TABLE OF CONTENTS

Customer Support ..........................................................................................................................v
Technical Assistance ...............................................................................................................v
Product Service and Repair .....................................................................................................v
Limited Warranty ......................................................................................................................v

Chapter 1 - Get Started ..............................................................................................................1

About This Manual.........................................................................................................................1
System Requirements ..................................................................................................................1
  Target Operating Systems .......................................................................................................1
Use the SDK................................................................................................................................1
  C++ Programming ..................................................................................................................2
  C# Programming .....................................................................................................................2
Device Detection..........................................................................................................................2

Chapter 2 - Definitions ..............................................................................................................3

DecodeResult .............................................................................................................................3
LanguageOptions........................................................................................................................3
LOG_LEVEL ...............................................................................................................................4
Return Value ..............................................................................................................................4
ScannerInfo.................................................................................................................................4
ScannerStatus ............................................................................................................................5
TextLineType ...............................................................................................................................5
TextColorType.............................................................................................................................6
TextColors ...................................................................................................................................6
TextFontSizes ............................................................................................................................6
Chapter 3 - API .................................................................................................................................9

API List ................................................................................................................................................9

Connect.........................................................................................................................................10
Disconnect....................................................................................................................................10
RegResponseCallback..............................................................................................................11
UnregResponseCallback........................................................................................................11
SetSymbProp................................................................................................................................12
GetSymbProp................................................................................................................................12
DecodeSync..................................................................................................................................13
CancelDecode..............................................................................................................................13
DecodeAsync................................................................................................................................13
SetLogLevel...................................................................................................................................14
SetupWifi .......................................................................................................................................14
setDisplayText..............................................................................................................................15
setDisplayColor.............................................................................................................................15
setTextSize..................................................................................................................................16
EnableNfyBtnPress......................................................................................................................16
EnableNfyBtnPressBarcode..........................................................................................................17
SendMenuCmdSync......................................................................................................................17
ShowStatusAlert..........................................................................................................................18
GetGen7SDKVersion.....................................................................................................................18
SetLanguage................................................................................................................................19
setDisplayColorHex.....................................................................................................................19

Chapter 4 - Callback Events .............................................................................................................21

ResponseCallbackType...............................................................................................................21
ButtonPressFlag..........................................................................................................................21
ResponseCallbackResult............................................................................................................22

Chapter 5 - Sample Code .................................................................................................................23

Connection/Disconnection.............................................................................................................23
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection</td>
<td>23</td>
</tr>
<tr>
<td>Disconnection</td>
<td>23</td>
</tr>
<tr>
<td>Auto Reconnect</td>
<td>24</td>
</tr>
<tr>
<td>Configure Scanner</td>
<td>24</td>
</tr>
<tr>
<td>Pre-Defined Menu Command Parameters</td>
<td>24</td>
</tr>
<tr>
<td>Setup WiFi</td>
<td>24</td>
</tr>
<tr>
<td>Configure Screen Layout</td>
<td>25</td>
</tr>
<tr>
<td>Set Language</td>
<td>25</td>
</tr>
<tr>
<td>Set Display Text</td>
<td>25</td>
</tr>
<tr>
<td>Set Text Color</td>
<td>25</td>
</tr>
<tr>
<td>Set Text Size</td>
<td>25</td>
</tr>
<tr>
<td>Configure Text Properties</td>
<td>25</td>
</tr>
<tr>
<td>Trigger a Scan</td>
<td>26</td>
</tr>
<tr>
<td>Scan Synchronously</td>
<td>26</td>
</tr>
<tr>
<td>Scan Asynchronously</td>
<td>26</td>
</tr>
<tr>
<td>Send Menu Command</td>
<td>26</td>
</tr>
<tr>
<td>Show Alert Popup</td>
<td>26</td>
</tr>
<tr>
<td>Get Version</td>
<td>27</td>
</tr>
<tr>
<td>Handle Button Press Event</td>
<td>27</td>
</tr>
<tr>
<td>Handle Response Callback Events</td>
<td>28</td>
</tr>
</tbody>
</table>
Customer Support

Technical Assistance

To search our knowledge base for a solution or to log in to the Technical Support portal and report a problem, go to www.hsmcontactsupport.com.

For our latest contact information, see www.honeywellaidc.com/locations.

Product Service and Repair

Honeywell International Inc. provides service for all of its products through service centers throughout the world. To obtain warranty or non-warranty service, return your product to Honeywell (postage paid) with a copy of the dated purchase record. To learn more, go to www.honeywellaidc.com and select Service & Repair at the bottom of the page.

Limited Warranty

For warranty information, go to www.honeywellaidc.com and click Get Resources > Product Warranty.
About This Manual

The 8680i Bluetooth Software Development Kit (SDK) provides a set of tools and sample source code to help software developers create Windows® desktop applications for the 8680i Wearable Mini-Mobile Computer using Bluetooth SPP protocol.

