

青风带你玩蓝牙 nRF52832 系列教程.....	2
-----作者: 青风.....	2
作者: 青风.....	3
出品论坛: www.qfv8.com	3
淘宝店: http://qfv5.taobao.com	3
QQ 技术群: 346518370.....	3
硬件平台: 青云 QY-nRF52832 开发板.....	3
2.21 蓝牙任务的 UUID 设置与总结.....	3
1: UUID 设置规则及原理:	3
1.1 蓝牙技术联盟 UUID.....	3
1.2 供应商特定的 UUID.....	4
2: 程序中 UUID 的设置:	6
2.1 蓝牙协议小组公共 UUID 设置.....	6
2.2 私有服务 UUID 设置:	7
2.3 UUID 类型切换:	9
3 应用与调试.....	11
4.1 下载.....	11
3.2 测试.....	12

青风带你玩蓝牙 nRF52832 系列教程

-----作者: 青风

出品论坛: www.qfv8.com 青风电子社区



作者: 青风

出品论坛: www.qfv8.com

淘宝店: <http://qfv5.taobao.com>

QQ 技术群: 346518370

硬件平台: 青云 QY-nRF52832 开发板

21 蓝牙任务的 UUID 设置与总结

本节我们讲主要探讨一下蓝牙任务的 UUID 的设置, 本节其实也是前面几章里提到的 UUID 的总结。

在前面详解篇第一节到第四节中都详细提过 UUID 的设置, 但是还是有读者不能很好理解, 现在我们拿出来单独总结归纳一下。

这里我们通过一个简单的例子: 蓝牙 BLE 蓝牙串口, 来进行一个简单的思路验证。注意本例在蓝牙串口的基础上进行修改。

1: UUID 设置规则及原理:

UUID 含义是通用唯一识别码 (Universally Unique Identifier), 这是一个软件建构的标准。UUID 是指在一台机器上生成的数字, 它保证对在同一时空中的所有机器都是唯一的。通常平台会提供生成的 API。

在“GATT 层”中规范定义的所有属性都有一个 UUID 值, UUID 是全球唯一的 128 位的号码, 它用来识别不同的特性。

1.1 蓝牙技术联盟 UUID

所有的蓝牙技术联盟定义 UUID 共用了一个基本的 UUID:

0x0000xxxx-0000-1000-8000-00805F9B34FB

为了进一步简化基本 UUID, 每一个蓝牙技术联盟定义的属性有一个唯一的 16 位 UUID, 以代替上面的基本 UUID 的‘x’部分。例如, 心率测量特性使用 0X2A37 作为它的 16 位 UUID, 因此它完整的 128 位 UUID 为:

0x00002A37-0000-1000-8000-00805F9B34FB

虽然蓝牙技术联盟使用相同的基本 UUID,但是 16 位的 UUID 足够唯一地识别蓝牙技术联盟所定义的各种属性。

蓝牙技术联盟所用的基本 UUID 不能用于任何定制的属性、服务和特性。对于定制的属性,必须使用另外完整的 128 位 UUID。

1.2 供应商特定的 UUID

SoftDevice 根据蓝牙技术联盟定义 UUID 类似的方式定义 UUID: 先增加一个特定的基本 UUID,再定义一个 16 位的 UUID (类似于一个别名),再加载在基本 UUID 之上。这种采用为所有的定制属性定义一个共用的基本 UUID 的方式使得应用变为更加简单,至少在同一服务中更是如此。

使用软件 **nRFgo Studio** 非常容易产生一个新的基本 UUID,生成过程如下:

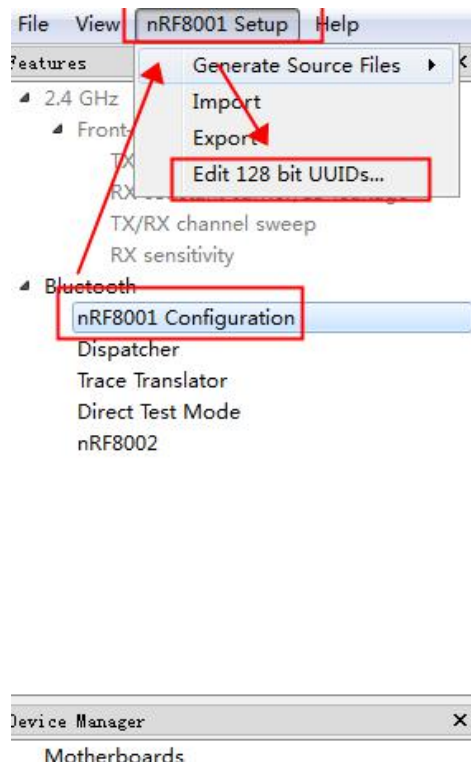
UUID 需要重新设置,因为本服务中将要使用一个定制(私有)的 UUID,以代替蓝牙技术联盟所定义的 UUID。

首先,先定义一个基本 UUID,一种方式是采用 **nRFgo Studio** 来生成:

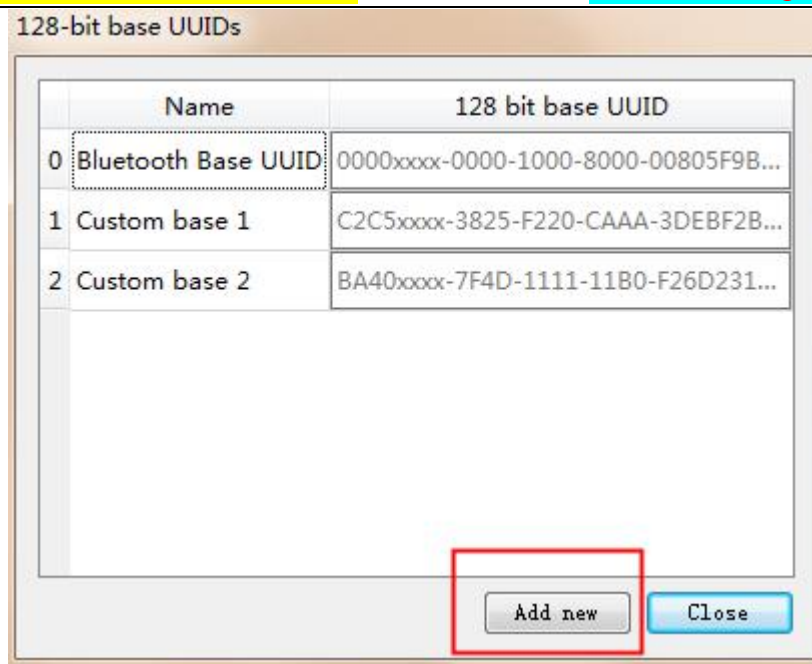
1.打开 nRFgo Studio

2.在 nRF8001 Setup 菜单中,选择 Edit 128-bit UUIDs 选项,点击 Add new。

如下图所示:



这就产生了一个随机的 UUID,可以用于你的定制服务中。



上图 0 就是蓝牙协议小组定义的基本 UUID, 1 和 2 就是随机生成的 UUID, xxxx 就是 16 位的 UUID。

例如, 在 BLE 串口 示例中, 采用

0x6E40xxxx-B5A3-F393-E0A9-E50E24DCCA9E 作为基本 UUID。

蓝牙核心规范没有任何规则或是建议如何对加入基本 UUID 的 16 位 UUID 进行分配, 因此你可以按照你的意图来任意分配。

例如, 在 BLE 串口示例中, 0x0001 作为服务的 UUID, 0x0002 作为串口 TX 特性的 UUID。0x0003 作为串口 RX 特性的 UUID。

为了可读性, 在头文件 `ble_nus.h` 中以宏定义的方式添加, 连同用于服务和特性的 16 位 UUID 也一起定义:

```
#define BLE_UUID_NUS_SERVICE          0x0001
/**< The UUID of the Nordic UART Service. */

#define BLE_UUID_NUS_TX_CHARACTERISTIC 0x0002
/**< The UUID of the TX Characteristic. */

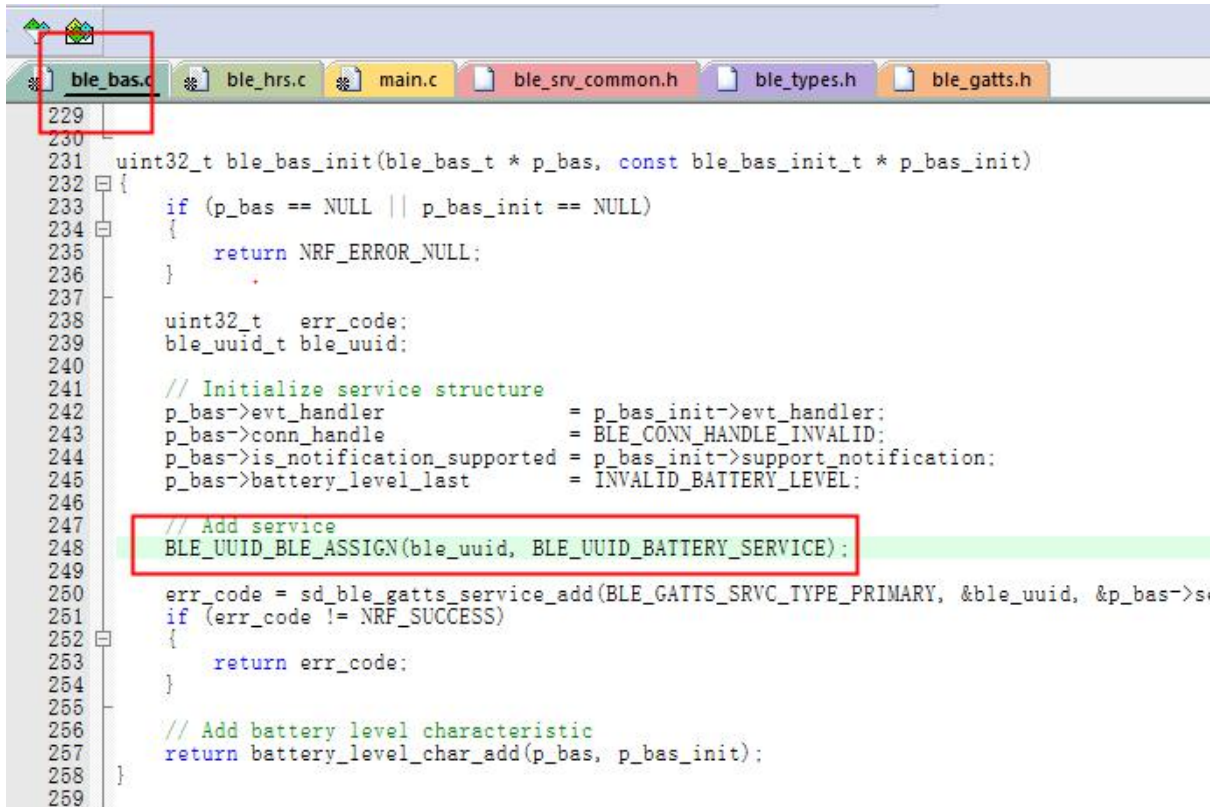
#define BLE_UUID_NUS_RX_CHARACTERISTIC 0x0003
```

综上所述, 大家能够理解了, 在使用蓝牙协议小组的共有服务的时候, 使用的 UUID 为蓝牙技术联盟定义的 UUID, 128 位。而在使用用户自己定义的私有服务的时候, 则使用自己定义生成的 UUID。那么在代码里如何设置的了?

