/**

Barcode Reader DLL User Guide 1.1

First Written in : 2012-04-18

**/

Content:

1. Development Environment

2. Usage

3. Execution

4. API Reference

5. Program Flow Sheet

6. Example for VS2005.

1. Development Environment

Visual Studio 2005 C# .

2. Usage

(1) Library header

: “BCR_DLL.h” is the library header file.

(2) Library file

: “BCR_DLL.dll” is the library

3. Execution

(1) Copy the provided BCR_DLL.dll and MF2351DLL.dll to the device. The location can be“ \ Windows” directory or the same directory with the application which uses the library .

(2) Copy the application that uses the BCR library in the device and execute it.

4. API Reference

[1] CODEAPI UINT WINAPI BCROpen(HWND);

Description :

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

Return Values :

BARCODE_FAIL : the BCR has been opened once .

BARCODE_OK : Success

BARCODE_ERROR_POWERON : Failed to turn on power

Parameter :

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

---

[2] CODEAPI UINT WINAPI BCRClose(void);

Description :

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

Return Values :

BARCODE_FAIL : the BCR has been Closed .

BARCODE_OK : Success

---

[3] CODEAPI UINT WINAPI `BCREnableSideKey` (BOOL `bEanble`);

Description :

Enable/Disable the side key to trigger on. This should be set before BCROpen () executed. Default: enable the side key to trigger on.

Return Values :

BARCODE_OK : Success

---

[4] CODEAPI UINT WINAPI BCRBeepOn(BOOL bBeep);

Description :

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

Return Value :

BARCODE_FAIL: the BCR has been Closed .

BARCODE_OK : Success

Parameter :

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

FALSE :Clear the beep sound

---

[5] CODEAPI UINT WINAPI BCRRead(char *a_pData, int length);

CODEAPI UINT WINAPI BCRReadString(TCHAR *a_tData, int length);

Description :

Reads the barcode data.

Return Value :

BARCODE_FAIL : the BCR has been Closed .

UINT : the actual length of the data was read.

Parameters :

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

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 the actual situation to decide, the general is not too small.**

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

---

[6] CODEAPI UINT WINAPI BCRTriggerStart(void);

Description :

Start barcode scanning .

Return Value :

BARCODE_FAIL: the BCR has been Closed .

BARCODE_ERROR_POWEROFF: failed.

BARCODE_OK : Success

---

[7] CODEAPI UINT WINAPI BCRTriggerStop(void);

Description :

Stop barcode scanning .

Return Value :

0L : the BCR has been Closed .

BARCODE_ERROR_POWEROFF: failed.

BARCODE_OK : Succes

---

[8] CODEAPI UINT WINAPI BCRSetDefault(void);

Description :

Set the Barcode Scanner in Default situation. BCR models must be reopened after this command.

Return Value :

BARCODE_FAIL : the BCR has been Closed .

UINT : TRUE （Non-zero）:Success.

FALSE（zero）:Failed.

---

[9] CODEAPI UINT WINAPI BCRSetParameters(char paramNumber, char paramValue, BOOL bTemporary);

Description :

1. Support or not support one barcode type .
2. Support the length of barcode type.
3. Set trigger mode.
4. Transmit code ID charactor.

   The code ID character is inserted between the prefix character(if selected) and the decode symbol. The symbol code ID character are listed below

   A = UPC-A, UPC-E, UPC-E1, EAN-8, EAN-13

   B = Code 39, Code32

   C = Codabar

   D = Code 128, ISBT 128

   E = Code 93

   F = Interleaved 2 of 5

   G = Discrete 2 of 5

   J = MSI

   K = GS1-Databar

   L = Bookland EAN

   M = Trioptic Code 39

   N = Coupon Code

   R = GS1 Databar-14, GS1 Databar Limited, GS1 Databar Expanded
5. Set add prefix and suffix type
6. Add prefix、suffix

Return Value :

BARCODE_FAIL: the BCR has been Closed .

UINT : TRUE （Non-zero）:Success.

FALSE（zero）:Failed.