The following abbreviations are used in this guide:

- API  Application Programming Interface
- SPP  Serial Port Profile

System Requirements

.Net Framework 4.0 must be on the system.

Target Operating Systems

Microsoft® Windows® 7 and Windows 10, 32 and 64 bit.

Use the SDK

There are four folders inside the installation folder:

- include
- lib
- bin
- samples
C++ Programming

- Add the header files `HonScannerAPI.h`, `HonScannerSettings.h` and `HonScannerStructs.h` from the `include` folder into your application project.

- In the C++ desktop application, link the released lib file `Gen7SDK.lib` under the `lib` folder. `Gen7SDK.lib` has different versions for 32bit and 64bit. Make sure the right version is integrated into your application.

- The `Gen7SDK.dll` is in the `bin` folder and the sample projects are in the `samples` folder.

C# Programming

- Add the `Gen7SDKAssembly.dll` from the `bin` folder into your desktop application project.

Device Detection

The 8680i SDK uses Bluetooth SPP protocol. Refer to the 8680i User Guide for information about connecting to a laptop or tablet.

**Note:** The 8680i can be paired to only one host at a time. You must un-pair the 8680i in order to connect to another host.
The following definitions are in the *HonScannerStructs.h* file.

**DecodeResult**

- Structure.
- Holds the decoded bar code message.

**Decode Result Structure**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>char chCodeID</td>
<td>Honeywell Code ID</td>
</tr>
<tr>
<td>char chAimID</td>
<td>AIM ID, the Symbology Identification</td>
</tr>
<tr>
<td>char chAimModifier</td>
<td>AIM Modifier character</td>
</tr>
<tr>
<td>short nLength</td>
<td>The length of the decode data</td>
</tr>
<tr>
<td>char chMessage[2048]</td>
<td>The decode data buffer</td>
</tr>
</tbody>
</table>

**LanguageOptions**

- Enumeration.
- Sets the language characters.

**Language Options Enumerations**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>loEnglish</td>
<td>English</td>
</tr>
<tr>
<td>loCyrillic</td>
<td>Cyrillic</td>
</tr>
</tbody>
</table>
LOG_LEVEL

• Enumeration.

Log Level Enumerations

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOG_TRACE</td>
<td>Output Trace level log entries.</td>
</tr>
<tr>
<td>LOG_INFO</td>
<td>Output Trace, Information levels log entries.</td>
</tr>
<tr>
<td>LOG_WARNING</td>
<td>Output Trace, Information, Warning levels log entries.</td>
</tr>
<tr>
<td>LOG_ERROR</td>
<td>Output Trace, Information, Warning, Error levels log entries.</td>
</tr>
<tr>
<td>LOG_FATAL</td>
<td>Output Trace, Information, Warning, Error, Fatal levels log entries.</td>
</tr>
<tr>
<td>LOG_NONE</td>
<td>Don’t output any log entries.</td>
</tr>
</tbody>
</table>

Return Value

• Enumeration.
• API function result codes.

Return Values

<table>
<thead>
<tr>
<th>Return Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESULT_INITIALIZE</td>
<td>SDK is not ready.</td>
</tr>
<tr>
<td>RESULT_SUCCESS</td>
<td>Operation was successful.</td>
</tr>
<tr>
<td>RESULT_ERR_DRIVER</td>
<td>Error detected in image engine driver.</td>
</tr>
<tr>
<td>RESULT_ERR_NODECODE</td>
<td>Image engine unable to decode a symbology.</td>
</tr>
<tr>
<td>RESULT_ERR_NOTCONNECTED</td>
<td>Not connected to an engine.</td>
</tr>
<tr>
<td>RESULT_ERR_PARAMETER</td>
<td>One of the function parameters was invalid.</td>
</tr>
<tr>
<td>RESULT_ERR_UNSUPPORTED</td>
<td>The operation was not supported by the engine.</td>
</tr>
<tr>
<td>RESULT_ERR_EXCEPTION</td>
<td>An exception was detected in the engine.</td>
</tr>
</tbody>
</table>

ScannerInfo

• Structure.
• Holds the scanner information.

Scanner Information Structure

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>char chName[128]</td>
<td>The module name of the scanner.</td>
</tr>
<tr>
<td>short nNameLength</td>
<td>The real length of the module name array.</td>
</tr>
<tr>
<td>char chDesc[128]</td>
<td>The description of the scanner.</td>
</tr>
<tr>
<td>short nDescLength</td>
<td>The real length of the description array.</td>
</tr>
</tbody>
</table>
### Scanner Information Structure (continued)

