RECOMMENDED 128 BIT CUSTOM UUID MANAGEMENT

Application Note **February 2014**

INTRODUCTION

The goal of this document includes the following:

- Explain best practice when registering custom 128 bit UUIDs on *smart*BASIC modules.

OVERVIEW

There is a limited number of 128 bit base UUIDs that can be registered to the GATT table when using a microcontroller with a small amount of memory. On the BL6xx modules running the latest firmware, the number of unique 128 bit base UUIDs is limited to ten. To work around this limit, generate and register a custom base 128 bit UUID, then register 16 bit UUIDs which will be offset from that base UUID at bytes two and three. This technique allows you to register up to 65535 custom UUIDs per base UUID. Therefore 655350 total custom UUIDs can be utilised.

Laird recommends that a user generates a single 128 bit base UUID and then manage 65535 16 bit UUIDs. It is unlikely that you will ever require more.

For example, Laird has generated a base UUID (569a0000-b87f-490c-92cb-11ba5ea5167c) which is always registered in the firmware. This means that nine are available for the user.

UUIDs can be generated on the following website: http://www.guidgenerator.com/. As stated on the website:

> "128-bits is big enough and the generation algorithm is unique enough that if 1,000,000,000 GUIDs per second were generated for 1 year the probability of a duplicate would be only 50%. Or if every human on Earth generated 600,000,000 GUIDs there would only be a 50% probability of a duplicate."

Laird maintains a spreadsheet that ensures that no two 16 bit UUIDs are the same for any custom service or characteristic published by Laird.

EXAMPLE

Using the Laird base UUID as an example, notice that bytes two and three are zeroes (0).

```
'// Laird Technologies 128 bit Base UUID
#define LT_BASE_UUID              "\56\9a\00\00\b8\7f\49\0c\92\cb\11\ba\5e\a5\16\7c"
```

Register the base UUID by calling BleHandleUuid128(). This returns a handle which is used when registering subsequent 16 bit UUIDs.

```
'//Laird Technologies Base UUID Handle
dim bseUuid$ : bseUuid$ = LT_BASE_UUID
dim hBseUuid : hBseUuid=BleHandleUuid128(bseUuid$)
```

Next we register a few of the custom Laird iBeacon Service UUIDs using BleHandleUuidSibling() which takes the handle of the base UUID obtained above as its first parameter and a 16 bit offset UUID as the other.

```
dim hSvc : hSvc = BleHandleUuidSibling(hBseUuid, 0x1900) //custom iBeacon service UUID
dim hMaj : hMaj = BleHandleUuidSibling(hBseUuid, 0x2013) //custom iBeacon Major Value UUID
dim hAdI : hAdI = BleHandleUuidSibling(hBseUuid, 0x2015) //custom iBeacon advertising interval UUID
```

When connected to a GATT client the UUIDs registered above will be shown as follows:

Full iBeacon Service UUID: **569a*1900*-b87f-490c-92cb-11ba5ea5167c**

Full iBeacon Major value UUID: **569a2013-b87f-490c-92cb-11ba5ea5167c**

Full iBeacon Advertising Interval UUID: **569a*2015*-b87f-490c-92cb-11ba5ea5167c**

FURTHER INFORMATION

For more information on the characteristics and their UUIDs, see the **$autorun$.iBeacon.sblib** file.

Further information relating to firmware and the use of UWTerminal is available from the Laird website at http://www.lairdtech.com/Products/Embedded-Wireless-Solutions/Bluetooth-Radio-Modules/BL600-Series