2: 程序中 UUID 的设置:

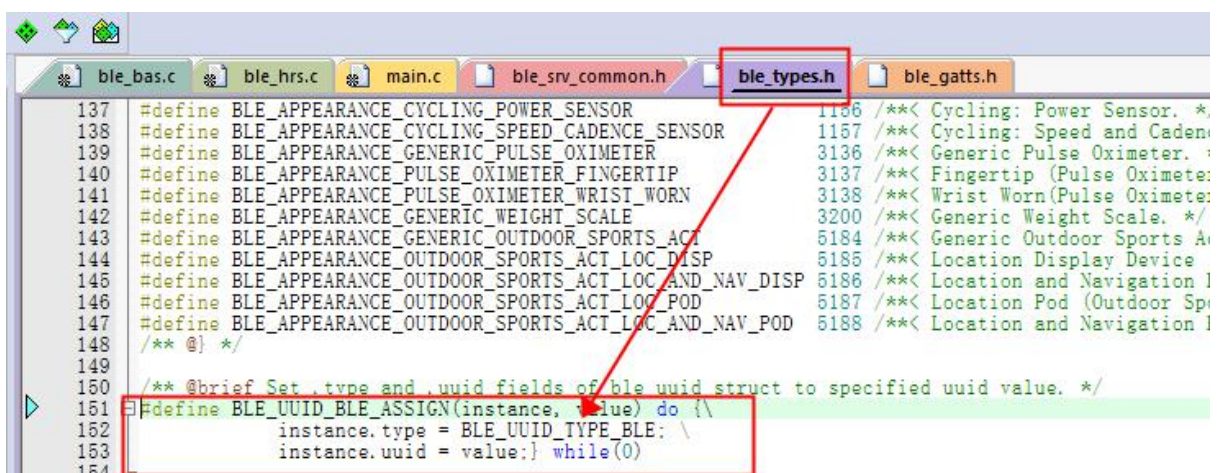
2.1 蓝牙协议小组公共 UUID 设置

举一个简单的例子, 打开任意一个蓝牙协议下载定义的服务文件, 比如我们打开心电服务的例子, 找到心电的服务函数 `ble_bas.c`, 找到协议栈 UUID 设置代码如下:



```
229 uint32_t ble_bas_init(ble_bas_t * p_bas, const ble_bas_init_t * p_bas_init)
230 {
231     if (p_bas == NULL || p_bas_init == NULL)
232     {
233         return NRF_ERROR_NULL;
234     }
235
236     uint32_t err_code;
237     ble_uuid_t ble_uuid;
238
239     // Initialize service structure
240     p_bas->evt_handler = p_bas_init->evt_handler;
241     p_bas->conn_handle = BLE_CONN_HANDLE_INVALID;
242     p_bas->is_notification_supported = p_bas_init->support_notification;
243     p_bas->battery_level_last = INVALID_BATTERY_LEVEL;
244
245     // Add service
246     BLE_UUID_BLE_ASSIGN(ble_uuid, BLE_UUID_BATTERY_SERVICE);
247
248     err_code = sd_ble_gatts_service_add(BLE_GATTS_SRVC_TYPE_PRIMARY, &ble_uuid, &p_bas->s;
249     if (err_code != NRF_SUCCESS)
250     {
251         return err_code;
252     }
253
254     // Add battery level characteristic
255     return battery_level_char_add(p_bas, p_bas_init);
256 }
257
258
259
```

找到了 UUID 设置函数 `BLE_UUID_BLE_ASSIGN`, 点击反键找到其源函数:



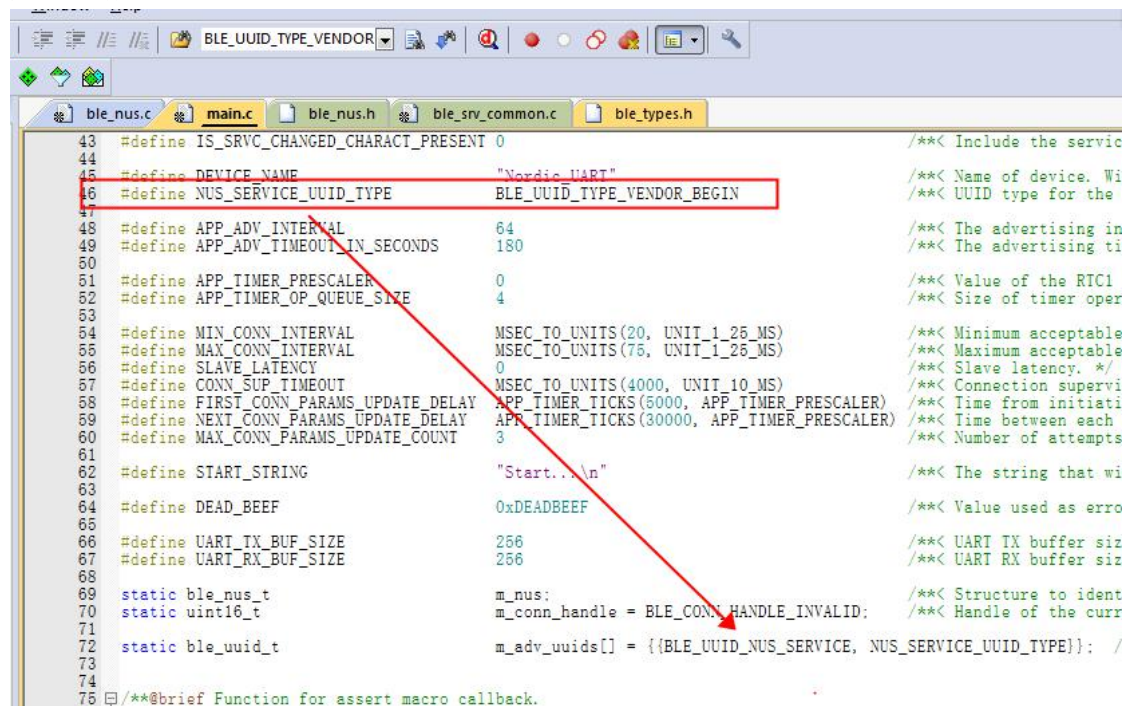
```
137 #define BLE_APPEARANCE_CYCLING_POWER_SENSOR 1186 /**< Cycling: Power Sensor. */
138 #define BLE_APPEARANCE_CYCLING_SPEED_CADENCE_SENSOR 1187 /**< Cycling: Speed and Cadence. */
139 #define BLE_APPEARANCE_GENERIC_PULSE_OXIMETER 3136 /**< Generic Pulse Oximeter. */
140 #define BLE_APPEARANCE_PULSE_OXIMETER_FINGERTIP 3137 /**< Fingertip (Pulse Oximeter). */
141 #define BLE_APPEARANCE_PULSE_OXIMETER_WRIST_WORN 3138 /**< Wrist Worn (Pulse Oximeter). */
142 #define BLE_APPEARANCE_GENERIC_WEIGHT_SCALE 3200 /**< Generic Weight Scale. */
143 #define BLE_APPEARANCE_GENERIC_OUTDOOR_SPORTS_ACTIVITY 5184 /**< Generic Outdoor Sports Activity. */
144 #define BLE_APPEARANCE_OUTDOOR_SPORTS_ACTIVITY_LOCATION_DISPLAY 5185 /**< Location Display Device. */
145 #define BLE_APPEARANCE_OUTDOOR_SPORTS_ACTIVITY_LOCATION_AND_NAVIGATION_DISPLAY 5186 /**< Location and Navigation Display. */
146 #define BLE_APPEARANCE_OUTDOOR_SPORTS_ACTIVITY_LOCATION_POD 5187 /**< Location Pod (Outdoor Sports Activity). */
147 #define BLE_APPEARANCE_OUTDOOR_SPORTS_ACTIVITY_LOCATION_AND_NAVIGATION_POD 5188 /**< Location and Navigation Pod. */
148 /** @ */
149
150 /** @brief Set .type and .uuid fields of ble_uuid struct to specified uuid value. */
151 #define BLE_UUID_BLE_ASSIGN(instance, value) do { \
152     instance.type = BLE_UUID_TYPE_BLE; \
153     instance.uuid = value; } while(0)
154
```

设置了心电的 UUID 的类型为 `BLE_UUID_TYPE_BLE`, 这个 UUID 就是蓝牙协议小组的 UUID, 也就是说只需要管 `0x0000xxxx-0000-1000-8000-00805F9B34FB` 中的第 XXXX 为这个 16 位 UUID 就可以了:

```
/** @defgroup BLE_UUID_TYPES Types of UUID
 * @{ */
#define BLE_UUID_TYPE_UNKNOWN 0x00 /**< Invalid UUID type. */
#define BLE_UUID_TYPE_BLE 0x01 /**< Bluetooth SIG UUID (16-bit). */
#define BLE_UUID_TYPE_VENDOR_BEGIN 0x02 /**< Vendor UUID types start at this index (128-bit). */
/** @} */
```

2.2 私有服务 UUID 设置:

打开蓝牙串口历程, 在主函数 main.c 中, 就定义了广播的私有服务 UUID 类型了, 如下图所示:

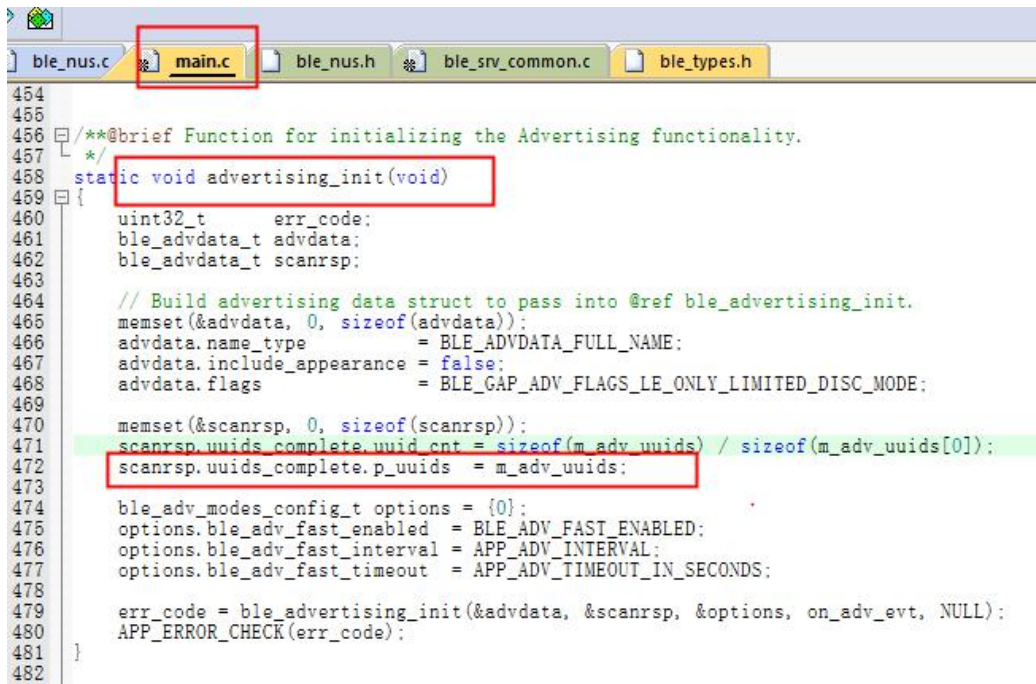


```
43 #define IS_SRVC_CHANGED_CHARACT_PRESENT 0 /**< Include the servic
44
45 #define DEVICE_NAME "Nordic UART" /**< Name of device. Wi
46 #define NUS_SERVICE_UUID_TYPE BLE_UUID_TYPE_VENDOR_BEGIN /**< UUID type for the
47
48 #define APP_ADV_INTERVAL 64 /**< The advertising in
49 #define APP_ADV_TIMEOUT_IN_SECONDS 180 /**< The advertising ti
50
51 #define APP_TIMER_PRESCALER 0 /**< Value of the RTC1
52 #define APP_TIMER_OP_QUEUE_SIZE 4 /**< Size of timer oper
53
54 #define MIN_CONN_INTERVAL MSEC_TO_UNITS(20, UNIT_1_25_MS) /**< Minimum acceptable
55 #define MAX_CONN_INTERVAL MSEC_TO_UNITS(75, UNIT_1_25_MS) /**< Maximum acceptable
56 #define SLAVE_LATENCY 0 /**< Slave latency. */
57 #define CONN_SUP_TIMEOUT MSEC_TO_UNITS(4000, UNIT_10_MS) /**< Connection supervi
58 #define FIRST_CONN_PARAMS_UPDATE_DELAY APP_TIMER_TICKS(5000, APP_TIMER_PRESCALER) /**< Time from initiati
59 #define NEXT_CONN_PARAMS_UPDATE_DELAY APP_TIMER_TICKS(30000, APP_TIMER_PRESCALER) /**< Time between each
60 #define MAX_CONN_PARAMS_UPDATE_COUNT 3 /**< Number of attempts
61
62 #define START_STRING "Start...\n" /**< The string that wi
63
64 #define DEAD_BEEF 0xDEADBEEF /**< Value used as erro
65
66 #define UART_TX_BUF_SIZE 256 /**< UART TX buffer siz
67 #define UART_RX_BUF_SIZE 256 /**< UART RX buffer siz
68
69 static ble_nus_t m_nus; /**< Structure to ident
70 static uint16_t m_conn_handle = BLE_CONN_HANDLE_INVALID; /**< Handle of the curr
71
72 static ble_uuid_t m_adv_uuids[] = {{BLE_UUID_NUS_SERVICE, NUS_SERVICE_UUID_TYPE}}; /
73
74
75 /**@brief Function for assert macro callback.
```

也就是说的 128bit UUID---->BLE_UUID_TYPE_VENDOR_BEGIN 类型。

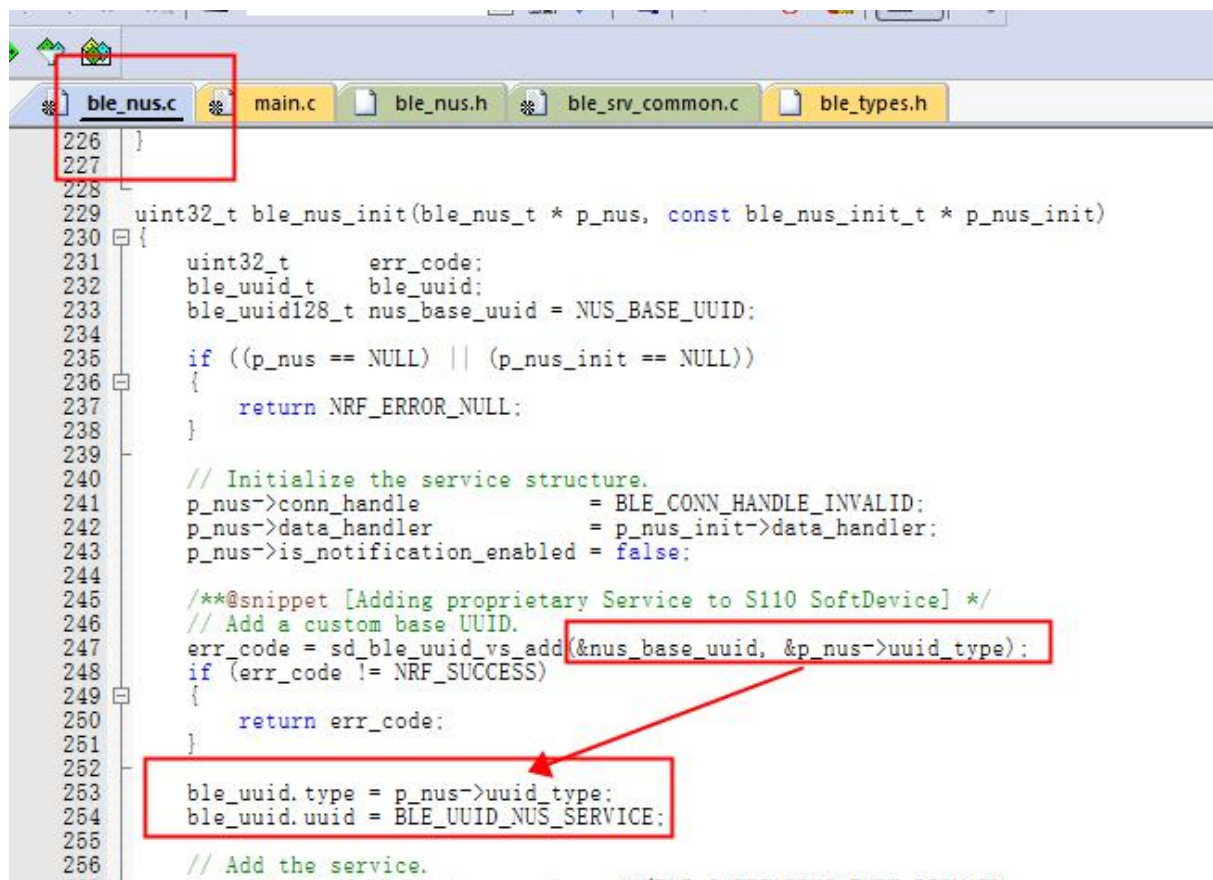
```
6
7
8 /** @defgroup BLE_UUID_TYPES Types of UUID
9 * @{ */
0 #define BLE_UUID_TYPE_UNKNOWN 0x00 /**< Invalid UUID type. */
1 #define BLE_UUID_TYPE_BLE 0x01 /**< Bluetooth SIG UUID (16-bit). */
2 #define BLE_UUID_TYPE_VENDOR_BEGIN 0x02 /**< Vendor UUID types start at this index (128-bit). */
3 /** @} */
4
```