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>chSerialNum[128] The serial number of the scanner.</td>
</tr>
<tr>
<td>short</td>
<td>nSerialNumLength The real length of the serial number array.</td>
</tr>
<tr>
<td>char</td>
<td>chAppVersion[128] The application version of the scanner.</td>
</tr>
<tr>
<td>short</td>
<td>nAppVerLength The real length of the application version array.</td>
</tr>
<tr>
<td>char</td>
<td>chAppDate[64] The application date of the scanner.</td>
</tr>
<tr>
<td>short</td>
<td>nAppDateLength The real length of the application date array.</td>
</tr>
<tr>
<td>char</td>
<td>chAppTime[64] The application time of the scanner.</td>
</tr>
<tr>
<td>short</td>
<td>nAppTimeLength The real length of the application time array.</td>
</tr>
<tr>
<td>char</td>
<td>chBootVersion[128] The boot version of the scanner.</td>
</tr>
<tr>
<td>short</td>
<td>nBootVerLength The real length of the boot version array.</td>
</tr>
<tr>
<td>char</td>
<td>chBootDate[64] The boot date of the scanner.</td>
</tr>
<tr>
<td>short</td>
<td>nBootDateLength The real length of the boot date array.</td>
</tr>
<tr>
<td>char</td>
<td>chBootTime[64] The boot time of the scanner.</td>
</tr>
<tr>
<td>short</td>
<td>nBootTimeLength The real length of the boot time array.</td>
</tr>
<tr>
<td>char</td>
<td>chBluetoothName[128] The Bluetooth name of the scanner.</td>
</tr>
<tr>
<td>short</td>
<td>nBTNameLength The real length of the Bluetooth name array.</td>
</tr>
</tbody>
</table>

### ScannerStatus

- Enumeration.
- Sets alert popup.

#### Scanner Status Enumerations

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssNormal</td>
<td>Don’t show any popup.</td>
</tr>
<tr>
<td>ssBadScan</td>
<td>Show bad scan alert.</td>
</tr>
<tr>
<td>ssGoodScan</td>
<td>Show good scan popup.</td>
</tr>
</tbody>
</table>

### TextLineType

- Enumeration.
- Sets on which line text should be set on the display.

#### Text Line Type Enumerations

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UpLine</td>
<td>Up line.</td>
</tr>
<tr>
<td>BottomLine</td>
<td>Bottom line.</td>
</tr>
</tbody>
</table>

---

8680i SDK User Guide 5
TextColorType

- Enumeration.
- Sets foreground or background color on the display.

Text Color Type Enumerations

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BgColor</td>
<td>Background color</td>
</tr>
<tr>
<td>FgColorUpLine</td>
<td>Foreground color for up line</td>
</tr>
<tr>
<td>FgColorBottomLine</td>
<td>Foreground color for bottom line</td>
</tr>
</tbody>
</table>

TextColors

- Enumeration.
- Sets foreground or background text color.

Text Color Enumerations

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefaultColor</td>
<td>Background color, default is black. Foreground color, default is white.</td>
</tr>
<tr>
<td>Red</td>
<td>Red color.</td>
</tr>
<tr>
<td>Green</td>
<td>Green color.</td>
</tr>
<tr>
<td>Blue</td>
<td>Blue color.</td>
</tr>
</tbody>
</table>

TextFontSizes

- Enumeration.
- Sets text font size.

Text Font Size Enumerations

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>Small size</td>
</tr>
<tr>
<td>Medium</td>
<td>Medium size, default value.</td>
</tr>
<tr>
<td>Large</td>
<td>Large size</td>
</tr>
<tr>
<td>ExtraLarge</td>
<td>Extra Large size</td>
</tr>
<tr>
<td>SmallBold</td>
<td>Small and bold</td>
</tr>
<tr>
<td>MediumBold</td>
<td>Medium and bold</td>
</tr>
<tr>
<td>LargeBold</td>
<td>Large and bold</td>
</tr>
<tr>
<td>ExtraLargeBold</td>
<td>Extra Large and bold</td>
</tr>
<tr>
<td>SingleLineSmall</td>
<td>Small size for single line, only works for the up line</td>
</tr>
<tr>
<td>SingleLineMedium</td>
<td>Medium size for single line, only works for the up line</td>
</tr>
</tbody>
</table>
Text Font Size Enumerations (continued)

| SingleLineLarge | Large size for single line, only works for the up line |

**WifiEncryptType**

- Enumeration.

**WiFi Encryption Type Structure**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>No encryption.</td>
</tr>
<tr>
<td>WEP</td>
<td>WEP mode.</td>
</tr>
<tr>
<td>WPA_WPA2</td>
<td>WPA or WPA2 mixed mode.</td>
</tr>
</tbody>
</table>

**WifiSettings**

- Structure.
- Holds the settings for WiFi setup.

