Honeywell

THE POWER OF **CONNECTED**

8680i

Bluetooth Software Development Kit

User Guide

Disclaimer

Honeywell International Inc. (“HII”) reserves the right to make changes in specifications and other information contained in this document without prior notice, and the reader should in all cases consult HII to determine whether any such changes have been made. The information in this publication does not represent a commitment on the part of HII.

HII shall not be liable for technical or editorial errors or omissions contained herein; nor for incidental or consequential damages resulting from the furnishing, performance, or use of this material. HII disclaims all responsibility for the selection and use of software and/or hardware to achieve intended results.

Web Address: www.honeywellaidc.com

For patent information, refer to www.hsmpats.com.

TABLE OF CONTENTS

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**.

CHAPTER

1 GET STARTED

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.*

CHAPTER

2 DEFINITIONS

The following definitions are in the **HonScannerStructs.h** file.

DecodeResult

- Structure.
- Holds the decoded bar code message.

Decode Result Structure

Field		Description
char	chCodeID	Honeywell Code ID
char	chAimID	AIM ID, the Symbology Identification
char	chAimModifier	AIM Modifier character
short	nLength	The length of the decode data
char	chMessage[2048]	The decode data buffer

LanguageOptions

- Enumeration.
- Sets the language characters.

Language Options Enumerations

Value	Description
loEnglish	English
loCyrillic	Cyrillic

LOG_LEVEL

- Enumeration.

Log Level Enumerations

Value	Description
LOG_TRACE	Output Trace level log entries.
LOG_INFO	Output Trace, Information levels log entries.
LOG_WARNING	Output Trace, Information, Warning levels log entries.
LOG_ERROR	Output Trace, Information, Warning, Error levels log entries.
LOG_FATAL	Output Trace, Information, Warning, Error, Fatal levels log entries.
LOG_NONE	Don't output any log entries.

Return Value

- Enumeration.
- API function result codes.

Return Values

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

ScannerInfo

- Structure.
- Holds the scanner information.

Scanner Information Structure

Field		Description
char	chName[128]	The module name of the scanner.
short	nNameLength	The real length of the module name array.
char	chDesc[128]	The description of the scanner.
short	nDescLength	The real length of the description array.

Scanner Information Structure (continued)

char	chSerialNum[128]	The serial number of the scanner.
short	nSerialNumLength	The real length of the serial number array.
char	chAppVersion[128]	The application version of the scanner.
short	nAppVerLength	The real length of the application version array.
char	chAppDate[64]	The application date of the scanner.
short	nAppDateLength	The real length of the application date array.
char	chAppTime[64]	The application time of the scanner.
short	nAppTimeLength	The real length of the application time array.
char	chBootVersion[128]	The boot version of the scanner.
short	nBootVerLength	The real length of the boot version array.
char	chBootDate[64]	The boot date of the scanner.
short	nBootDateLength	The real length of the boot date array.
char	chBootTime[64]	The boot time of the scanner.
short	nBootTimeLength	The real length of the boot time array.
char	chBluetoothName[128]	The Bluetooth name of the scanner.
short	nBTNameLength	The real length of the Bluetooth name array.

ScannerStatus

- Enumeration.
- Sets alert popup.

Scanner Status Enumerations

Value	Description
ssNormal	Don't show any popup.
ssBadScan	Show bad scan alert.
ssGoodScan	Show good scan popup.

TextLineType

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

Text Line Type Enumerations

Value	Description
UpLine	Up line.
BottomLine	Bottom line.

TextColorType

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

Text Color Type Enumerations

Value	Description
BgColor	Background color
FgColorUpLine	Foreground color for up line
FgColorBottomLine	Foreground color for bottom line

TextColors

- Enumeration.
- Sets foreground or background text color.

Text Color Enumerations

Value	Description
DefaultColor	Background color, default is black. Foreground color, default is white.
Red	Red color.
Green	Green color.
Blue	Blue color.

TextFontSizes

- Enumeration.
- Sets text font size.

Text Font Size Enumerations

Value	Description
Small	Small size
Medium	Medium size, default value.
Large	Large size
ExtraLarge	Extra Large size
SmallBold	Small and bold
MediumBold	Medium and bold
LargeBold	Large and bold
ExtraLargeBold	Extra Large and bold
SingleLineSmall	Small size for single line, only works for the up line
SingleLineMedium	Medium size for single line, only works for the up line

Text Font Size Enumerations (continued)

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

WifiEncryptType

- Enumeration.