同时在广播中设置:



```
454
455
456 /**@brief Function for initializing the Advertising functionality.
457 */
458 static void advertising_init(void)
459 {
460     uint32_t err_code;
461     ble_advdata_t advdata;
462     ble_advdata_t scanrsp;
463
464     // Build advertising data struct to pass into @ref ble_advertising_init.
465     memset(&advdata, 0, sizeof(advdata));
466     advdata.name_type = BLE_ADVDATA_FULL_NAME;
467     advdata.include_appearance = false;
468     advdata.flags = BLE_GAP_ADV_FLAGS_LE_ONLY_LIMITED_DISC_MODE;
469
470     memset(&scanrsp, 0, sizeof(scanrsp));
471     scanrsp.uuids_complete.uuid_cnt = sizeof(m_adv_uuids) / sizeof(m_adv_uuids[0]);
472     scanrsp.uuids_complete.p_uuids = m_adv_uuids;
473
474     ble_adv_modes_config_t options = {0};
475     options.ble_adv_fast_enabled = BLE_ADV_FAST_ENABLED;
476     options.ble_adv_fast_interval = APP_ADV_INTERVAL;
477     options.ble_adv_fast_timeout = APP_ADV_TIMEOUT_IN_SECONDS;
478
479     err_code = ble_advertising_init(&advdata, &scanrsp, &options, on_adv_evt, NULL);
480     APP_ERROR_CHECK(err_code);
481 }
482
```

服务函数 ble_nus.c 中, 把我们自己定义的 nus_base_uuid 这个定义的 128 位 UUID 赋给指针 p_nus->uuid_type, 然后添加到我们的服务中。如下图所示。



```
226 }
227
228
229 uint32_t ble_nus_init(ble_nus_t * p_nus, const ble_nus_init_t * p_nus_init)
230 {
231     uint32_t err_code;
232     ble_uuid_t ble_uuid;
233     ble_uuid128_t nus_base_uuid = NUS_BASE_UUID;
234
235     if ((p_nus == NULL) || (p_nus_init == NULL))
236     {
237         return NRF_ERROR_NULL;
238     }
239
240     // Initialize the service structure.
241     p_nus->conn_handle = BLE_CONN_HANDLE_INVALID;
242     p_nus->data_handler = p_nus_init->data_handler;
243     p_nus->is_notification_enabled = false;
244
245     /**@snippet [Adding proprietary Service to S110 SoftDevice] */
246     // Add a custom base UUID.
247     err_code = sd_ble_uuid_vs_add(&nus_base_uuid, &p_nus->uuid_type);
248     if (err_code != NRF_SUCCESS)
249     {
250         return err_code;
251     }
252
253     ble_uuid.type = p_nus->uuid_type;
254     ble_uuid.uuid = BLE_UUID_NUS_SERVICE;
255
256     // Add the service.
257
```