**WiFi Settings Structure**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>bool enableWifi</td>
<td>Wireless Ethernet enable.</td>
<td>Yes</td>
</tr>
<tr>
<td>bool enableDHCP</td>
<td>DHCP server enable. If disabled or fails, fall back on the static addresses.</td>
<td>Yes</td>
</tr>
<tr>
<td>char scannerIPAddress[32]</td>
<td>Scanner IP address. Ignored if DHCP is used.</td>
<td>Yes if DHCP disabled</td>
</tr>
<tr>
<td>char scannerSubnetMask[32]</td>
<td>Scanner subnet mask. Ignored if DHCP is used.</td>
<td>Yes if DHCP disabled</td>
</tr>
<tr>
<td>char scannerDefaultGateway[32]</td>
<td>Scanner default gateway. Ignored if DHCP is used.</td>
<td>Yes if DHCP disabled</td>
</tr>
<tr>
<td>char dnsIPAddress[32]</td>
<td>DNS server address (IP). Empty means no DNS.</td>
<td>No</td>
</tr>
<tr>
<td>char hostIPAddress[32]</td>
<td>Host address you want to connect to (DNS or IP).</td>
<td>Yes</td>
</tr>
<tr>
<td>char hostTcpPortNum[32]</td>
<td>Host TCP port number.</td>
<td>Yes</td>
</tr>
<tr>
<td>char ssid[128]</td>
<td>SSID (service set identifier). WiFi name.</td>
<td>Yes</td>
</tr>
<tr>
<td>WifiEncryptType encryptType</td>
<td>SSID encryption type.</td>
<td>Yes</td>
</tr>
<tr>
<td>char password[128]</td>
<td>SSID encryption key.</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Windows Native C/C++ APIs are listed below. They have the same return value as defined in Return Value (page 4). Additional information about APIs is available in the HonScannerAPI.h file.

### API List

<table>
<thead>
<tr>
<th>API</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect</td>
<td>Enumerate paired Bluetooth scanners and create connection if one is found.</td>
</tr>
<tr>
<td>Disconnect</td>
<td>Disconnect from the connected scanner.</td>
</tr>
<tr>
<td>SetSymbProp</td>
<td>Configure the symbology settings.</td>
</tr>
<tr>
<td>GetSymbProp</td>
<td>Get the symbology configurations.</td>
</tr>
<tr>
<td>DecodeSync</td>
<td>Trigger a synchronous scan and wait for the decoded data.</td>
</tr>
<tr>
<td>DecodeAsync</td>
<td>Trigger an asynchronous scan and don’t wait for the result.</td>
</tr>
<tr>
<td>CancelDecode</td>
<td>Cancel a synchronous scan.</td>
</tr>
<tr>
<td>RegResponseCallback</td>
<td>Register a Callback in SDK, SDK forwards all events to it.</td>
</tr>
<tr>
<td>UnregResponseCallback</td>
<td>Unregister the Callback.</td>
</tr>
<tr>
<td>SetLogLevel</td>
<td>Set the log level for outputting the specified log information.</td>
</tr>
<tr>
<td>SetupWifi</td>
<td>Enable\Disable and set the settings of WiFi through sending a menu command asynchronously.</td>
</tr>
<tr>
<td>SetDisplayText</td>
<td>Set the text content to display on the scanner screen through sending a menu command asynchronously.</td>
</tr>
<tr>
<td>SetDisplayColor</td>
<td>Set the background or foreground color of the text on the scanner screen through sending a menu command asynchronously.</td>
</tr>
</tbody>
</table>
Connect

The application searches for a valid scanner from the Bluetooth serial ports. This API automatically connects to a Bluetooth scanner if it is found.

Parameters
N.A.

Return Value
RESULT_SUCCESS if a valid scanner is found and successfully connected.
RESULT_INITIALIZE if the SDK is not initialized successfully.
RESULT_ERR_NOTCONNECTED if a valid scanner is not found or no connection can be made to a scanner that has been found.

Note: Invoke RegResponseCallback before invoking Connect so that the SDK can receive the connected callback event and retrieve the connected scanner information.

Disconnect

Disconnect from the scanner.
Parameters
N.A.

Return Value
RESULT_SUCCESS if the scanner is successfully disconnected.
RESULT_INITIALIZE if the SDK is not successfully initialized.

Note: Invoke UnregResponseCallback before invoking Disconnect so that the SDK can stop to receive callback events before releasing the connection resources.

RegResponseCallback
This API allows the application to register a callback to receive the events from the SDK such as bar code responses, disconnect events, button press events, and other responses from the scanner.

Parameters
ResponseCallback resCb
The function pointer that will receive the events from the SDK layer.

typedef void (*ResponseCallback)(const ResponseCallbackResult &respCallbackRes);

Return Value
RESULT_SUCCESS if successfully registered.
RESULT_INITIALIZE if the SDK is not successfully initialized.

Note: Invoke this API before invoking Connect.

UnregResponseCallback
This is used to stop receiving events from the SDK layer and must be called before closing the application.

Parameters
N.A.

Return Value
RESULT_SUCCESS if successfully unregistered.
RESULT_INITIALIZE if the SDK is not successfully initialized.

Note: Invoke this API before invoking Disconnect.
SetSymbProp

Set the symbol code property in the scanner by sending an asynchronous menu command.

