Dialog SDK 5.0.3 培训材料3– 自定义协议的GATT命令

2016.5

…personal
…portable
…connected

BLE 自定义协议

BLE 规范概览

自定义服务规范源码讨论

界面输出显示

BLE 自定义协议

开始准备一个demo…

- **在开始前, 建议你 …**
  - 看一下培训材料1的barebone工程应用
  - 看一下培训材料2的自定义规范应用

- **你将会学到…**
  - 通用属性规范
  - 自定义通用属性的规范应用和消息流程
  - 自定义数据库构建流程
  - 在练习中，学会在自定义数据库里添加用于控制LED灯亮灭的属性

- **接下来…**
  - 继续在SDK5.0.3 DA4580 Dev-kit-Pro的环境下，学习培训材料4,5,6
  - 学习附录的参考文献

BLE 自定义协议

BLE 协议概览

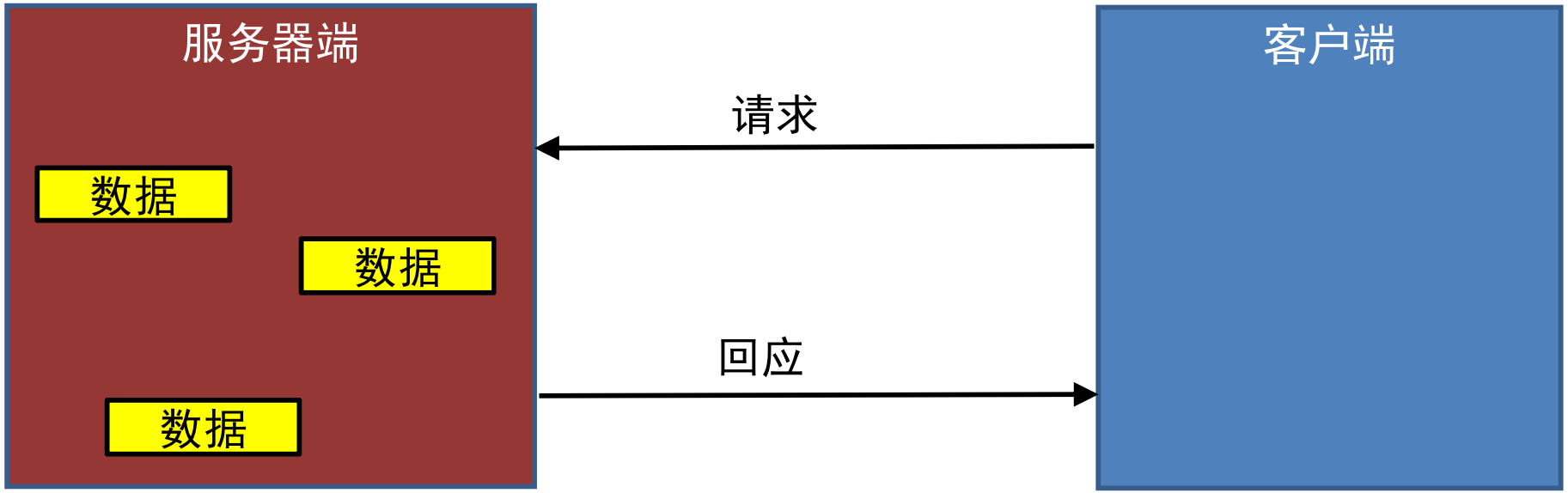
- 低功耗蓝牙(BLE)协议是一组规范关于蓝牙应用行为的标准定义，它是构建在通用属性协议(GATT)上的。
- BLE profile 遵循特定的结构，使得(服务器/外围设备)端能够给(客户端/中央设备)端提供信息接口，用来描述本身的性能和如何获取这些信息的途径.
- 服务器端是数据的拥有者，大多数情况下作为外围设备端.
- 客户端是数据的消费者，大多数情况下作为中央设备端 (智能手机和平板).
- https://developer.bluetooth.org/gatt/services/Pages/ServicesHome.aspx

