



Prepared by  
**Anders Hagberg**  
Contents responsible if other than preparer

Document Revision  
**P121113-1UG002-v0.3**  
Date  
**2014-08-11**  
Remarks

Approved by

---

## H-27 System Integrator User Guide

1	Overview .....	2
2	H-27 Development Prerequisites .....	2
3	H-27 Specific APIs .....	2
4	Qualcomm APIs .....	12



## 1 Overview

The H-27 SDK consists of several parts. The Google Android 4.2 SDK, a few H-27 specific APIs, and Qualcomms Android SDK.

## 2 H-27 Development Prerequisites

1. Download and install Android SDK (<http://developer.android.com/sdk/index.html>)
2. Download Opticon H-27 SDK libs
3. Install ABD driver (please see *P121113-1UG001-v0.1 H27 Enterprise Administrator User Guide.docx*)
4. Enable USB debugging on H-27 (<http://www.wugfresh.com/faq/6/>)

## 3 H-27 Specific APIs

### 3.1 Barcode Reader API

#### 3.1.1 Java library

.libs\com.oem.barcode.sdk.jar

#### 3.1.2 Intent Mode

This is the recommended mode for developing a solution/App which uses the H-27 barcode reader. Intent mode extends Buffer Mode with support for getting an Intent when a barcode is scanned. The intent will contain

- Raw barcode data
- Type of barcode
- Barcode data as a string decoded in a best effort manner. I.e., in case the charset encoding cannot be determined, a best guess algorithm is used to decode the raw barcode data. ECI is handled in this algorithm.
- Encoding of barcode. Detected or guessed encoding used to decode the barcode data into a string.

##### 3.1.2.1 Intent Mode Example

1. In Settings->Barcode reader->Output mode select **Intent mode**
2. Create a new project by File->New->Android Application Project
  - a. Use Android 4.2 API
3. Add barcode reader library by right clicking the root of the new Android project and select Import...->File System
4. For the "From directory" browse to the location of the barcode reader library
5. Check the checkbox for com.oem.barcodeV2.4.jar
6. Open the Android projects AndroidManifest.xml and add these lines before the `<application>` XML tag

```
<permission android:protectionLevel="normal"
            android:name="android.permission.BCR" />
<uses-permission android:name="android.permission.RECEIVE_SCANNER" />
```

7. Open MainActivity.java

8. Add this code at the top of the MainActivity class

```
final BroadcastReceiver mBroadcastReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        if (intent != null) {
            final String action = intent.getAction();
            if (BCRIntents.ACTION_NEW_DATA.equals(action)){
                byte[] data =
                    intent.getByteArrayExtra(BCRIntents.EXTRA_BCR_DATA);

                int type =
                    intent.getIntExtra(BCRIntents.EXTRA_BCR_TYPE, -1);

                String decodedBarcode =
                    intent.getStringExtra(BCRIntents.EXTRA_BCR_STRING);

                String charset =
                    intent.getStringExtra(BCRIntents.EXTRA_BCR_CHARSET);

                // Add your handling of the barcode data here...
            }
        }
    }
};
```

9. Add these lines in the end of the method **protected void** onCreate(Bundle savedInstanceState):

```
IntentFilter filter = new IntentFilter();
filter.addAction(BCRIntents.ACTION_NEW_DATA);
this.registerReceiver(mBroadcastReceiver, filter);
```

10. Add this method in order to clean up when the MainActivity class is destroyed

```
@Override
public void onDestroy()
{
    this.unregisterReceiver(mBroadcastReceiver);
    super.onDestroy();
}
```

### 3.1.2.2 Sample Code

See

- `.\samples - src\BarcodeReaderIntentReceiver.zip`
- `.\samples - APKs\BarcodeReaderIntentReceiver.apk`