Parameters

**unsigned long symbolCodeID**
Indicates the symbol code properties found in **HonScannerSettings.h**.

**unsigned long value**
The new value for the property to set.

Return Value

RESULT_SUCCESS if the menu command is successfully sent.
RESULT_INITIALIZE if the SDK is not successfully initialized.
RESULT_ERR_NOTCONNECTED if the SDK doesn’t connect to a scanner.
RESULT_ERR_PARAMETER if the **symbolCodeID** is not correct.
RESULT_ERR_EXCEPTION if the menu command fails.

GetSymbProp

Get the symbol code property from the scanner by sending a synchronous menu command.

Parameters

**unsigned long symbolCodeID**
Indicates the symbol code properties found in **HonScannerSettings.h**.

**void* pValue**
Pointer to receive the specified symbol code property.

Return Value

RESULT_SUCCESS if the property is successfully retrieved.
RESULT_INITIALIZE if the SDK is not successfully initialized.
RESULT_ERR_NOTCONNECTED if the SDK doesn’t connect to a scanner.
RESULT_ERR_PARAMETER if the **symbolCodeID** is not correct or the **pValue** pointer is null.
RESULT_ERR_EXCEPTION if the menu command fails.
DecodeSync

Trigger a scan and wait for the scan results.

Parameters

DecodeResult* pDecResult

Pointer for receiving the decoded data. See DecodeResult on page 3 for more details.

unsigned long | TimeOut

This is the timeout, in milliseconds, for synchronous decoding.

Return Value

RESULT_SUCCESS if successfully decoded.
RESULT_INITIALIZE if the SDK is not successfully initialized.
RESULT_ERR_NOTCONNECTED if the SDK doesn’t connect to a scanner.
RESULT_ERR_PARAMETER if the pDecResult pointer is null.
RESULT_ERR_EXCEPTION if the menu command for scanning fails.
RESULT_ERR_NODECODE if no bar code is scanned before the timeout.

CancelDecode

Cancel the decode action in process.

Parameters

N.A.

Return Value

RESULT_SUCCESS if the decode is successfully canceled.
RESULT_INITIALIZE if the SDK is not successfully initialized.
RESULT_ERR_NOTCONNECTED if the SDK doesn’t connect to a scanner.
RESULT_ERR_EXCEPTION if the menu command for canceling the decode fails.

DecodeAsync

Trigger an asynchronous scan and return the scan result with a callback event.
Parameters
N.A.

Return Value
RESULT_SUCCESS if the decode menu command is successfully sent.
RESULT_INITIALIZE if the SDK is not successfully initialized.
RESULT_ERR_NOTCONNECTED if the SDK doesn’t connect to a scanner.
RESULT_ERR_EXCEPTION if the menu command for scanning fails.

SetLogLevel
Set the log level for the specified log entries output.

Parameters
LOG_LEVEL level
Indicates the log level. Trace < Info < Warning < Error < Fatal.
See LOG_LEVEL on page 4 for more details.

Return Value
N.A.

Note: This API doesn’t rely on the connection to scanner, so you can invoke it before invoking RegResponseCallback.

SetupWifi
Enable and set the WiFi settings. Disable the WiFi by sending a series of asynchronous menu commands.

Parameters
WifiSettings settings
The structure includes WiFi settings. See WifiSettings on page 7 for more details.

Return Value
RESULT_SUCCESS if the menu command for WiFi setup is successfully sent.
RESULT_INITIALIZE if the SDK is not successfully initialized.
RESULT_ERR_NOTCONNECTED if the SDK doesn’t connect to a scanner.
RESULT_ERR_PARAMETER if the IP addresses are invalid. Expect IPV4 to be a dotted decimal (for example, 192.168.0.1) and IPV6 to use the long form (xxxx:xxxx:xxxx:xxxx).

RESULT_ERR_DRIVER if any menu command for WiFi settings fails.

SetDisplayText

Set the text content to display on the scanner screen by sending an asynchronous menu command.

Parameters

**TextLineType whichLine**
Enumerations: UpLine, BottomLine. See TextLineType on page 5 for more details.

*const wchar_t* text
The text to display. Supports Unicode.

Return Value

RESULT_SUCCESS if the menu command for setting display text is successfully sent.

RESULT_INITIALIZE if the SDK is not successfully initialized.

RESULT_ERR_NOTCONNECTED if the SDK doesn’t connect to a scanner.

RESULT_ERR_PARAMETER if the text is null.

RESULT_ERR_DRIVER if the menu command for setting display text fails.

SetDisplayColor

Set the background or foreground color of the text on the scanner screen by sending an asynchronous menu command.

Parameters

**TextColorType colorType**
Enumerations: BgColor, FgColorUpLine, FgColorBottomLine. See TextColorType on page 6 for more details.

**TextColors color**
Enumerations: DefaultColor, Red, Green, Blue. See TextColors on page 6 for more details.
Return Value
RESULT_SUCCESS if the menu command for setting text color is successfully sent.
RESULT_INITIALIZE if the SDK is not successfully initialized.
RESULT_ERR_NOTCONNECTED if the SDK doesn’t connect to a scanner.
RESULT_ERR_DRIVER if the menu command for setting text color fails.