WiFi Encryption Type Structure

Value	Description
Open	No encryption.
WEP	WEP mode.
WPA_WPA2	WPA or WPA2 mixed mode.

WifiSettings

- Structure.
- Holds the settings for WiFi setup.

WiFi Settings Structure

Field	Description	Required
bool enableWifi	Wireless Ethernet enable.	Yes
bool enableDHCP	DHCP server enable. If disabled or fails, fall back on the static addresses.	Yes
char scannerIPAddress[32]	Scanner IP address. Ignored if DHCP is used.	Yes if DHCP disabled
char scannerSubnetMask[32]	Scanner subnet mask. Ignored if DHCP is used.	Yes if DHCP disabled
char scannerDefaultGateway[32]	Scanner default gateway. Ignored if DHCP is used.	Yes if DHCP disabled
char dnsIPAddress[32]	DNS server address (IP). Empty means no DNS.	No
char hostIPAddress[32]	Host address you want to connect to (DNS or IP).	Yes
char hostTcpPortNum[32]	Host TCP port number.	Yes
char ssid[128]	SSID (service set identifier), WiFi name.	Yes
WifiEncryptType encryptType	SSID encryption type.	Yes
char password[128]	SSID encryption key.	Yes

CHAPTER

3 API

API List

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

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

API List (continued)

SetTextSize	Set the text font size on the scanner screen through sending a menu command asynchronously.
EnableNfyBtnPress	Enable the scanner to send notification to the host when the scanner buttons are pressed through sending a menu command asynchronously.
EnableNfyBtnPressBarcode	Enable the scanners to send notifications of button presses and bar code data to the host through sending a menu command asynchronously.
SendMenuCmdSync	Send the raw menu command to the scanner synchronously.
ShowStatusAlert	Show the status screen image on the scanner by sending a menu command asynchronously.
GetGen7SDKVersion	Get the current version of the SDK.
SetLanguage	Set which language characters are displayed on the scanner.
SetDisplayColorHex	Set the background or foreground color of the text on the scanner screen with RGB hex code through sending a menu command asynchronously.

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_SUCCES 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_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 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_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 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

Value	Description
rctUnknown	Unknown type.
rctConnected	The event is sent when the scanner is connected.
rctDisconnected	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.
rctDecodeCompleted	The event is sent when asynchronous scanning is successful.
rctMenuCmdResponsed	The event is sent when an asynchronous menu command is executed.
rctButtonPressed	The event is sent when one or both of the scanner buttons are pressed.

ButtonPressFlag

- Enumeration.

Button Press Flag Enumerations

Value	Description
NoButtonPressed	No button is pressed.
LeftButtonPressed	Left button is pressed.
RightButtonPressed	Right button is pressed.
BothButtonsPressed	Left and right buttons are pressed.

ResponseCallbackResult

- Structure.

Holds the information of events.

Response Callback Result Structure

Field	Description
ResponseCallbackType respCallbackType	Callback event type.
DecodeResult decResult	The bar code data for asynchronous scanning. When the callback event type is rctDecodeCompleted, this structure will be filled with data.
ScannerInfo scannerInfo	The information for the connected scanner. When the callback event type is rctConnected, this structure will be filled with data.
bool isMenuCmdExecutedSuccessful	The flag indicates whether the asynchronous menu command is executed successfully when the callback event type is rctMenuCmdResponsed.
ButtonPressFlag whichButtonPressed	The flag indicates which button is pressed when the callback event type is rctButtonPressed.

CHAPTER

5 SAMPLE CODE

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_CHECK_DIGIT_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.hostIPAddress = "192.168.1.1";
setting.hostTcpPortNum = "8080";
setting.ssid = "TestWifi";
setting.encryptType = WPA_WPA2;
setting.password = "xxxxxxxxx";
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?”;
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

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

Handle Button Press Event

```
// 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;
    }
}
```

Handle Response Callback Events

```
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;
            }
            str.Append("Pressed");
            Log(str);
            break;
        }
    }
}
```

Honeywell Scanning & Mobility
9680 Old Bailes Road
Fort Mill, SC 29707

www.honeywellaidc.com

8680SDK-EN-UG Rev B
7/18