application Hint

Before performing any read/write/value operations on a Mifare Tag, the tag should be **selected** first. Then the given block needs to be authenticated.

Value block is a special formatted block and one of Mifare's most important features. User can write and read 4 byte signed number to these blocks. There are also Increment and Decrement Value commands. These commands can add to or subtract from the value block with the determined amount with just one command. Mifare tag will handle addition and subtraction routines.

Every sector, have a special block called Sector Trailer. Sector trailer block includes the access conditions and "key" (password) information of the blocks that belongs to that sector.

Block 0 of the Mifare Tag where Mifare Serial number is stored and the Sector Trailer Blocks could not be used as Value blocks. Any other blocks can be used as Value blocks.

To format a block for Value operation, CMD_WriteValue should have been used at first.

`bool CMD_WriteValue(byte BlockNo,long Value, out byte ReturnCode)`	
Arguments	**BlockNo:** Block number to be written with Value **Value:** 4 Byte signed integer value will be written in the block **ReturnCode:** If the Write Value command is not successful then the error code will be passed to this parameter. Possible Return Code: 0x00 - Communication Error 0x4E 'N' - No Tag present. 0x46 'F' - Write Failed 0x55 'U' - Read after write failed. 0x58 'X' - Unable to read after write.
Returns	True : Write Value operation is successful False : Write Value operation is not successful

Example:

```
sm_mifare_lib.mifare sm132 = new sm_mifare_lib.mifare();

byte BlockNo = 0;
long Value = 0;
byte ReturnCode = 0;

BlockNo = byte.Parse(cmb_RWBlockNo.Text);
Value = Int32.Parse(t_WriteValue.Text);

if (sm132.CMD_WriteValue(BlockNo,Value, out ReturnCode))
{
     textBox1.Text = "Write Value is Successfull";
     textBox_S.Text = textBox1.Text;
     t_ReadValue.Text = Value.ToString();
 }
 else //WriteValue not successful
 {
 if (ReturnCode == 0x4E) //'N'
 textBox1.Text = "No Tag found. Error Code:" + ReturnCode.ToString("X2");
 else if (ReturnCode == 0x46) //'F'
 textBox1.Text = "Write Failed. Error Code:" + ReturnCode.ToString("X2");
 else if (ReturnCode == 0x58) //'X'
textBox1.Text = "Unable to read after write. Error Code:" + ReturnCode.ToString("X2");
else if (ReturnCode == 0x55) //'U'
textBox1.Text = "Read after write failed. Error Code:" + ReturnCode.ToString("X2");
else
textBox1.Text = "Communication Error. Error Code:" + ReturnCode.ToString("X2");

textBox_S.Text = textBox1.Text;

 }
```

2.11 CMD_IncrementValue

This function sends Increment Value command to the mifare device. To Increment a value block successfully, prior Select Tag and Authentication commands needs to be successful and the format of the block should be a value block, in other words, the block should have been written value with CMD_WriteValue previously. User can select desired increment quantity.

Application Hint

Before performing any read/write/value operations on a Mifare Tag, the tag should be **selected** first. Then the given block needs to be authenticated.

Value block is a special formatted block and one of Mifare's most important features. User can write and read 4 byte signed number to these blocks. There are also Increment and Decrement Value commands. These commands can add to or subtract from the value block with the determined amount with just one command. Mifare tag will handle addition and subtraction routines.

Every sector, have a special block called Sector Trailer. Sector trailer block includes the access conditions and "key" (password) information of the blocks that belongs to that sector.

Block 0 of the Mifare Tag where Mifare Serial number is stored and the Sector Trailer Blocks could not be used as Value blocks. Any other blocks can be used as Value blocks.

To format a block for Value operation, CMD_WriteValue should have been used at first.

```
bool CMD_IncrementValue(byte BlockNo, long IncValue,out long NewValue,
out byte ReturnCode)
```

Arguments	**BlockNo:** Block number to be incremented **Value:** 4 Byte signed integer value to be added/incremented to existing value in the block **NewValue:** 4 Byte signed integer value (final value) after increment is done. **ReturnCode:** If the Increment Value command is not successful then the error code will be passed to this parameter. Possible Return Code: 0x00 - Communication Error 0x4E 'N' - No Tag present. 0x46 'F' - Read Failed during verification. 0x49 'I' - Invalid Value Block.
Returns	True : Increment Value operation is successful False : Increment Value operation is not successful

Example:

```
sm_mifare_lib.mifare sm132 = new sm_mifare_lib.mifare();

byte BlockNo = 0;
long IncValue = 0;
long NewValue = 0;
byte ReturnCode = 0;

BlockNo = byte.Parse(cmb_RWBlockNo.Text);
IncValue = Int32.Parse(t_IncDec.Text);

if (sm132.CMD_IncrementValue(BlockNo, IncValue, out NewValue,out ReturnCode))
{
    textBox1.Text = "Increment is successfull";
    textBox_S.Text = textBox1.Text;
    t_ReadValue.Text = NewValue.ToString();
}
else //Increment Value not successful
{

  if (ReturnCode == 0x4E) //'N'
  textBox1.Text = "No Tag found. Error Code:" + ReturnCode.ToString("X2");
  else if (ReturnCode == 0x46) //'F'
  textBox1.Text = "Read Failed during verif. Error Code:" + ReturnCode.ToString("X2");
  else if (ReturnCode == 0x49) //'I'
  textBox1.Text = "Invalid Value Block. Error Code:" + ReturnCode.ToString("X2");
  else
  textBox1.Text = "Communication Error. Error Code:" + ReturnCode.ToString("X2");

  textBox_S.Text = textBox1.Text;

}
```

2.12 CMD_DecrementValue

This function sends Decrement Value command to the mifare device. To Decrement a value block successfully, prior Select Tag and Authentication commands needs to be successful and the format of the block should be a value block, in other words, the block should have been written value with CMD_WriteValue previously. User can select desired decrement quantity.