SetTextSize
Set the text font size on the scanner screen by sending an asynchronous menu command.

Parameters
TextLineType whichLine
Enumerations: UpLine, BottomLine. See TextLineType on page 5 for more details.

TextFontSizes fontSize
Enumerations: See TextFontSizes on page 6 for complete details.

Return Value
RESULT_SUCCESS if the menu command for setting text font size is successfully sent.
RESULT_INITIALIZE if the SDK is not successfully initialized.
RESULT_ERR_NOTCONNECTED if the SDK doesn’t connect to a scanner.
RESULT_ERR_DRIVER if the menu command for setting text font size fails.

EnableNfyBtnPress
Make the scanner send a notification to the host when one or both of the scanner buttons are pressed. This is done by sending an asynchronous menu command.

Parameters
bool enable
True or false.

Return Value
RESULT_SUCCESS if the menu command for the enable/disable button press notification is successfully sent.
RESULT_INITIALIZED if the SDK is not successfully initialized.
RESULT_ERR_NOTCONNECTED if the SDK doesn’t connect to a scanner.
RESULT_ERR_DRIVER if the menu command for enabling button press notifications fails.

**Note:** If you want to receive the button pressed callback event, you should invoke this API after invoking `Connect`.

### EnableNfyBtnPressBarcode

Make the scanner send a notification to the host when one or both of the scanner buttons are pressed and bar code data is sent. This is done by sending an asynchronous menu command.

**Parameters**

N.A.

**Return Value**

RESULT_SUCCESS if the menu command for enable notifications is successfully sent.
RESULT_INITIALIZED if the SDK is not successfully initialized.
RESULT_ERR_NOTCONNECTED if the SDK doesn’t connect to a scanner.
RESULT_ERR_DRIVER if the menu command for enabling button press notifications and bar code data transmission fails.

### SendMenuCmdSync

Send the raw menu command to the scanner by a synchronous command.

**Parameters**

- **const char* cmd**
  The raw text of the menu command.

  Add the command prefix, such as SYN_M or SYN_Y and the command suffix, such as RAM(!) or ROM(.).

  This function sends a series of commands with separator (;) such as EA8ENA1;C39ENA1;128ENA1. The length should not be larger than 128 characters.

- **unsigned long lTimeOut**
  The timeout for the synchronous menu command execution, in milliseconds.
**char* retData**
The raw returned data for menu command.

**int* retSize**

- **[In]** The max size of the returned data array to pass in.
- **[Out]** The real size of the returned data array. This may not be larger than the max size passed in.

**Return Value**
RESULT_SUCCESS if the raw menu command is successfully sent.
RESULT_INITIALIZE if the SDK is not successfully initialized.
RESULT_ERR_NOTCONNECTED if the SDK doesn’t connect to a scanner.
RESULT_ERR_PARAMETER if the command is null or its length is not in the valid range (1 – 128).
RESULT_ERR_EXCEPTION if the menu command fails.

**ShowStatusAlert**
Show the status image on the scanner screen by an asynchronous menu command.

**Parameters**

- **ScannerStatus status**
The scanner status. Such as good scan, bad scan. See ScannerStatus on page 5.

**Return Value**
RESULT_SUCCESS if the menu command is successfully executed.
RESULT_INITIALIZE if the SDK is not successfully initialized.
RESULT_ERR_NOTCONNECTED if the SDK doesn’t connect to a scanner.
RESULT_ERR_EXCEPTION if the menu command fails.

**GetGen7SDKVersion**
Get the current version of the SDK.
Parameters

char* version
The array for receiving the SDK version.

int* verSize

[In]  The max size of the returned version array to pass in.

[Out] The real size of the returned version array. This may not be larger than the max size passed in.

Return Value
RESULT_SUCCESS if the SDK version is successfully retrieved.
RESULT_ERR_PARAMETER if the version or verSize is null.
RESULT_ERR_EXCEPTION if the version retrieval fails.

SetLanguage

Parameters

LanguageOptions option
The language options. See LanguageOptions on page 3.

Return Value
RESULT_SUCCESS if the menu command is successfully executed.
RESULT_INITIALIZE if the SDK is not successfully initialized.
RESULT_ERR_NOTCONNECTED if the SDK doesn’t connect to a scanner.
RESULT_ERR_EXCEPTION if the menu command fails.

SetDisplayColorHex

Set the background or foreground color of the text on the scanner screen by sending an asynchronous menu command.

Parameters

TextColorType colorType
Enumerations: BgColor, FgColorUpLine, FgColorBottomLine. See TextColorType on page 6 for more details.

const char* hexColor
The RGB hex code string.
Return Value

RESULT_SUCCESS if the menu command for setting text color is successfully sent.
RESULT_INITIALIZE if the SDK is not successfully initialized.
RESULT_ERR_NOTCONNECTED if the SDK doesn’t connect to a scanner.
RESULT_ERR_DRIVER if the menu command for setting text color fails.
CHAPTER 4
CALLBACK EVENTS

ResponseCallbackType

- Enumeration.