Parameters :

char paramNumber : [in]

```
1.set enable/disable barcode type
BARCODE_TYPE_CODE39            0x00
BARCODE_TYPE_CODABAR           0x07
BARCODE_TYPE_CODE128           0x08
```

```
BARCODE_TYPE_DISCRETE2OF5       0x05
BARCODE_TYPE_INTERLEAVED2OF5    0x06
BARCODE_TYPE_CODE93             0x09
BARCODE_TYPE_UPC_A              0x01
BARCODE_TYPE_UPC_E              0x02
BARCODE_TYPE_UPC_E1             0x0C
BARCODE_TYPE_EAN8               0x04
BARCODE_TYPE_EAN13              0x03
BARCODE_TYPE_MSIPLESSEY         0x0B
BARCODE_TYPE_EAN128             0x08
BARCODE_UCCEAN_128_CODE         0x0E
BARCODE_ISBT_128_CODE           0x54
BARCODE_TYPE_TRIOPTIC_CODE39    0x0D
BARCODE_TYPE_BOOKLAND_EAN       0x53
BARCODE_TYPE_COUPON_CODE        0x55
2.set barcode length
LENGTH_INTERLEAVED2OF5_CODE     0x16
LENGTH_DISCRETE2OF5_CODE        0x14
LENGTH_CODABAR_CODE             0x18
 3.set trigger mode
 BAR_SET_TRIGGER              0x8A
 4.add\delete transmit code ID charactor
 BAR_SET_CODEID               0x2D
 5.set adding prefix、suffix type
 BAR_SET_PRESUFFIX            0xEB
 6.add prefix、suffix
 SET_PREFIX                   0x69
 SET_SUFFIX1                  0x68
 SET_SUFFIX2                  0x6A
```

char paramValue : [in]

```
//1.set enable/disable barcode type
 BARCODE_ENABLE          0x01
 BARCODE_DISABLE         0x00
 2.set barcode length
 ANY_LENGTH                   0x00
 3.set trigger mode
 TRIGGER_PULSE                0x02
 TRIGGER_HOST                 0x08
 TRIGGER_BLINK                0x07
 TRIGGER_CONTINUOUS           0x04
 TRIGGER_LEVEL                0x00
 4.add\delete transmit code ID charactor
 ENABLE_CODEID                0x02
```

```
DISABLE_CODEID              0x00
5.set adding prefix、suffix type
SET_MODE_PRESUF1SUF2        0x07
SET_MODE_NONE               0x00
SET_MODE_SUF1               0x01
SET_MODE_SUF2               0x02
SET_MODE_SUF1SUF2           0x03
SET_MODE_PRE                0x04
SET_MODE_PRESUF1            0x05
SET_MODE_PRESUF2            0x06
6.add prefix、suffix
The char to be added
```

BOOL bTemporary : [in] TRUE （Temporary change - lost when power removed.）
FALSE （Permanent change.）

5. Application Flow Sheet：

(1)Read data.

step1: Open the Scaner

```
BCREnableSideKey(bool)
BCROpen(MsgWin.Hwnd);
```

Step2: Set trigger mode ： “Host”

```
CodeSetParameters(BAR_SET_TRIGGER, TRIGGER_HOST, true);
```

Step3: Start barcode scanning

```
BCRTriggerStart();
```

Step4 : Wait message for barcode data received

```
WM_BARCODE_COMPLETE
```

Step5: Read the data

```
BCRReadString(…,…);
```

Step6: Stop barcode scanning

```
BCRTriggerStop();
```

Step7: Close Scanning Equipment

```
CodeClose();
```

(2)Set beep.

step1: Open the Scaner

```
BCROpen(MsgWin.Hwnd);
```

step2: Turn on the beep.

```
BCRtBeepOn(true);
```

Turn off the beep.

```
BCRBeepOn(false);
```

step3: Close Scaner

```
BCRClose();
```

(3) Enable or Disnable one Type barcode

step1: Open the Scaner

BCROpen(MsgWin.Hwnd);

step2: Enable or disnable one type

BCRSetParameters(...,...,false);

step3: Close Scaner

BCRClose();

(4) Set trigger mode

step1: Open the Scaner

BCROpen(MsgWin.Hwnd);

step2: Set trigger mode

BCRSetParameters(BAR_SET_TRIGGER, TRIGGER_BLINK, false); //Blink

BCRSetParameters(BAR_SET_TRIGGER, TRIGGER_PULSE, false); //Pulse

step3: Close Scaner

BCRClose();

(5) Transmit code ID charactor

step1: Open the Scaner

BCROpen(MsgWin.Hwnd);

step2: Set transmit code ID charactor

add transmit code ID charactor

BCRSetParameters(BAR_SET_CODEID, ENABLE_CODEID, false);

delete transmit code ID charactor

BCRSetParameters(BAR_SET_CODEID, DISABLE_CODEID, false);

step3: Close Scaner

BCRClose();