自定义服务内容

通用属性协议（GATT）

- 客户端/服务器 架构
  - •服务器端存储数据，通常在GAP协议里作为外部设备端
  - •客户端从服务器端申请数据，在GAP协议里作为中央控制端

- 服务器端采用属性的形态开放数据访问接口



BLE Custom profile

BLE 协议概览

- 一个BLE 协议可以包含1个或多个服务.
- 服务通常会把数据分包成逻辑实体，并且包含特定的字段，这些字段被称为属性.
- 一个服务可以有一个或多个属性。服务之间通过UUID来进行区分，UUID可以有16-bit（BLE服务官方采用）或者128-bit(自定义服务）两种形式
- 属性是在GATT交互数据里最底层的概念.
- 属性之间也通过UUID（16bit/128bit）来区分，应用场景类似服务。

BLE 自定义协议

自定义协议服务和源码讨论

显示输出

自定义服务内容

自定义服务涉及的协议实例

- 这个实例会展示
  - Barebone应用工程
  - 128 bit UUID的自定义服务移植
  - 如何访问自定义的服务数据库
  - 这份教程描述了如何逐步建立和广播新的属性特性，并且怎样在中央设备和外部设备间发送和接收GATT的命令
- IDE 环境，使用 KEIL 5
- Dialog semiconductor SDK 版本 5.0.3
- 工程位置: 5.0.3\projects\target_apps\ble_examples\ble_app_profile

自定义服务内容

target_apps\ble_examples\ble_app_peripheral 工程描述了

- 检查自定义服务数据库的访问权限.
- 检查设备的广播包名字.
- 采用DISS服务.
- 检查自定义服务下的属性设定.
- 创建用户自定义属性的格式.

.