Response Callback Type Enumerations

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rctUnknown</td>
<td>Unknown type.</td>
</tr>
<tr>
<td>rctConnected</td>
<td>The event is sent when the scanner is connected.</td>
</tr>
<tr>
<td>rctDisconnected</td>
<td>The event is sent when the connection to the scanner is lost, for example, if the scanner is far away from the PC or laptop. Invoking Disconnect won’t send this event. See Auto Reconnect on page 24 for more details.</td>
</tr>
<tr>
<td>rctDecodeCompleted</td>
<td>The event is sent when asynchronous scanning is successful.</td>
</tr>
<tr>
<td>rctMenuCmdResponsed</td>
<td>The event is sent when an asynchronous menu command is executed.</td>
</tr>
<tr>
<td>rctButtonPressed</td>
<td>The event is sent when one or both of the scanner buttons are pressed.</td>
</tr>
</tbody>
</table>

ButtonPressFlag

- Enumeration.

Button Press Flag Enumerations

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NoButtonPressed</td>
<td>No button is pressed.</td>
</tr>
<tr>
<td>LeftButtonPressed</td>
<td>Left button is pressed.</td>
</tr>
<tr>
<td>RightButtonPressed</td>
<td>Right button is pressed.</td>
</tr>
<tr>
<td>BothButtonsPressed</td>
<td>Left and right buttons are pressed.</td>
</tr>
</tbody>
</table>
ResponseCallbackResult

- Structure.

Holds the information of events.

Response Callback Result Structure

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResponseCallbackType</td>
<td>Callback event type.</td>
</tr>
<tr>
<td>respCallbackType</td>
<td></td>
</tr>
<tr>
<td>DecodeResult</td>
<td>The bar code data for asynchronous scanning. When the callback event type is rctDecodeCompleted, this structure will be filled with data.</td>
</tr>
<tr>
<td>decodeResult</td>
<td></td>
</tr>
<tr>
<td>ScannerInfo</td>
<td>The information for the connected scanner. When the callback event type is rctConnected, this structure will be filled with data.</td>
</tr>
<tr>
<td>scannerInfo</td>
<td></td>
</tr>
<tr>
<td>bool</td>
<td>The flag indicates whether the asynchronous menu command is executed successfully when the callback event type is rctMenuCmdResponded.</td>
</tr>
<tr>
<td>isMenuCmdExecutedSuccessful</td>
<td></td>
</tr>
<tr>
<td>ButtonPressFlag</td>
<td>The flag indicates which button is pressed when the callback event type is rctButtonPressed.</td>
</tr>
<tr>
<td>whichButtonPressed</td>
<td></td>
</tr>
</tbody>
</table>
Connection/Disconnection

Connection

// Logging doesn’t rely on the Connection, so we can initialize the level at the beginning.
SetLogLevel(LOG_TRACE);

// Register callback function before invoking Connect, so that we can receive connected event to get scanner information in the callback function.
RegResponseCallback(SdkResponseCallback);

Result_t res = Connect();
If (res == RESULT_SUCCESS)
{
    // Enable to receive button pressed callback events once connect to the scanner.
    // Otherwise you can invoke this API wherever as you need.
    EnableNfyBtnPress(true);
    // Do other things.
}

Disconnection

UnregResponseCallback(); // Unregister callback function before invoking Disconnect.

Result_t res = Disconnect();
If (res == RESULT_SUCCESS)
{
    //
}
Auto Reconnect

If you invoke Connect API and it returns RESULT_SUCCESS, a thread in the SDK starts to monitor the connection between the host (PC or laptop) and the scanner. If the host loses the connection with the scanner, the disconnected callback event is sent to host. Another reconnecting thread is then started to try to reconnect to the scanner.

If you have connected to the scanner by successfully invoking Connect, the SDK handles the reconnection automatically. If you can’t connect to the scanner, invoke Connect again.

Configure Scanner

Pre-Defined Menu Command Parameters

SetSymbProp(DEC_EAN8_ENABLED, 1); // Enable EAN-8
SetSymbProp(DEC_EAN8_CHECKDIGIT_TRANSMIT, 1);
SetSymbProp(DEC_EAN8_2CHAR_ADDENDA_ENABLED, 0); // Disable 2 char addenda
SetSymbProp(DEC_CODE128_MIN_LENGTH, 5); // Set the minimum length of code 128 to be 5

Setup WiFi

// Disable WI-FI usage
WifiSettings setting;
setting.enableWifi = false;
SetupWifi(setting);

// Enable WI-FI usage
WifiSettings setting;
setting.enableWifi = true;
// Disable DHCP. If enable, scanner IP, default gateway and subnet mask are ignored.
setting.enableDHCP = false;
setting.scannerIPAddress = "192.168.1.15";
setting.scannerDefaultGateway = "192.168.1.1";
setting.scannerSubnetMask = "255.255.255.0";
setting.dnsIPAddress = "192.168.1.1";
setting.hostIPAdrrss = "192.168.1.1";
setting.hostTcpPortNum = "8080";
setting.ssid = "TestWifi";
setting.encryptType = WPA_WPA2;
setting.password = "xxxxxxxxxx";
SetupWifi(setting);
Configure Screen Layout

Set Language

SetLanguage(loCyrillic);  // Set the scanner to be ready to show Cyrillic characters.