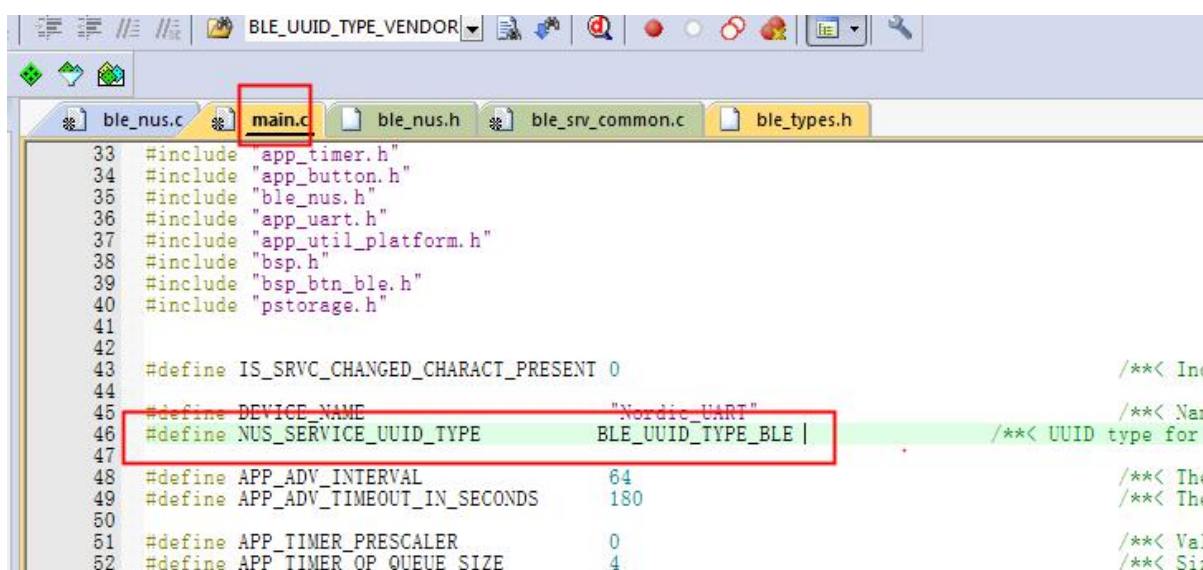

2.3 UUID 类型切换:

虽然我们不建议大家在设置私有服务的时候使用蓝牙协议小组公共 UUID 类型,但是有些读者为了简单之间占用蓝牙协议小组公共 UUID,我们下面简单的验证下如何切换:

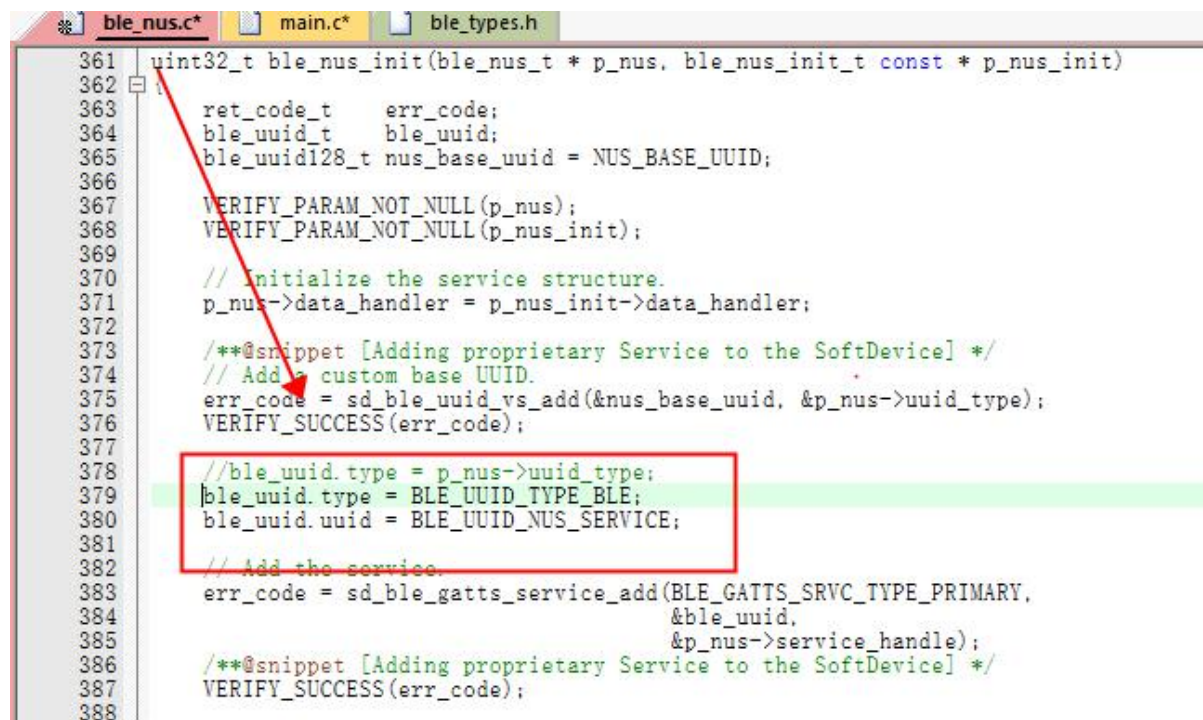
打开蓝牙串口工程,下载协议栈后下载程序,打开 MCP APP,找到服务,查看设置的 UUID 如图所示:



我们修改一下 UUID 类型为 BLE_UUID_TYPE_BLE:

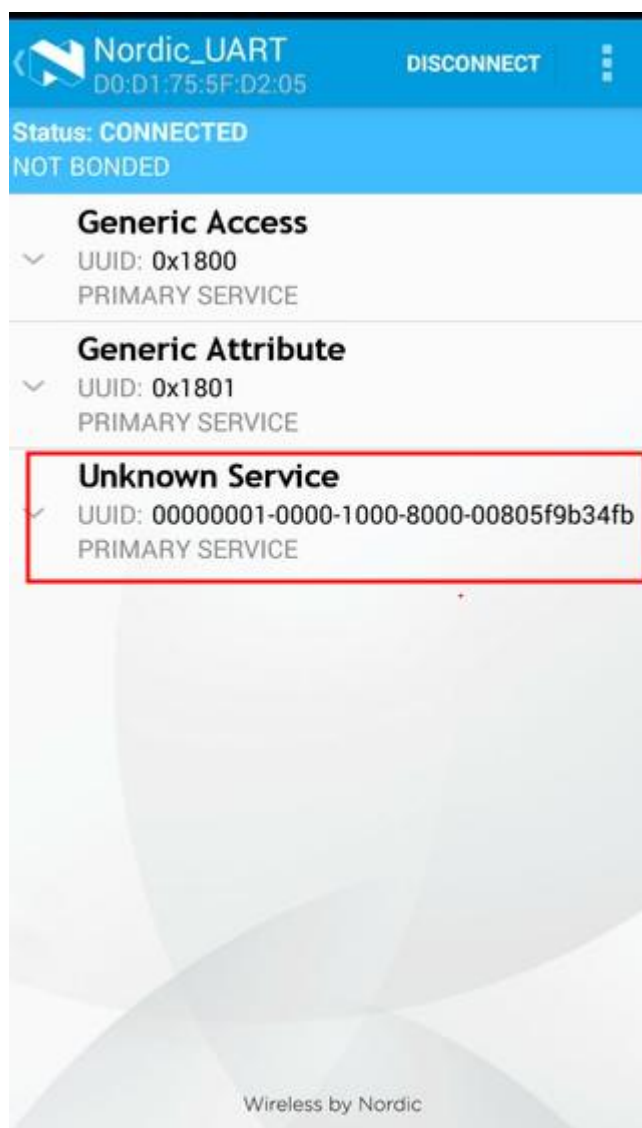


设置服务初始化的主服务 UUID 类型:



```
361 uint32_t ble_nus_init(ble_nus_t * p_nus, ble_nus_init_t const * p_nus_init)
362 {
363     ret_code_t    err_code;
364     ble_uuid_t    ble_uuid;
365     ble_uuid128_t nus_base_uuid = NUS_BASE_UUID;
366
367     VERIFY_PARAM_NOT_NULL(p_nus);
368     VERIFY_PARAM_NOT_NULL(p_nus_init);
369
370     // Initialize the service structure.
371     p_nus->data_handler = p_nus_init->data_handler;
372
373     /**@snippet [Adding proprietary Service to the SoftDevice] */
374     // Add a custom base UUID.
375     err_code = sd_ble_uuid_vs_add(&nus_base_uuid, &p_nus->uuid_type);
376     VERIFY_SUCCESS(err_code);
377
378     //ble_uuid.type = p_nus->uuid_type;
379     ble_uuid.type = BLE_UUID_TYPE_BLE;
380     ble_uuid.uuid = BLE_UUID_NUS_SERVICE;
381
382     // Add the service.
383     err_code = sd_ble_gatts_service_add(BLE_GATTS_SRVC_TYPE_PRIMARY,
384                                         &ble_uuid,
385                                         &p_nus->service_handle);
386     /**@snippet [Adding proprietary Service to the SoftDevice] */
387     VERIFY_SUCCESS(err_code);
388 }
```

编译后下载, 打开 APP MCP 后发现 UUID 发生变化, 变成了蓝牙协议小组定义的公共 UUID 类型, 如下图所示:

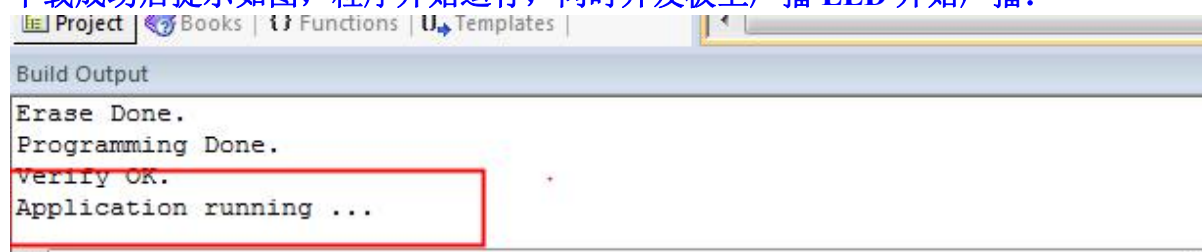


3 应用与调试

4.1 下载

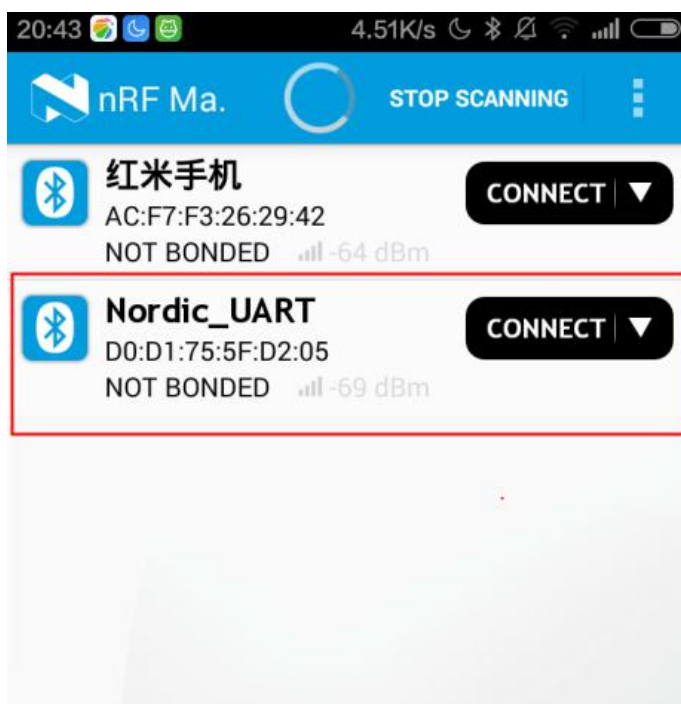
打开 NRFgo 进行下载协议栈，可以蓝牙样例下载方法里介绍。首先整片擦除，后下载协议栈。

下载完后可以下载工程，首先把工程编译一下，通过后点击 KEIL 上的下载按键，下载成功后提示如图，程序开始运行，同时开发板上广播 LED 开始广播：



4.2 测试

打开手机 APP 软件 MCP, 如下图所示:



点击连接后如下图所示, 点击服务查看 UUID, 如下图所示:

