Microvision Scanner SDK for Windows v2.0 Introduction

The SDK for Windows allows developers to quickly and easily integrate bar code scanning into a Windows application. The SDK package includes the C Application Programming Interface (API), ActiveX control, documentation, and sample projects that demonstrate how to create a scanner interface. Sample applications were developed and built using Visual Studio 2005.

The SDK package contains:

BcsSdkWindows	
ReadMeFirst.pdf	*This document*
\BcsSdk	
Bcs.h	*The header file to include when creating an SDK interface*
BcsMsgs.h	*The header file that contains the messages used in the SDK*
Bcs.dll	*The DLL file that containing the SDK API*
Bcs.lib	*Library file to statically link the SDK functions into your project*
BcsCtl.dll	*ActiveX control file*
\Docs	*Documentation directory*
DC0121181 EULA.pdf	*SDK End User License Agreement*
DA0120591 Microvision Scanner Programming Guide.pdf	*Detailed Scanner Programming and Interface Guide that includes scanner commands, communications protocols, and interface best practices*
\Samples	
\BcsActiveX.CS-NET*.*	*C# demo project utilizing the ActiveX control*
\BcsActiveX.VB-NET*.*	*Visual Basic.Net demo project utilizing the ActiveX control*
\BcsActiveX.VB6*.*	*Visual Basic 6 demo project utilizing the ActiveX control*
\BcsBarCode.C*.*	*Demo Project utilizing the native C API*
\BcsBarCode.CS-NET*.*	*C# demo project interfacing with the full API*
\BcsBarCode.VB-NET*.*	*Visual Basic.Net demo project interfacing with the full API*
\BcsConsole.C*.*	*Simple Console Application demonstrating the basic usage of the native C API functions*
\Wtl\wedge*.*	*Advanced Scanner Wedge Application Sample utilizing the C language API and C++ Windows Template Library (WTL)*

Select ActiveX control or C-language API

The SDK for Windows includes an ActiveX control as well as a C API Library. The ActiveX control handles the requirements of most applications and it makes it very easy to create a scanner interface. Use the C API when you need a very fine level of scanner control. The SDK package includes demo projects in a variety of Windows languages. Each project includes fully documented source code with Intellisense comments. Review the interface guidelines in this document and then proceed to examine one of the demo projects.

If you decide to use the C API then you also have the option of using a DLL or statically linking the API library into your application. For the .NET languages (C#, VB) the SDK provides a .NET component which allows your .NET application to access the C functions in the API.

Microvision recommends that you first consider using the ActiveX control. It handles the interface requirements of most applications, it is very simple, and the complete source code interface requires very little code. Review the ActiveX demo projects prior to making a final decision on your implementation. If the ActiveX control does not suit your requirements then use the C API.

Install the DLL

The C API can be accessed using either a statically linked library or a DLL. If you are using the DLL then copy the `Bcs.dll` file to the system directory on your host computer. If you are using the ActiveX control then you must register the control before you can use it.

- Copy the ActiveX `BcsCtl.dll file` to a directory on your host computer. If you have a single scanner project then we recommend copying the ActiveX control to your application directory. If you have multiple scanner applications, then copy the control to a directory common to your applications.

- Get into a DOS prompt by selecting Start → Programs → Accessories → Command Prompt.

- Navigate to the directory that contains the `BcsCtl.dll` file. Register the DLL by entering `Regsvr32 BcsCtl.dll`. You should see a dialog box that tells you that the DLL was registered successfully.

The Scanner Interface

The Microvision SDK for Windows allows you to easily create an interface between your host computer and the ROV scanner. This section explains the integration process. The scanner interface requires each of these steps described below. Please refer to the appropriate demo project for more details.

Enumerating Scanners

The first step in the communication process is creating a connection by discovering and/or enumerating the scanners. The SDK can enumerate new scanners and show details about scanners that are already paired. Normally you use "0000" as the PIN when pairing Bluetooth-enabled scanners.

C SDK: Call `BcsEnumConnections` to discover and enumerate scanners. Press and release the scan button to put the scanner into Discoverable mode (the LED should double-blink repeatedly). `BcsEnumConnections` takes a callback function as an argument. This callback is invoked with each connection. Typical applications use the callback function to create a list of scanners and the user selects the specific scanner to connect to.

ActiveX: The ActiveX control is displayed as a list box that lists the Bluetooth scanners and the valid COM ports for serial scanners. Call `.AddBluetoothDevice` to refresh the device list.

Connecting to the Scanner

After enumerating the scanners you must connect to a specific scanner. Normally, you display a list of scanners from the previous step and select the scanner from this list.

C SDK: Call `BcsOpenConnection` to open a connection to a specific scanner. `BcsOpenConnection` takes an Event Handler callback function as an argument. This Event Handler Callback handles the scanner-related system events.

ActiveX: Call the `.Open` method to open a connection to the scanner highlighted in the list box of the ActiveX control.

Events and Event Handling (SDK Only)

As part of the communication, the SDK generates a number of events in response to scanner data that was sent or received. These events allow the application to monitor, and act upon, the behavior of the connected scanner. The SDK can generate these events:

`eEventFlicBanner`	*Banner received from the scanner*
`eEventDataReceived`	*Data received from the scanner*
`eEventBarCodeDataReceived`	*Bar code data received from scanner*
`eEventCommandPacket`	*Packet that will be sent to the scanner*
`eEventResponsePacket`	*Packet received from scanner*
`eEventBluetoothAttemptingToConnect`	*Trying to connect to a Bluetooth device*
`eEventBluetoothConnectionLost`	*Connection with Bluetooth device lost*
`eEventBluetoothDeviceConnected`	*Bluetooth connection established*

With these event notifications your application can display scanned data, display a live status showing that your Bluetooth scanner is in or out of range, and so on. You can also ignore the events that do not concern your application. For example, if your application does not need to know if a Bluetooth scanner is connected or out of range then simply ignore the Bluetooth connection events.

The Event Handler Callback function passed in the `BcsOpenConnection` call can respond to the desired events. The actions taken by the Event Handler Callback will vary depending on your requirements and on your scanner configuration. The SDK samples demonstrate how to use the Event Handler Callback.

Reading Bar Codes (C API Only)

When the API detects scanner data it generates an `eEventDataReceived` event. Your application should then post a system notification to alert the application to process the bar codes. We normally create a custom notification called `SW_GETBARCODES` that is typically defined as `WM_APP + 0x100`. Your application event loop then calls a bar code reading function after it detects the `SW_GETBARCODES` notification. This function calls `BcsGetBarCodes` to send a download request to the scanner. One of the parameters to `BcsGetBarCodes` is a callback function that is called for each bar code downloaded. If the host receives a batch of bar codes then the callback function is called for each bar code. Your application handles the bar codes one at a time through the callback function.

Reading Bar Codes (ActiveX Only)

Each bar code that arrives at the host generates a `BarCode` event. You then create a callback function that processes the bar code data. Display the events from the property list of the ActiveX control. Select the `BarCode` event and Visual Studio will add a skeleton callback function in your application. Add code in this callback function to process the bar code data. Below is a C# example that formats the bar code data and displays the data in the OutputWindow field:

```
// Display the bar code and the bar code details.
OutputWindow.SelectedText =
      eventArgs.barCode +
      "\t(si: " + eventArgs.symbology +
      ", ts: "  + eventArgs.timeStamp +
      ", tt: "  + eventArgs.timeType  + ")\r\n";
```

Scanner Configuration

The Microvision Scanner Programming Guide contains the complete list of scanner properties and their descriptions. It also contains a list of commands to get and set the scanner properties. The developer can get and/or set properties of a connected scanner in two ways:

- The `BcsCommand` SDK function or the `.Command` ActiveX method sends a command to the scanner. Use `BcsCommand` to get and set scanner properties, send a download request, clear scanner data, etc.

- The `BcsGetProp` / `BcsSetProp` SDK functions or the `.GetProp` / `.SetProp` ActiveX methods get and set a specific scanner property. These require the property name to set or get. The `Set` functions also require the value of the property to set (see the Programming Guide for details).

Disconnect Scanner

Calls the `BcsCloseConnection` SDK function or the ActiveX `.Close` method to terminate the scanner connection.

The BcsActiveX Demo Projects

The SDK includes sample projects that demonstrate how to create an scanner interface application using the ActiveX control. These projects include full source code and are provided for Visual Basic 6 (VB6), Visual Basic.NET (VB.NET), and C#.NET.

You may use any of these demo projects as an example or you may cut and paste source code directly into your project.

The BcsBarCode Demo Projects

The SDK also includes sample projects that demonstrate how to create an scanner interface application using the C language API.

The Wedge Demo Project

The Wedge sample project demonstrates an implementation of a full featured scanner wedge utilizing the C language API. The wedge also utilizes Windows Template Library (WTL), which is a C++ library for developing Windows applications and UI components. It extends ATL (Active Template Library) and provides a set of classes for controls, dialogs, frame windows, etc. The WTL version 8.0 can be downloaded from http://sourceforge.net/projects/wtl/

Support

Contact Microvision Product Support
Telephone: 1(866) 333 3542
Email: scannersupport@microvision.com
Web: http://www.microvision.com/barcode/support