### 3.1.3 Buffer Mode

JAVA API to get full control of the barcode reader. In this mode the developer need to decode and handle the result of a scanned barcode (i.e., the result of `BCRReadBarcode()`). The following methods are exposed:

```
public int BCRSetDefault()
public int BCRSetOutputMode(int mode)
public int BCRGetOutputMode()
public int BCRTriggerPress()
public int BCRTriggerRelease()
public int BCRSetBuzzerStatus(Context context,int buzzer)
public int BCRGetBuzzerStatus(Context context)
public byte[] BCRReadBarcode()
public int BCRSetReadMode(int readMode)
public int BCRGetReadMode()
public int BCRSetReadTimeOption(int readTime)
public int BCRGetReadTimeOption()
public int BCRSetNegativeBarcode(int flag)
public int BCRGetNegativeBarcode()
public int BCRSetRedundancy(int redundancy)
public int BCRGetRedundancy()
public int BCRSetMarginCheck(int margin)
public int BCRGetMarginCheck()
public String BCRGetFirmwareVersion()
public int BCRGetModuleType()
public String BCRGetLibraryVersion()
public int BCRSendCommand(String command)
public int BCRSetPrefixChar(byte[] prefix)
public byte[] BCRGetPrefixChar()
public int BCRSetSuffixChar(byte[] suffix)
public byte[] BCRGetSuffixChar()
public int BCREnable()
public int BCRDisable()
public int BCRGetStatus()
public boolean BCRBackupSettings(String path)
public boolean BCRRestoreSettings(String path)
public int BCRSetReadableCode(int codeType,boolean status)
public boolean BCRGetReadableCode(int codeType)
public int BCRSetAllReadable(int status)
public int BCRGetAllReadable()
public int BCRSetBarcodeLength(int codeType,int lengthMode,int length)
public int BCRGetBarcodeLength(int codeType,int lengthMode)
public int BCRSetGS1Conversion(boolean conversion)
public boolean BCRGetGS1Conversion()
public int BCRSetUPCOptions(UPC_Option option)
public UPC_Option BCRGetUPCOptions()
public int BCRSetEANOptions(EAN_Option option)
public EAN_Option BCRGetEANOptions()
public int BCRSetCODE39Options(CODE39_Option option)
public CODE39_Option BCRGetCODE39Options()
public int BCRSetCODABAROptions(CODABAR_Option option)
public CODABAR_Option BCRGetCODABAROptions()
public int BCRSet25SCODEOptions(CODE25S_Option option)
public CODE25S_Option BCRGet25SCODEOptions()
public int BCRSetCODE128Options(CODE128_Option option)
public CODE128_Option BCRGetCODE128Options()
public int BCRSetIATAOptions(IATA_Option option)
public IATA_Option BCRGetIATAOptions()
public int BCRSetMSIOptions(MSI_Option option)
public MSI_Option BCRGetMSIOptions()
public int BCRSetUKOptions(UK_Option option)
public UK_Option BCRGetUKOptions()
public int BCRSetTELEPENOptions(TELEPEN_Option option)
public TELEPEN_Option BCRGetTELEPENOptions()
public int BCRSetCODE11Options(CODE11_Option option)
public CODE11_Option BCRGetCODE11Options()
public int BCRSetKOREANPAOptions(KOREANPA_Option option)
public KOREANPA_Option BCRGetKOREANPAOptions()
```

### 3.1.4 Web Page API

H-27 mimics the function of Googles barcode reader application ZXing (<https://play.google.com/store/apps/details?id=com.google.zxing.client.android>) and thereby provides a way to invoke a barcode scan from a web page and have the result returned via a callback URL. See <https://github.com/zxing/zxing/wiki/Scanning-From-Web-Pages>.



The H-27 implementation will register to handle the following URLs:

- `barcode://scan?ret=`
- `http://barcode.opticon.com/scan?`
- `zxing://scan/?ret=`
- `http://zxing.appspot.com/scan?ret=`

In order to make H-27 a simple yet a ZXing compatible implementation we implement:

1. {CODE} is set to:
  - a. decoded string if `raw=true`
  - b. an empty string (i.e., `""`) otherwise
2. {RAWCODE} is set to decoded string
3. {FORMAT} is set to the barcode type
4. {TYPE} is set to an empty string (i.e., `""`)
5. {META} is set to an empty string (i.e., `""`)
6. {SCAN\_FORMATS} is not supported and will be ignored. Barcode reader settings has to be configured in the Settings->Barcode reader.

