/**

Magnetic Strip Reader DLL User Guide

First Written in : 2012-04-19

**/

Content:

1. Development Environment

2. Usage

3. Execution

4. API Reference

5. Program Flow Sheet

6.Example for VS 2005

1. Development Environment

Visual Studio 2005 C# .

2. Usage

(1) Library header
: “MSR_DLL.h” is the library header file.
(2) Library file
: “MSR_DLL.dll” is the library

3. Execution

(1) Copy the provided MSR_DLL.dll and MF2351DLL.dll to the device. The location can be the same directory where an application exists or “ \ Windows” directory.
(2) Copy the application that uses the MSR library in the device and execute it.

4. API Reference

[1] CODEAPI UINT WINAPI MSROpen(HWND);

Description :

Turns on the MSR power, opens the serial port and sets the window handle to receive data.

Return Values :

MSR_FAIL : Failed to open MSR reader
MSR_OK : Success

Parameter :

HWND: [in] the handle of the window to receive data

---

[2] CODEAPI UINT WINAPI MSRClose(void);

Description :

Turns off the MSR power, closes the serial port and releases the window handle.

Return Values :

MSR_FAIL : Failed to close the Card
MSR_OK : Success

---

[3] CODEAPI UINT WINAPI MSRBeepOn(BOOL bBeep);

Description :

Sets the beep sound to notify successful reading of the card data.

Return Value :

MSR_FAIL : Failed to on/off the beep sound..
MSR_OK : Success
MSR_ERROR_NOT_OPENED : The MSR has not been Opened.

Parameter :

BOOL bBeep: [in] TRUE Set the beep sound
FALSE Clear the beep sound

---

[4] CODEAPI UINT WINAPI MSRRead(int CardType, char *a_pData, int length);

Description :

Reads the card data.

Return Value :

MSR_ERROR_NOT_OPENED : The MSR has not been Opened.

UINT : the actual length of the data was read.

Parameters :

int CardType : [in] Track number

TRK_1, read track 1 data

TRK_2, read track 2 data

TRK_3, read track 3 data

ALL_DATA, read all data,include track1,track2,track3

char *a_pData : [in,out] pointer to the buffer which is used for storing the data you get from the card.

int length : [in] the length of the buffer which is used for storing the data you get from the card.

---

[5] CODEAPI UINT WINAPI MSRReadString(int CardType, TCHAR *a_tData, int length);

Description :

Reads the card data.

Return Value :

MSR_ERROR_NOT_OPENED : The MSR has not been Opened.

UINT : the actual length of the data was read.

Parameters :

int CardType : [in] Track number

TRK_1, read track 1 data

TRK_2, read track 2 data

TRK_3, read track 3 data

ALL_DATA, read all data,include track1,track2,track3

TCHAR *a_tData : [in,out] pointer to the buffer which is used for storing the data get from the card. **The buffer length of the user based on their actual conditions, the general TRK_1 not less than 79, TRK_2 not less than 40, TRK_3 not less than 107, ALL_DATA not less than 226.**

int length : [in] the length of buffer which is used for storing the data get from the card.

---

5. Application Flow：

1) Read card

step1: Open the card

MSROpen(MsgWin.Hwnd);

step2: Wait message

override void WndProc(ref Message msg);

a. track 1 data received

WM_MSR_TRACK_1_RECEIVE:

read track 1 data

MSRReadString(TRK_1,…,…);

b. track 2 data received

WM_MSR_TRACK_2_RECEIVE:

read track 2 data

MSRReadString(TRK_2,…,…);

c. track 3 data received

WM_MSR_TRACK_3_RECEIVE:

read track 3 data

MSRReadString(TRK_3,…,…);

d. all tracks data received

WM_COMM_RXCHAR_COMPLETE:

read all tracks data

MSRReadString(ALL_DATA,…,…);

e. fail to read track data

WM_MSR_ERROR_READ:

step3: Close the card

CardClose();

2) Set beep

step1: Open the card

MSROpen(MsgWin.Hwnd);

step2: Turn on the beep

MSRBeepOn(true) ;

Turn off the beep

MSRBeepOn(false);

step3: Close the card

MSRClose();