(6) Add prefix、suffix

step1: Open the Scaner

BCROpen(MsgWin.Hwnd);

step2: Set the format type of adding prefix、suffix

BCRSetParameters(BAR_SET_PRESUFFIX, …, false);

step3: Add prefix、suffix

BCRSetParameters(SET_PREFIX, …, false);

BCRSetParameters(SET_SUFFIX1, …, false);

BCRSetParameters(SET_SUFFIX2, …, false);

step4: Close Scaner

BCRClose();

6. Example for VS2005 C#.

1) Call the API of BCR_DLL.dll

```
namespace BCR1DSample
{
    class BCR_DLL
    {
        public const string DLLName = "BCR_DLL.dll";
        public const Int32 BARCODE_DRIVER_VERSION = 100;
```

```
public const Int32 BARCODE_OK = 2000;
public const Int32 BARCODE_ERROR_POWERON = 2001;
public const Int32 BARCODE_ERROR_POWEROFF = 2002;
public const Int32 BARCODE_ERROR_SERIAL_OPEN = 2010;
public const Int32 BARCODE_ERROR_SERIAL_CLOSE = 2011;
public const Int32 BARCODE_ERROR_SERIAL_CLEAR = 2012;
public const Int32 BARCODE_ERROR_SERIAL_DATA = 2013;
public const Int32 BARCODE_ERROR_FILE_SETTING = 2014;
public const Int32 BARCODE_ERROR_ALREADY_USED = 2015;
//Set trigger mode
public const Int32 BAR_SET_TRIGGER = 0x8A;
public const Int32 TRIGGER_PULSE = 0x02;
public const Int32 TRIGGER_HOST = 0x08;
public const Int32 TRIGGER_BLINK = 0x07;
public const Int32 TRIGGER_CONTINUOUS = 0x04;
public const Int32 TRIGGER_LEVEL = 0x00;
//Transmit code ID charactor
public const Int32 BAR_SET_CODEID = 0x2D;
public const Int32 ENABLE_CODEID = 0x02;
public const Int32 DISABLE_CODEID = 0x00;
//Prefix and Suffix
public const Int32 BAR_SET_PRESUFFIX = 0xEB;
public const Int32 SET_MODE_PRESUF1SUF2 = 0x07;
public const Int32 SET_MODE_NONE = 0x00;
public const Int32 SET_MODE_SUF1 = 0x01;
public const Int32 SET_MODE_SUF2 = 0x02;
public const Int32 SET_MODE_SUF1SUF2 = 0x03;
public const Int32 SET_MODE_PRE = 0x04;
public const Int32 SET_MODE_PRESUF1 = 0x05;
public const Int32 SET_MODE_PRESUF2 = 0x06;
public const Int32 SET_PREFIX = 0x69;
public const Int32 SET_SUFFIX1 = 0x68;
public const Int32 SET_SUFFIX2 = 0x6A;

public const Int32 WM_USER = 0x0400;
public const Int32 WM_BARCODE_MESSAGE = WM_USER + 5001;
public const Int32 WM_BARCODE_COMPLETE = WM_USER + 5002;
public const Int32 BARCODE_ENABLE = 0x01;
public const Int32 BARCODE_DISABLE = 0x00;

// Decode Data Supperted Code Types
public const Int32 BARCODE_TYPE_CODE39 = 0x00;
public const Int32 BARCODE_TYPE_CODABAR = 0x07; // Default Disable
public const Int32 BARCODE_TYPE_CODE128 = 0x08;
```

```
        public const Int32 BARCODE_TYPE_DISCRETE2OF5 = 0x05;// Default Disable
        public const Int32 BARCODE_TYPE_INTERLEAVED2OF5 = 0x06;
        public const Int32 BARCODE_TYPE_CODE93 = 0x09; // Default Disable
        public const Int32 BARCODE_TYPE_UPC_A = 0x01;
        public const Int32 BARCODE_TYPE_UPC_E = 0x02;
        public const Int32 BARCODE_TYPE_UPC_E1 = 0x0C; // Default Disable
        public const Int32 BARCODE_TYPE_EAN8 = 0x04;
        public const Int32 BARCODE_TYPE_EAN13 = 0x03;
        public const Int32 BARCODE_TYPE_MSIPLESSEY = 0x0B; // Default Disable
        public const Int32 BARCODE_TYPE_EAN128 = 0x08;
        public const Int32 BARCODE_UCCEAN_128_CODE = 0x0E;
        public const Int32 BARCODE_ISBT_128_CODE = 0x54;
        public const Int32 BARCODE_TYPE_TRIOPTIC_CODE39 = 0x0D;
        public const Int32 BARCODE_TYPE_BOOKLAND_EAN = 0x53; // Default Disable
        public const Int32 BARCODE_TYPE_COUPON_CODE = 0x55;
        //barcode length.....................................................
        public const Int32 ANY_LENGTH = 0x00;