Example:

[barcode://scan/?ret=http%3A%2F%2Fwww.google.com%2Fsearch%3Fq%3D%7BFORMAT%7D%2B%7BCODE%7D%2B%7BRAWCODE%7D&SCAN\\_FORMATS=UPC\\_A,EAN\\_13](http://barcode://scan/?ret=http%3A%2F%2Fwww.google.com%2Fsearch%3Fq%3D%7BFORMAT%7D%2B%7BCODE%7D%2B%7BRAWCODE%7D&SCAN_FORMATS=UPC_A,EAN_13)

### 3.1.5 Handling of URI Content Barcodes

H-27 have an option to let the built in barcode scanning implementation check the content of a scanned barcode and in case the content is a URI that has a registered intent handler, this URI intent will be launched.

There's a large amount of standard and non standard URI schemes, see [http://en.wikipedia.org/wiki/URI\\_scheme](http://en.wikipedia.org/wiki/URI_scheme). Some examples generated with this <http://zxing.appspot.com/generator/>.

http-URI - <http://apps.opticon.com>



geo-URI - <geo:42.374260,-71.120824>



tel-URI - <tel:+46123456789>



sms-URI (ZXings generator doesn't use the proper URI scheme) -  
<sms:+15105550101?body=hello%20there>



#### 3.1.5.1 Registering Custom URI Intent Handler

Custom URI handlers can be registered by using the standard Android SDK. See,

- <http://stackoverflow.com/questions/3471503/how-to-listen-for-a-custom-uri>
- <http://developer.android.com/guide/topics/manifest/data-element.html>



### 3.1.6 Barcode Reader Logcat Output

For debugging purpose the following logcat filter can be used:

```
adb logcat OPTICON-BCR:D BCR_SERIAL:D BCRJ:D BCRB:D BCR Tracker:D  
BCREnabler:D BCRConfigStore:D BCRD_COMMAND:D BCRCodeReadableSettings:D *:S
```

To understand and analyze the serial commands sent to/received from the barcode reader module please see the Opticon Universal Menu book.

## 3.2 Main Battery Health API

H-27 provide access to the main battery fuel gauge to read detailed battery health information. The fuel gauge used in H-27 batteries are <http://www.ti.com/lit/ds/symlink/bq27541-v200.pdf>. For full understanding of the functions exposed in this API, please see the TI fuel gauge specification linked to above.

### 3.2.1 Java library

.libs\ com.oem.battery.sdk.jar

### 3.2.2 Methods

*getAtRate()*

returns the AtRate( ) value (in mA) is a signed integer, with negative values interpreted as a discharge current value.

*getAtRateTimeToEmpty()*

returns the predicted operating time (in minutes) at the AtRate value of discharge

*getTemperature()*

returns an unsigned integer value of the battery temperature in units of 0.1K measured by the fuel gauge.

*getVoltage()*

returns an unsigned integer value of the measured cell-pack voltage in mV

*getFlags()*

returns the contents of the gas-gauge status register, depicting the current operating status

*getNominalAvailableCapacity*

returns the uncompensated (less than C/20 load) battery capacity remaining. Units are mAh.

*getFullAvailableCapacity()*

returns the uncompensated (less than C/20 load) capacity of the battery when fully charged. Units are mAh.

*getRemainingCapacity()*

returns the compensated battery capacity remaining. Units are mAh.

*getFullChargeCapacity()*

returns the compensated capacity of the battery when fully charged. Units are mAh.

*getAverageCurrent()*

returns a signed integer value that is the average current flow through the sense resistor. It is updated every 1 second. Units are mA.