Application Hint

Before performing any read/write/value operations on a Mifare Tag, the tag should be **selected** first. Then the given block needs to be authenticated.

Value block is a special formatted block and one of Mifare's most important features. User can write and read 4 byte signed number to these blocks. There are also Increment and Decrement Value commands. These commands can add to or subtract from the value block with the determined amount with just one command. Mifare tag will handle addition and subtraction routines.

Every sector, have a special block called Sector Trailer. Sector trailer block includes the access conditions and "key" (password) information of the blocks that belongs to that sector.

Block 0 of the Mifare Tag where Mifare Serial number is stored and the Sector Trailer Blocks could not be used as Value blocks. Any other blocks can be used as Value blocks.

To format a block for Value operation, CMD_WriteValue should have been used at first.

`bool CMD_DecrementValue(byte BlockNo, long DecValue, out long NewValue, out byte ReturnCode)`	
Arguments	**BlockNo:** Block number to be decremented **Value:** 4 Byte signed integer value to be substracted/decremented from the existing value in the block **NewValue:** 4 Byte signed integer value (final value) after decrement is performed. **ReturnCode:** If the Decrement Value command is not successful then the error code will be passed to this parameter. Possible Return Code: 0x00 - Communication Error 0x4E 'N' - No Tag present. 0x46 'F' - Read Failed during verification. 0x49 'I' - Invalid Value Block.
Returns	True : Decrement Value operation is successful False : Decrement Value operation is not successful

Example:

```
sm_mifare_lib.mifare sm132 = new sm_mifare_lib.mifare();

byte BlockNo = 0;
long DecValue = 0;
long NewValue = 0;
byte ReturnCode = 0;

BlockNo = byte.Parse(cmb_RWBlockNo.Text);
DecValue = Int32.Parse(t_IncDec.Text);

if (sm132.CMD_DecrementValue(BlockNo, DecValue, out NewValue, out ReturnCode))
{
  textBox1.Text = "Decrement is successfull";
  textBox_S.Text = textBox1.Text;
  t_ReadValue.Text = NewValue.ToString();
 }
 else //Decrement Value not successful
 {
  if (ReturnCode == 0x4E) //'N'
  textBox1.Text = "No Tag found. Error Code:" + ReturnCode.ToString("X2");
  else if (ReturnCode == 0x46) //'F'
  textBox1.Text = "Read Failed during verif. Error Code:" + ReturnCode.ToString("X2");
  else if (ReturnCode == 0x49) //'I'
  textBox1.Text = "Invalid Value Block. Error Code:" + ReturnCode.ToString("X2");
  else
  textBox1.Text = "Communication Error. Error Code:" + ReturnCode.ToString("X2");

  textBox_S.Text = textBox1.Text;

 }
```

2.13 CMD_SwitchRF

This command turns ON or OFF the RF field. RF field can be switched off when it is not required. This helps to reduce the active current consumption. The RF field can be switched ON whenever a read or write operation is required.

```
bool CMD_SwitchRF(bool OnOffState, out byte ReturnCode)
```

Arguments	**OnOffState:** Use “True” to Switch ON RF , False to turn off RF **ReturnCode:** If the Switch RF command is not successful then the error code will be passed to this parameter. Possible Return Code: 0x00 – Communication Error
Returns	True : Switch RF On/Off operation is successful False : Switch RF On/Off operation is not successful

Example:

```
sm_mifare_lib.mifare sm132 = new sm_mifare_lib.mifare();

byte ReturnCode = 0;


if (sm132.CMD_SwitchRF(false, out ReturnCode))
textBox1.Text = " RF Switched off";
else
textBox1.Text = "Switch Off.Error Code:" + ReturnCode.ToString("X2");


if (sm132.CMD_SwitchRF(true, out ReturnCode))
textBox1.Text = " RF Switched On";
else
textBox1.Text = "Switch On. Error Code:" + ReturnCode.ToString("X2");
```

3. SALES AND SERVICE INFORMATION

To obtain information about SonMicro Electronics products and technical support, reference the following information.

SonMicro ELECTRONICS LTD.
Cankaya M. Soguksu C.
Aslihan Ishani 2/15
Mersin, 33070
TURKIYE

Phone:	+90 324 237 21 28	
Facsimile:	+90 324 237 21 86	
Email:	info@sonmicro.com	
Web Site:	http://www.sonmicro.com	
	Sales	http://www.sonmicro.com/sales.php
	Support	http://www.sonmicro.com/ask.php
	Documents & Software	http://www.sonmicro.com/1356/d1356.php
	User Forums	http://www.sonmicro.com/forums/