        public const Int32 LENGTH_INTERLEAVED2OF5_CODE = 0x16;
        public const Int32 LENGTH_DISCRETE2OF5_CODE = 0x14;
        public const Int32 LENGTH_CODABAR_CODE = 0x18;

        [DllImport(DLLName, SetLastError = true, EntryPoint = "CodeOpen")]
        public static extern UInt32 CodeOpen(IntPtr hAppWinHnd);
        [DllImport(DLLName, SetLastError = true, EntryPoint = "CodeClose")]
        public static extern UInt32 CodeClose();
        [DllImport(DLLName, SetLastError = true, EntryPoint = "BcrEnableSideKey")]
        public static extern UInt32 BcrEnableSideKey (bool bEnable);
        [DllImport(DLLName, SetLastError = true, EntryPoint = "CodeVersion")]
        public static extern UInt32 CodeVersion();
        [DllImport(DLLName, SetLastError = true, EntryPoint = "CodeRead")]
        public static extern UInt32 CodeRead(char[] a_pData, Int32 length);
        [DllImport(DLLName, SetLastError = true, EntryPoint = "CodeReadString")]
        public static extern UInt32 CodeReadString(char[] a_pData, Int32 length);
        [DllImport(DLLName, SetLastError = true, EntryPoint = "CodeSetDefault")]
        public static extern UInt32 CodeSetDefault();
        [DllImport(DLLName, SetLastError = true, EntryPoint = "CodeSetParameters")]
        public static extern UInt32 CodeSetParameters(char paramNumber, char paramValue, bool
bTemporary);
        [DllImport(DLLName, SetLastError = true, EntryPoint = "CodeTriggerStart")]
        public static extern UInt32 CodeTriggerStart();
        [DllImport(DLLName, SetLastError = true, EntryPoint = "CodeTriggerStop")]
        public static extern UInt32 CodeTriggerStop();
        [DllImport(DLLName, SetLastError = true, EntryPoint = "CodeSetBeep")]
        public static extern UInt32 CodeSetBeep(bool bBeep);
```

```
        [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 BCR1DSample
{
    public partial class BCR1DSample : Form
    {
        private MsgWindow MsgWin;
        public BCR1DSample()
        {
            InitializeComponent();
            this.MsgWin = new MsgWindow(this);
        }
        private void BCROpen_Click(object sender, EventArgs e)
        {
            BCR_DLL.BcrEnableSideKey(true);
            BCR_DLL.CodeOpen(MsgWin.Hwnd);
        }
        private void BCRClose_Click(object sender, EventArgs e)
        {
            BCR_DLL.CodeClose();
        }
        private void TriggerOn_Click(object sender, EventArgs e)
        {
           BCR_DLL.CodeSetParameters((char)BCR_DLL.BAR_SET_TRIGGER,
                   (char)BCR_DLL.TRIGGER_HOST, true);
           BCR_DLL.CodeTriggerStart();
        }
        private void TriggerOff_Click(object sender, EventArgs e)
        {
            BCR_DLL.CodeTriggerStop();
        }
        private void BeepOn_Click(object sender, EventArgs e)
        {
```

```
            BCR_DLL.CodeSetBeep(true);
        }
        private void BeepOff_Click(object sender, EventArgs e)
        {
            BCR_DLL.CodeSetBeep(false);
        }
    }
}
    public class MsgWindow : MessageWindow
    {
        private BCR1DSample msgform;
        public MsgWindow(BCR1DSample msgform)
        {
            this.msgform = msgform;
        }
        protected override void WndProc(ref Message msg)
        {
            char[] BarData = new char[1024];
            switch (msg.Msg)
            {
                case BCR_DLL.WM_BARCODE_COMPLETE:
                    BCR_DLL.CodeReadString(BarData, 1024);
                    break;
            }
            base.WndProc(ref msg);
        }
    }
}
```