Set Display Text

SetDisplayText(UpLine, "Welcome");  // Show ‘Welcome’ at the up line
SetDisplayText(BottomLine, "Bad Code"); // Show ‘Bad Code’ at the bottom line

Set Text Color

SetDisplayColor(BgColor, Red);  // Show the background color in red
SetDisplayColor(FgColorUpLine, Green); // Show the foreground color of the up line in green

Set Text Size

SetTextSize(UpLine, Large);  // Show the text in large size at the up line
SetTextSize(BottomLine, Small); // Show the text in small size at the bottom line

Configure Text Properties

// Show ‘Welcome’ in large size with green foreground and red background at the up line
SetDisplayColor(BgColor, Red);
SetDisplayColor(FgColorUpLine, Green);
SetTextSize(UpLine, Large);
SetDisplayText(UpLine, “Welcome”);

// Show ‘Bad Code’ in small size with blue foreground at the bottom line
SetDisplayColor(FgColorBottomLine, Blue);
SetTextSize(BottomLine, Small);
SetDisplayText(BottomLine, “Bad Code”);
Trigger a Scan

Scan Synchronously

```
DecodeResult decRes;
memset(&decRes, 0, sizeof(DecodeResult)); // Initialize the structure to receive the decoded data
Result_t res = DecodeSync(decRes, 5000);
if (res == RESULT_SUCCESS)
    OutputDecodeResult(decRes);
```

Scan Asynchronously

```
DecodeAsync(); // Send scan command
// Retrieve the scan result in the callback function
void SdkResponseCallback(const ResponseCallbackResult &respCallbackRes)
{
    switch(respCallbackRes.respCallbackType)
    {
        case rctDecodeCompleted:
        {
            OutputDecodeResult(respCallbackRes.decResult);
            break;
        }
    }
}
```

Send Menu Command

```
#define CMD_SYN_M   "\x16\x4d\x0d"
#define CMD_RAM      "\x21" // !

std::string cmd = "EA8ENA?";
std::string cmd = CMD_SYN_M + cmd + CMD_RAM;
int retSize = 1024;
char retData[1024];
Result_t res = SendMenuCmdSync(cmd.c_str(), 2000, retData, &retSize);
if(res == RESULT_SUCCESS)
    Log(CString(retData, retSize));
```

Show Alert Popup

```
ShowStatusAlert(ssGoodScan);
```
Get Version

```c
char version[20];
int verLen = 20;
Result_t res = GetGen7SDKVersion(version, &verLen);
if (res == RESULT_SUCCESS)
    Log(CString(version, verLen));
```

Handle Button Press Event

```c
// Should enable this functionality first, so we can receive button pressed events.
EnableNfyBtnPress(true);
// Handle the button press event in the callback function
void SdkResponseCallback(ResponseCallbackResult &respCallbackRes)
{
    switch(respCallbackRes.respCallbackType)
    {
        case rctButtonPressed:
            switch(respCallbackRes.whichButtonPressed)
            {
                case LeftButtonPressed:
                    // Do something when left button is pressed
                    break;
                case RightButtonPressed:
                    // Do something when right button is pressed
                    break;
            }
        break;
    }
}
```
void SdkResponseCallback(const ResponseCallbackResult &respCallbackRes)
{
  switch(respCallbackRes.respCallbackType)
  {
    case rctConnected:
    {
      CString str(respCallbackRes.scannerInfo.chBluetoothName);
      str.Append(" is connected");
      Log(str);
      break;
    }
    case rctDisconnected:
    {
      Log("Scanner is disconnected");
      break;
    }
    case rctDecodeCompleted:
    {
      LogDecodeResult(respCallbackRes.decResult));
      break;
    }
    case rctMenuCmdResponsed:
    {
      CString str("Menu command is executed ");
      str.Append(respCallbackRes.isMenuCmdExecutedSuccessful ? "successfully" : "unsuccessfully");
      Log(str);
      break;
    }
    case rctButtonPressed:
    {
      CString str;
      switch(respCallbackRes.whichButtonPressed)
      {
        case LeftButtonPressed:
        str = "Left button is ";
        break;
        case RightButtonPressed:
        str = "Right button is ";
        break;
      }
case BothButtonsPressed:
    str = "Left and right buttons are ";
    break;

case NoButtonPressed:
    default:
        str = "No button is ";
        break;

}