Document reversion V1.0 2014-11-07

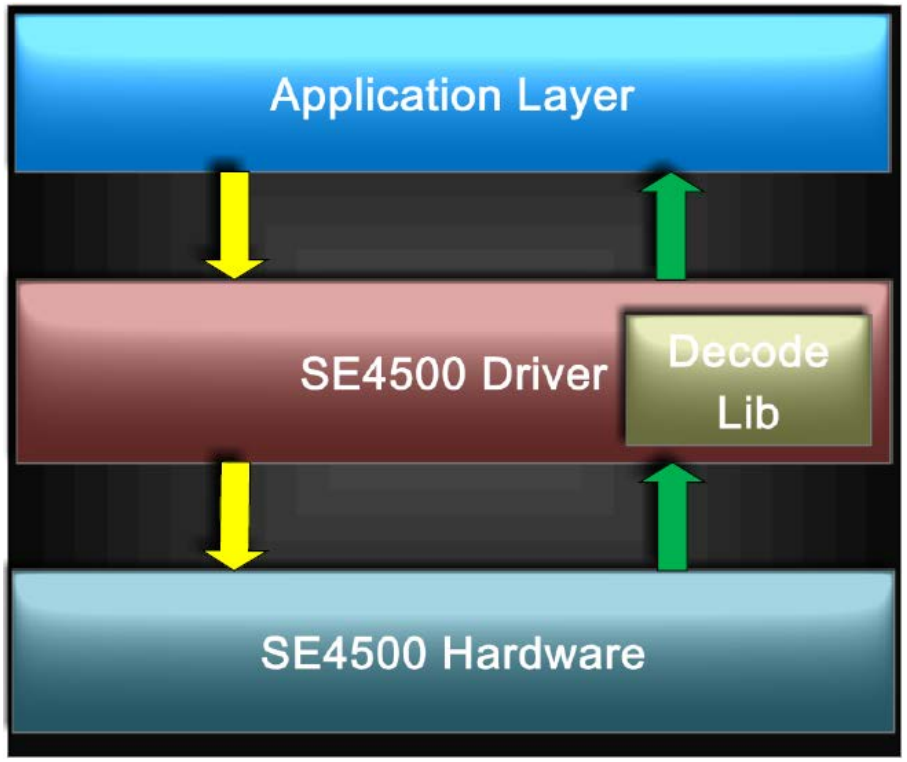
TT 43 Windows Mobile 6.5 SE 4500 2D Barcode Scanner Development User Manual

Yu Xie
11/07/2014

SE4500 2D Barcode Scanner Development

User Manual

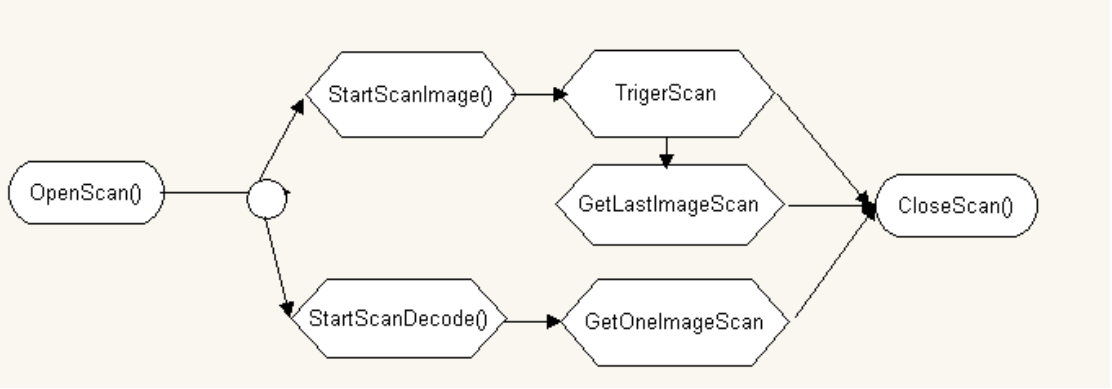
1. Introduction



System Structure

Application software should access SE4500 under lying hardware through SE4500 stream device driver interface, which provide a set of device IO controls for application layer software to control SE4500 hardware features, start or stop decode session, fetch decode result and image data and so on. The library of SE4500 is named 'MotoScanLib.dll'. The demo application is using the functions provided by the library to access the hardware.