*getTimeToEmpty()*





returns an unsigned integer value of the predicted remaining battery life at the present rate of discharge

*getTimeToFull()*

returns an unsigned integer value of predicted remaining time until the battery reaches full charge

*getStandbyCurrent()*

returns a signed integer value of the measured standby current through the sense resistor

*getStandbyTimeToEmpty()*

returns an unsigned integer value of the predicted remaining battery life at the standby rate of discharge, in minutes

*getMaxLoadCurrent()*

returns an unsigned integer value of the predicted remaining battery life at the maximum load current discharge rate, in minutes.

*getAvailableEnergy()*

returns an unsigned integer value of the predicted charge or energy remaining in the battery. The value is reported in units of mWh.

*getAveragePower()*

returns an unsigned integer value of the average power of the current discharge. It is negative during discharge and positive during charge. A value of 0 indicates that the battery is not being discharged. The value is reported in units of mW.

*getTTEatConstantPower()*

returns an unsigned integer value of the predicted remaining operating time if the battery is discharged at the AveragePower( ) value in minutes

*getCycleCount()*

returns an unsigned integer value of the number of cycles the battery has experienced with a range of 0 to 65,535. One cycle occurs when accumulated discharge  $\geq$  CC Threshold.

*getStateOfCharge()*

returns an unsigned integer value of the predicted remaining battery capacity expressed as a percentage of FullChargeCapacity, with a range of 0 to 100%. Be aware that the device need to shut down gracefully at 5% battery capacity as the hardware has a hard shutdown at 3.5V. Battery capacity in Android is therefore mapped so that 0% battery capacity in UI is actually 5% battery capacity when read from the battery fuel gauge.

*getDesignCapacity()*

returns the value is stored in Design Capacity and is expressed in mAh.

*getFirstFullyChargedCapacity ()*

return the FullChargeCapacity value in mAh from the first time this battery was fully charged in H-27

*getBatterySerialNumber ()*

returns the battery serial number as a string

*getManufacturingDate ()*

returns the manufacturing date as a string in the format yyyy-Wdd (i.e., <year>-W<week number>) E.g., 2014-W04

*getControl (String subcmd)*

provides a way to read 4 of the fuel gauge control values.



*BatteryInterfaceManager.BQ27541\_SUBCMD\_DEVCIE\_TYPE*  
reports the device type of 0x0541 (indicating bq27541)

*BatteryInterfaceManager.BQ27541\_SUBCMD\_CHEM\_ID*  
reports the chemical identifier of the Impedance Track <sup>TM</sup> configuration

*BatteryInterfaceManager.BQ27541\_SUBCMD\_FW\_VER*  
reports the firmware version on the device type

*BatteryInterfaceManager.BQ27541\_SUBCMD\_HW\_VER*  
reports the hardware version of the device type

*getBackupBatteryType ()*  
returns the device to use the current battery status, the return value is 0, no battery. The return value is 1, Main battery power. The return value is 2, Backup battery power.

*getBackupBatteryCapacity ()*  
returns the Backup battery charge percentage.

*getBackupBatteryMvoltage ()*  
returns the current voltage in mv Backup battery for.

*getBackupBatteryChargeState ()*  
returns Backup battery current state. Return 0, the current backup battery does not exist. Returns 5, Backup battery is already full, otherwise unfilled.

### 3.2.3 Sample Code

See,

- .\samples - src\BatteryApiTest.zip
- .\samples - APKs\BatteryApiTest.apk

## 3.3 Hotswap Backup Battery API

The backup battery information is exposed in two ways.

- Via the Battery Health API
- Via extensions of the standard Android ACTION\_BATTERY\_CHANGED intent

### 3.3.1 Via the Battery Health API

See the methods *getBackupBatteryType ()*, *getBackupBatteryCapacity ()*, *getBackupBatteryMvoltage ()*, and *getBackupBatteryChargeState ()* in section 3.2.2.