6. Example for VS2005 C#

1) Call the API of MSR_DLL.dll

```
namespace MSRSample
{
    class MSR_DLL
    {
        public const string DLLName = "MSR_DLL.dll";
        public const Int32 GET_TRK_1 = 1;
        public const Int32 GET_TRK_2 = 2;
        public const Int32 GET_TRK_3 = 3;
```

```
        public const Int32 GET_ALL = 4;

        public const Int32 WM_USER = 0x0400;
        public const Int32 WM_COMM_RXCHAR_MESSAGE = WM_USER + 1101;
        public const Int32 WM_COMM_RXCHAR_COMPLETE = WM_USER + 1102;
        public const Int32 WM_CARD_TRACK_1_RECEIVE = WM_USER + 1106;
        public const Int32 WM_CARD_TRACK_2_RECEIVE = WM_USER + 1103;
        public const Int32 WM_CARD_TRACK_3_RECEIVE = WM_USER + 1104;
        public const Int32 WM_CARD_ERROR_READ = WM_USER + 1105;

        [DllImport(DLLName, SetLastError = true, EntryPoint = "MSROpen")]
        public static extern UInt32 MSROpen(IntPtr hAppWinHnd);

        [DllImport(DLLName, SetLastError = true, EntryPoint = "MSRClose")]
        public static extern UInt32 MSRClose();

        [DllImport(DLLName, SetLastError = true, EntryPoint = "MSRRead")]
        public static extern UInt32 MSRRead(Int32 CardType, char[] a_pData, Int32 length);

        [DllImport(DLLName, SetLastError = true, EntryPoint = "MSRReadString")]
        public static extern UInt32 MSRReadString(Int32 CardType, char[] a_tData, Int32 length);

        [DllImport(DLLName, SetLastError = true, EntryPoint = "MSRBeepOn")]
        public static extern UInt32 MSRBeepOn(bool bBeep);

        public const Int32 WM_CLOSE = 0x0010;

        [DllImport("coredll.dll", EntryPoint = "FindWindow", SetLastError = true,
        CharSet = CharSet.Auto, CallingConvention = CallingConvention.Winapi)]
        public static extern IntPtr FindWindow(string lpClassName, string lpWindowName);

        [DllImport("coredll.dll", EntryPoint = "PostMessage", SetLastError = true,
        CharSet = CharSet.Auto, CallingConvention = CallingConvention.Winapi)]
        public static extern IntPtr PostMessage(IntPtr hwnd, UInt32 wMsg, UInt32 wParam, UInt32
lParam);
    }
}
```

2) Application sample code

```
namespace MSRSample
{
    public partial class MSRSample : Form
    {
```

```
    private MsgWindow MsgWin;
    public MSRSample()
    {
        InitializeComponent();
        this.MsgWin = new MsgWindow(this);
    }
    private void MSROpen_Click(object sender, EventArgs e)
    {
        MSR_DLL.MSROpen(MsgWin.Hwnd);
    }

    private void MSRClose_Click(object sender, EventArgs e)
    {
        MSR_DLL.MSRClose();
    }

    private void BeepOn_Click(object sender, EventArgs e)
    {
        MSR_DLL.MSRBeepOn(true);
    }

    private void BeepOff_Click(object sender, EventArgs e)
    {
        MSR_DLL.MSRBeepOn(false);
    }
}
public class MsgWindow : MessageWindow
{
    private MSRSample msgform;
    public MsgWindow(MSRSample msgform)
    {
        this.msgform = msgform;
    }
    protected override void WndProc(ref Message msg)
    {
        char[] szTrack1 = new char[1024];
        char[] szTrack2 = new char[1024];
        char[] szTrack3 = new char[1024];
        char[] szRawdata = new char[1024];
        switch (msg.Msg)
        {
            case MSR_DLL.WM_COMM_RXCHAR_MESSAGE:
                break;
            case MSR_DLL.WM_CARD_TRACK_1_RECEIVE:
```

```
                    MSR_DLL.MSRReadString(MSR_DLL.GET_TRK_1, szTrack1, 1024);
                    break;
                case MSR_DLL.WM_CARD_TRACK_2_RECEIVE:
                    MSR_DLL.MSRReadString(MSR_DLL.GET_TRK_2, szTrack2, 1024);
                    break;
                case MSR_DLL.WM_CARD_TRACK_3_RECEIVE:
                    MSR_DLL.MSRReadString(MSR_DLL.GET_TRK_3, szTrack3, 1024);
                    break;
                case MSR_DLL.WM_COMM_RXCHAR_COMPLETE:
                    MSR_DLL.MSRReadString(MSR_DLL.GET_ALL, szRawdata, 1024);
                    break;
                case MSR_DLL.WM_CARD_ERROR_READ:
                    break;
            }
            base.WndProc(ref msg);
        }
    }
}
```