2. Software Process



For the structures and macro definition used below, please refer to the appendix.

3. Open SE4500

bool OpenScan();

4. Start to scan

bool StartScanDecode();

5. Start to capture image

bool StartScanImage();

6. Wait for decoding to get the code

bool TrigerScan(refSCAN_SCANResult RetVal);

7. Get decode image

bool GetLastImageScan(refSDL_ImageDescriptor_s RetVal);
The decode image is captured after scanning.

8. Wait for capturing image to get image

bool GetOneImageScan(refSDL_ImageDescriptor_s RetVal);

9. Close SE4500

bool CloseScan();

10. Wait for decoding or image data example:

```
private void scan_thread()
{
HANDLE[] hWaitList = new HANDLE[2];
UInt32 udwRetVal;
UInt32 udwErrCount = 0;
bool fResult = false;
int len = 0;

const UInt32 WAIT_OBJECT_0 = 0;
const UInt32 WAIT_TIMEOUT = 0x00000102;

while (fRunning)
{
hWaitList[0] = this.hEventImage;
hWaitList[1] = this.hEventGen;

if (udwErrCount > 10) break;
udwRetVal = WaitForMultipleObjects(2, hWaitList, false, SCANTIMEOUT);
if (!fRunning)
break;

try
{
switch (udwRetVal)
{
case WAIT_OBJECT_0: // Image Event
{
fResult = SCAN_API.GetOneImageScan(ref oneInm);
if (fResult && (oneInm.udwActLength != 0))
{
showImage one = new showImage(DisPlayImage);
this.Invoke(one);
}
else
{
diserror two = new diserror(Dis_Error);
this.Invoke(two);
}
SCAN_API.CloseScan();
ScanRun = false;
```

```
udwErrCount = 0;
break;
}
case (WAIT_OBJECT_0 + 1): // Decode Event
{
fResult = SCAN_API.TrigerScan(ref In);
if (fResult)
{
len = (int)In.ScanCodeLen;
Decoded(this, new DecodeData(In.ScanCode, len));
}
else
{
diserror two = new diserror(Dis_Error);
this.Invoke(two);
}
SCAN_API.CloseScan();
ScanRun = false;
udwErrCount = 0;
break;
}
case WAIT_TIMEOUT:
{
break;
}
default:
{
udwErrCount += 1;
break;
}
}
}
catch
{
}
}
}
```

11. Appendix

Datastruct

```
public struct SCAN_SCANResult
{
public int ScanCodeLen;
public byte[] ScanCode;
```

```
}
public enum SDL_FRAME_ENCODING
{
SDL_IMAGE_ENCODING_GRAYSCALE = 1,
SDL_IMAGE_ENCODING_JPEG = 2,
SDL_IMAGE_ENCODING_BMP = 3,
SDL_IMAGE_ENCODING_TIFF = 4,
SDL_IMAGE_ENCODING_MAX
public struct SDL_ImageDescriptor_s
{
public Byte ubBitsPerPixel;
public UInt32 udwMaxLength;
public UInt32 udwActLength;
public SDL_FRAME_ENCODING Encoding;
public UInt16 uwVerticalRes;
public UInt16 uwHorizontalRes;
public byte[] pFrame;
}
```

API

```
[DllImport("MotoScanLib.dll")]
public static extern bool OpenScan();

[DllImport("MotoScanLib.dll")]
public static extern bool CloseScan();

[DllImport("MotoScanLib.dll")]
public static extern bool StartScanDecode();

[DllImport("MotoScanLib.dll")]
public static extern bool StartScanImage();

[DllImport("MotoScanLib.dll")]
public static extern bool TrigerScan(ref SCAN_SCANResult RetVal);

[DllImport("MotoScanLib.dll")]
public static extern bool GetOneImageScan(ref SDL_ImageDescriptor_s RetVal);

[DllImport("MotoScanLib.dll")]
public static extern bool GetLastImageScan(ref SDL_ImageDescriptor_s RetVal);
```