### 3.3.2 Via the ACTION\_BATTERY\_CHANGED Intent

The Android ACTION\_BATTERY\_CHANGED intent ([http://developer.android.com/reference/android/content/Intent.html#ACTION\\_BATTERY\\_CHANGE\\_D](http://developer.android.com/reference/android/content/Intent.html#ACTION_BATTERY_CHANGE_D)) is extended with H-27 backup battery information. In this intent there are three *IntExtra* values with the following key name strings:

- "backup\_battery\_mvoltage"
  - voltage level in mV
- "backup\_battery\_capacity"
  - capacity in percent

- *"backup\_battery\_charge\_state"*
  - Bit 0: 1 = battery present, 0 = no battery
  - Bit 1: 1 = charging, 0 = not charging
  - Bit 2: 1 = fully charged, 0 = not fully charged

### 3.4 Battery Cover API

When the H-27 battery back cover is removed or replaced an intent is broadcasted. The intent name is *"com.oem.hardware.intent.action.backcover"* and it has one IntExtra value:

- *"state"*
  - 0 for removed, 1 for reapplied

Observe: This intent is also fired when H-27 exits suspend.

#### 3.4.1 Sample Code

See,

- `.\samples - src\BackCoverDetection.zip`
- `.\samples - APKs\BackCoverDetection.apk`

### 3.5 Scan Button API

To simplify the usage of the left/right scan button in solutions built in H-27 the scan buttons fire intents:

- `com.oem.barcode.BCRIntents.ACTION_LEFT_SCAN` - Constant string value *"com.oem.hardware.intent.action.LEFT\_SCAN"*
  - Contains a KeyEvent
- `com.oem.barcode.BCRIntents.ACTION_LEFT_SCAN_LONG_PRESS` - Constant string value *"com.oem.hardware.intent.action.LEFT\_SCAN\_LONG\_PRESS"*
- `com.oem.barcode.BCRIntents.ACTION_RIGHT_SCAN` - Constant string value *"com.oem.hardware.intent.action.RIGHT\_SCAN"*
  - Contains a KeyEvent
- `com.oem.barcode.BCRIntents.ACTION_RIGHT_SCAN_LONG_PRESS` - Constant string value *"com.oem.hardware.intent.action.RIGHT\_SCAN\_LONG\_PRESS"*

In the intents `ACTION_LEFT_SCAN` and `ACTION_RIGHT_SCAN` a KeyEvent extra object can be extracted from the intent by

```
(KeyEvent)intent.getExtra(Intent.EXTRA_KEY_EVENT)
```

#### 3.5.1 Sample Code

See,

- `.\samples - src\KeyRemapViaIntents.zip`
- `.\samples - APKs\KeyRemapViaIntents.apk`

### 3.6 Controlling the Application Specific LED

To be described



### 3.7 Dock Detection API

When placing H-27 in a cradle a dock detection intent is fired. This is a default Android API. See <http://developer.android.com/training/monitoring-device-state/docking-monitoring.html>. Only Desk dock detection can be done with H-27 (i.e., EXTRA\_DOCK\_STATE\_DESK)

#### 3.7.1 Sample Code

See,

- .\samples - src\DockDetection.zip
- .\samples - APKs\DockDetection.apk

## 4 Qualcomm APIs

### 4.1 Proprietary BLE/GATT API for Android 4.2

Bluetooth BLE/GATT support wasn't officially added to Android until Android 4.3. In order to support Bluetooth BLE/GATT in Android 4.2 we expose the Qualcomm proprietary implementation in H-27 to support developers who need this function.

#### 4.1.1 Java library

.\libs\ com.oem.bluetooth.sdk.jar

To be described

### 4.2 Snapdragon SDK

The Qualcomm Snapdragon SDK is possible to use with H-27. See <https://developer.qualcomm.com/docs/snapdragon-sdk/reference-api/index.html>, <https://developer.qualcomm.com/download/snapdragon-sdk.zip>

OBSERVE: Please use the Qualcomm SDK verifier App to check that the API function needed is really supported by H-27.