自定义服务内容

自定义服务协议下的基本消息交互流程

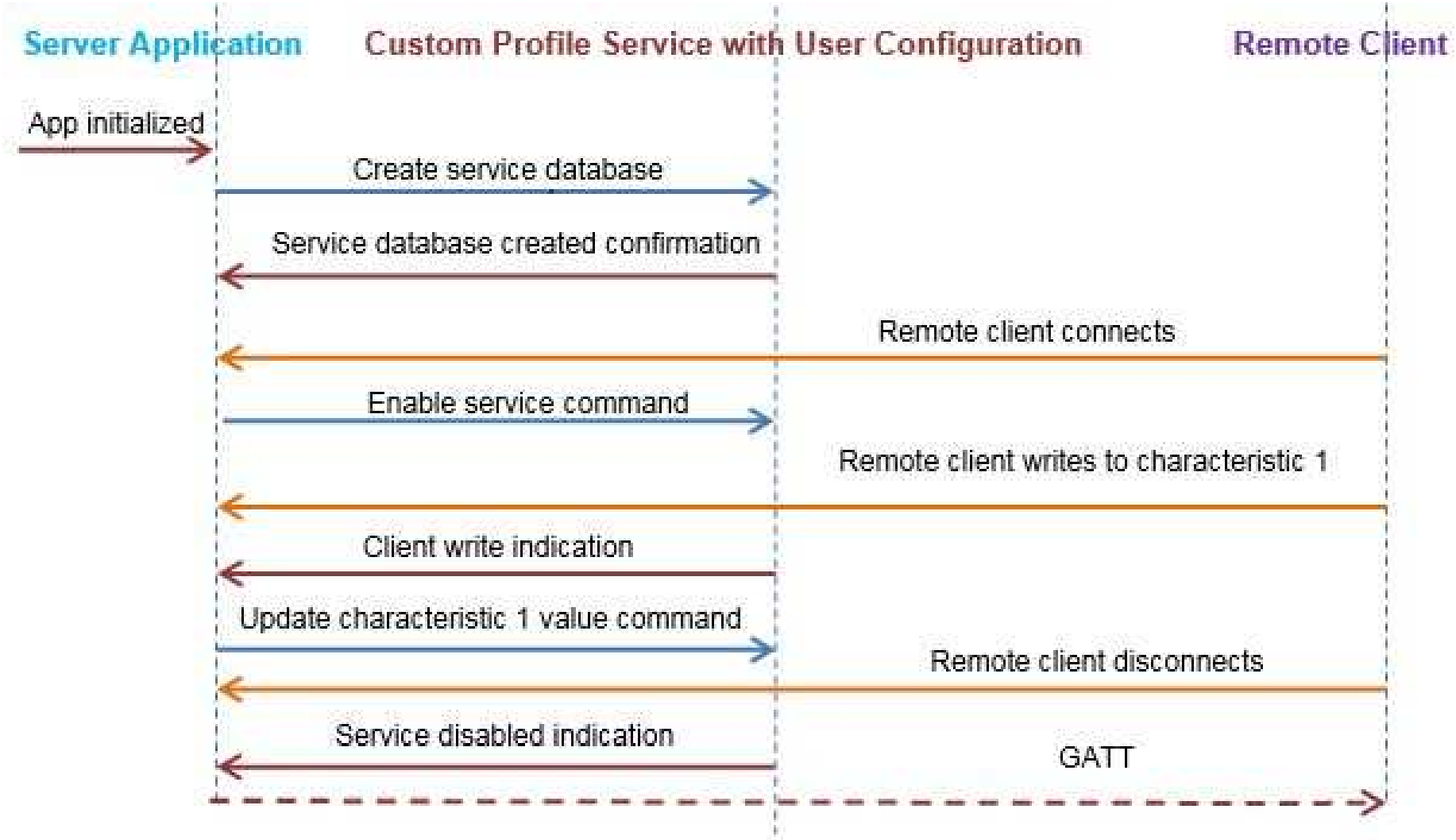
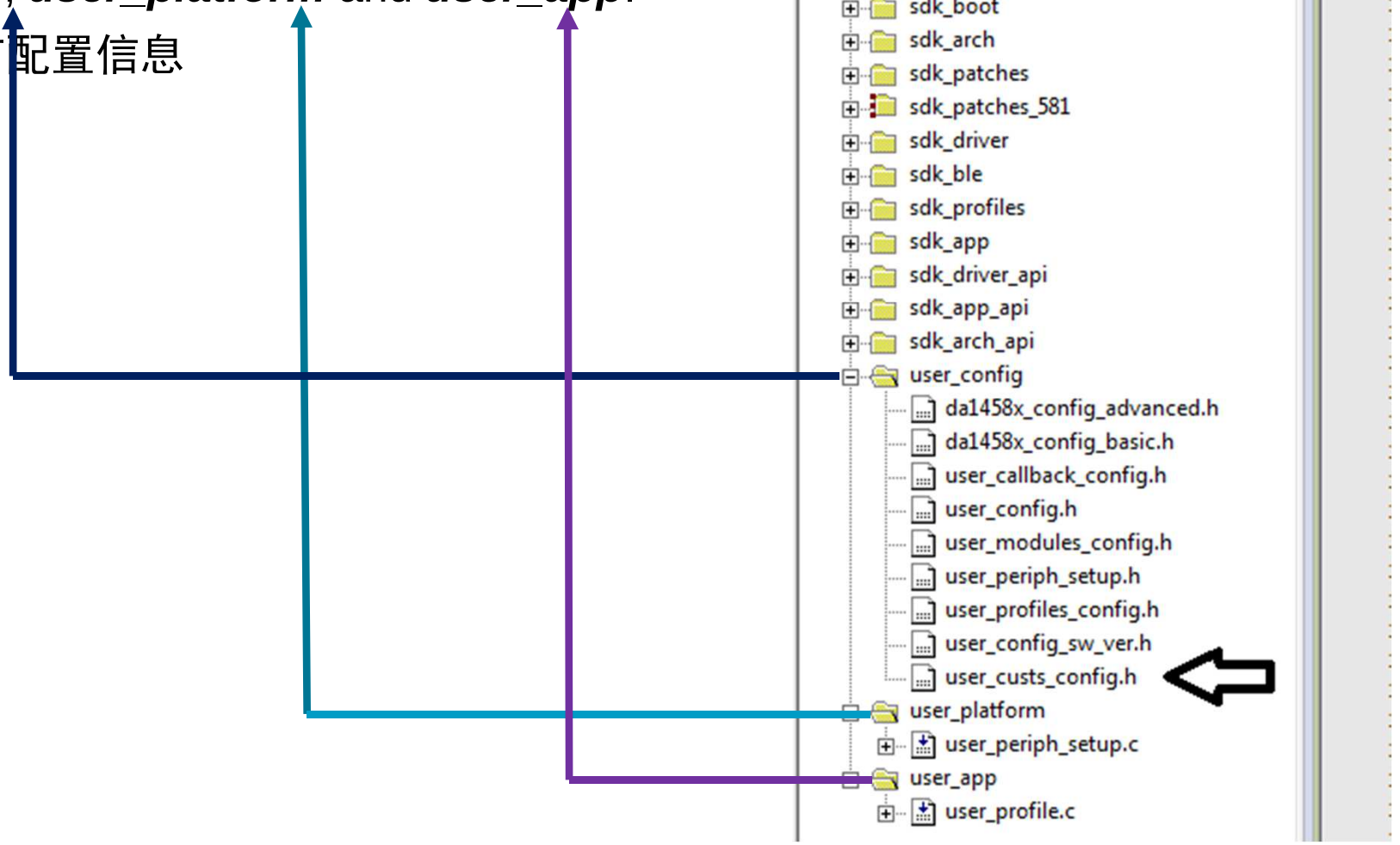


Figure: Message flow diagram

自定义服务内容

Keil 5 IDE ble_app_profile.uvprojx 工程布局

- 组 ***user_config***, ***user_platform*** and ***user_app***.
- 这些组里包含有配置信息



自定义服务内容

一些重要文件的描述

```
/* Holds DA14580/581/583 basic configuration settings.    */
da1458x_config_basic.h

/* Holds DA14580/581/583 advanced configuration settings. */
da1458x_config_advanced.h

/* Holds user specific information about software version. */
user_config_sw_ver.h

/* Defines which application modules are included or excluded from the user's application. */
user_modules_config.h

            /* The Device information application profile is excluded. */
            #define EXCLUDE_DLG_PROXR         (1)
            /* The Device information application profile is included. */
            #define EXCLUDE_DLG_CUSTS1        (0)

            /* Note:                                                                  */
            /*      This setting has no effect if the respective module is a BLE Profile        */
            /*      that is not used in the user's application.                       */


/* Callback functions that handle various events or operations. */
user_callback_config.h


/* Holds advertising parameters, connection parameters, etc. */
user_config.h
```

自定义服务内容

一些重要文件的描述

```
/* Defines which BLE profiles (Bluetooth SIG adopted or custom ones) will be included in user's application.
   each header file denotes the respective BLE profile*/
user_profiles_config.h

            #inlucde "diss.h"       // Includes Device Information Service.
            #include "custs1.h"     // Includes Custom service.


/* Defines the structure of the Custom profile database structure and
   cust_prf_funcs[] array, which contains the Custom profile API functions calls.*/
user_custs_config.h


/* Holds hardware related settings relative to the used Development Kit. */
user_periph_setup.h


/* Source code file that handles peripheral (GPIO, UART, SPI, etc.)
   configuration and initialization relative to the Development Kit.*/
user_periph_setup.c
```

自定义服务内容

按步骤添加属性

TODO 1 – 修改默认蓝牙地址 BD_ADDRESS, 地址在BLE网络上需要做到唯一.

```
/* @file da1458x_config_advanced.h */
```

```
/* copy and paste in code step 1 change the BLE device address */
#define CFG_NVDS_TAG_BD_ADDRESS              {0x19, 0x00, 0x00, 0x00, 0x00, 0x19}
```

TODO 2 – 在应用代码里检查和定义DLG_CUST1模块信息

```
/* @file user_modules_config.h */
```

```
#define EXCLUDE_DLG_SPOTAR          (1)         /* excluded */
/* copy and paste in code step 2 define DLG_CUST1 module in your application code */
#define EXCLUDE_DLG_CUSTS1          (0)         /* included */
```

TODO 3 – 检查和添加头文件cust1.h到应用代码里用于激活自定义profile

```
/* @file user_profiles_config.h */
```

```
#include "diss.h"
/* copy and paste in code step 3 add custs1.h */
#include "custs1.h"
```

自定义服务内容

按步骤添加属性

TODO 4 – 添加广播内容和修改广播设备名字

/* @file **user_config.h** */

```
/* default sleep mode. Possible values ARCH_SLEEP_OFF, ARCH_EXT_SLEEP_ON, ARCH_DEEP_SLEEP_ON
   ARCH_EXT_SLEEP_ON, ARCH_DEEP_SLEEP_ON – You cannot debug in these modes
*/
const static sleep_state_t app_default_sleep_mode = ARCH_SLEEP_OFF;
//-------------NON-CONNECTABLE & UNDIRECTED ADVERTISE RELATED COMMON -- //
/// Advertising service data
/// dev step 5 explanation of the following 3 items

#define USER_ADVERTISE_DATA ("\x03"\
            ADV_TYPE_COMPLETE_LIST_16BIT_SERVICE_IDS\
            ADV_UUID_DEVICE_INFORMATION_SERVICE\
            "\x11"\                                    /// The next section takes hex x11 = decimal 17 bytes
            ADV_TYPE_COMPLETE_LIST_128BIT_SERVICE_IDS\ /// Shows complete list of 128 bit Service IDs
            "\x2F\x2A\x93\xA6\xBD\xD8\x41\x52\xAC\x0B\x10\x99\x2E\xC6\xFE\xED") /// Your Custom Service UUID
/// Note– Custom service UUID is shown from right to left <-- EDFEC6...2F in the client LightBlue iOS app GUI
/* copy and paste in code step 4 change your advertising device name */
#define USER_DEVICE_NAME    ("A-CUST1")
```

自定义服务内容

按步骤添加属性

TODO 5 –概览现有的BLE协议对应的自定义服务属性值和特性

NAME	PROPERTIES	LENGTH	DESCRIPTION
Control Point	WRITE	1	Accept commands from peer
LED State	WRITE NO RESPONSE	1	Toggles a LED connected to a GPIO
ADC Value 1	READ, NOTIFY	2	Reads sample from an ADC channel
ADC Value 2	READ	2	Reads sample from an ADC channel
Button State	READ, NOTIFY	1	Reads the current state of a push button connected a GPIO
Indicate able	READ, INDICATE	20	Demonstrate indications
Long Value	READ, WRITE. NOTIFY	50	Demonstrate writes to long characteristic value

BLE自定义协议

按步骤添加属性

- 属性带有名字
  - 在客户端的扫描应用界面上会显示名字.
- 属性含有值
  - 最多512字节，固定或可变长度的数组.
- 属性带有句柄
  - 用于client端访问特定属性的接口.
- 属性带有描述
  - <<UUID>>, 描述具体数值的意义
  - 由 GAP, GATT, 或者用户自定义属性标准来决定
  - 举例来说，“Accept commands from peer”是关于控制点属性的描述
- 属性带有访问权限
  - 读，写，通知等.

自定义服务内容

按步骤添加属性

TODO 6 – UUID信息

```
/* @file user_custs_config.h */
```

```
/* step 5 and step 6 info:: 128 bit Service UUID this is displayed from Right to Left in the client scanner device */
#define DEF_CUST1_SVC_UUID_128 {0x2F, 0x2A, 0x93, 0xA6, 0xBD, 0xD8, 0x41, 0x52, 0xAC, 0x0B, 0x10, 0x99, 0x2E, 0xC6, 0xFE, 0xED} /* Displayed as EDFEC62E99100BAC5241D8BDA6932A2F */
```

TODO 7 – 添加控制点属性的UUID

```
/* @file user_custs_config.h */
```

```
#define DEF_CUST1_LONG_VALUE_UUID_128      {0x8C, 0x09, 0xE0, 0xD1, 0x81, 0x54, 0x42, 0x40, 0x8E, 0x4F, 0xD2, 0xB3, 0x77, 0xE3, 0x2A, 0x77}
/* copy and paste in code step 7 define your control point */
#define DEF_CUST1_YOUR_CTRL_POINT_UUID_128 {0x34, 0x33, 0x32, 0x31, 0x30, 0x29, 0x28, 0x27, 0x26, 0x25, 0x24, 0x23, 0x22, 0x21, 0x20, 0x19}
```

自定义服务内容

按步骤添加属性

TODO 8 – 添加自定义控制点数据的长度

```
/* @file user_custs_config.h */
```

```
#define DEF_CUST1_LONG_VALUE_CHAR_LEN       50
/* copy and paste in code step 8 define your control point data length */
#define DEF_CUST1_YOUR_CTRL_POINT_CHAR_LEN  1
```

TODO 9 – 添加自定义属性的注释字符串

```
/* @file user_custs_config.h */
```

```
#define CUST1_LONG_VALUE_CHAR_USER_DESC     "Long Value"
/* copy and paste in code step 9 define your characteristic description name */
#define CUST1_YOUR_CONTROL_POINT_USER_DESC  "Your Ctrl Point"
```

自定义服务内容

按步骤添加属性

TODO 10 – 添加自定义服务数据库里属性的枚举列表值

```
/* @file user_custs_config.h */
```

```
enum
{
    …
    CUST1_IDX_LONG_VALUE_CHAR,
    CUST1_IDX_LONG_VALUE_VAL,
    CUST1_IDX_LONG_VALUE_NTF_CFG,
    CUST1_IDX_LONG_VALUE_USER_DESC,

    /* copy and paste in code step 10 add your characteristic */
    CUST1_IDX_YOUR_CONTROL_POINT_CHAR,
    CUST1_IDX_YOUR_CONTROL_POINT_VAL,
    CUST1_IDX_YOUR_CONTROL_POINT_USER_DESC,
    CUST1_IDX_NB
};
```

自定义服务内容

按步骤添加属性

TODO 11 – 申明和添加自定义服务属性的值

```
/* @file user_custs_config.h */
static uint8_t CUST1_LONG_VALUE_UUID_128[ATT_UUID_128_LEN]       = DEF_CUST1_LONG_VALUE_UUID_128;
/* copy and paste in code step 11 declare and assign custom server attribute value */
static uint8_t CUST1_YOUR_CTRL_POINT_UUID_128[ATT_UUID_128_LEN]  = DEF_CUST1_YOUR_CTRL_POINT_UUID_128;
```

TODO 12 – 添加包含访问权限，句柄和UUID的属性说明

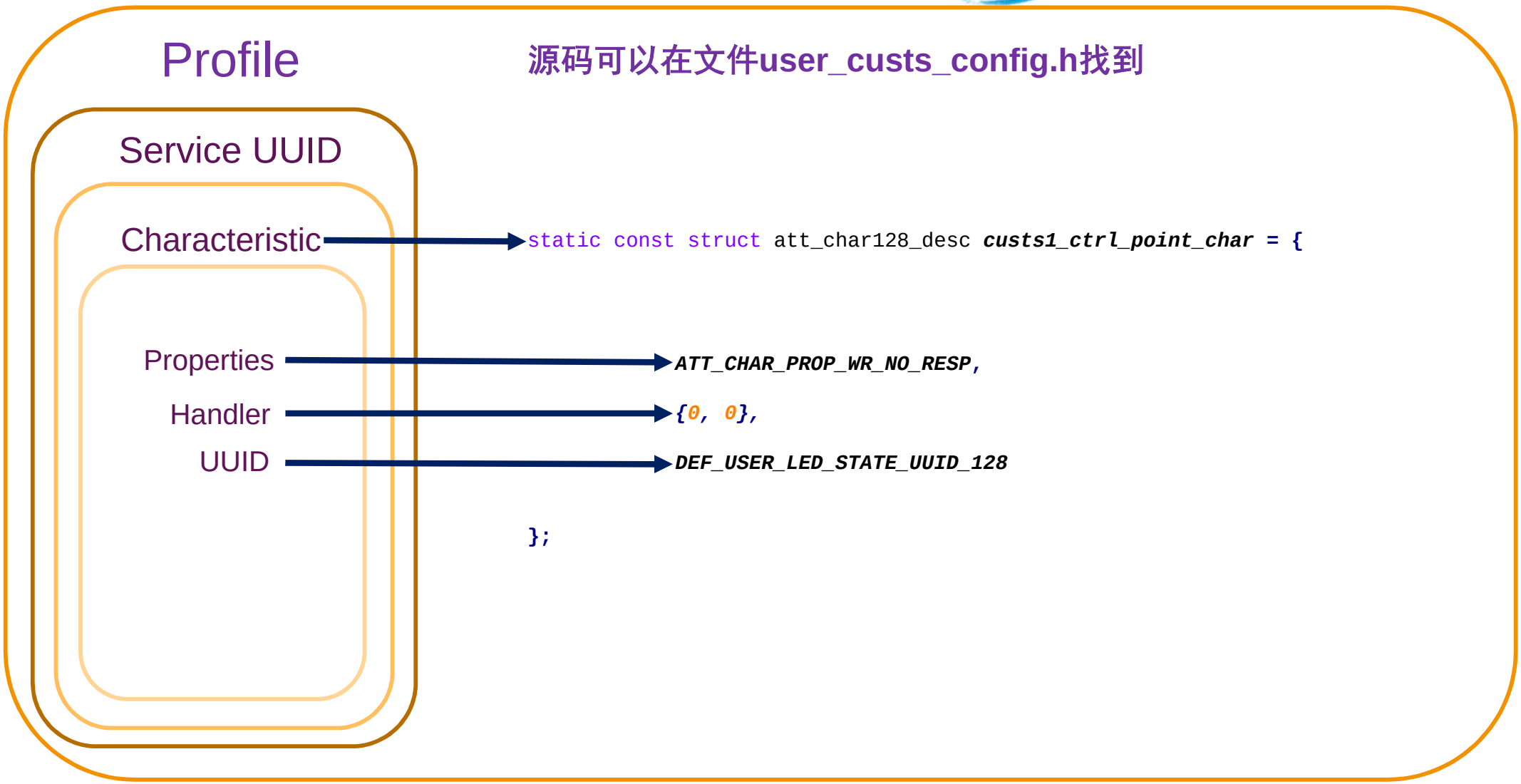
```
/* @file user_custs_config.h */
static const struct att_char128_desc custs1_long_value_char       = {ATT_CHAR_PROP_RD | ATT_CHAR_PROP_WR |
ATT_CHAR_PROP_NTF,
                                                                     {0, 0},
                                                                     DEF_CUST1_LONG_VALUE_UUID_128};

/* copy and paste in code step 12 */
/* Add your characteristic description with permission properties, handler and UUID */

static const struct att_char128_desc custs1_your_ctrl_point_char  = {ATT_CHAR_PROP_WR,
                                                                     {0, 0},
                                                                     DEF_CUST1_YOUR_CTRL_POINT_UUID_128};
```

BLE自定义协议

按步骤添加属性



自定义服务内容

按步骤添加属性

TODO 13 –在用户自定义服务数据库里，添加自定义属性申明，值和注释说明

```
/* @file user_custs_config.h */
```

```
/// Full CUSTOM1 Database Description - Used to add attributes into the database
static const struct attm_desc_128 custs1_att_db[CUST1_IDX_NB] =
{
    ...
    // Long Value Characteristic Declaration
    [CUST1_IDX_LONG_VALUE_CHAR]         = {(uint8_t*)&att_decl_char, ATT_UUID_16_LEN, PERM(RD, ENABLE),
                                           sizeof(custs1_long_value_char), sizeof(custs1_long_value_char),
                                           (uint8_t*)&custs1_long_value_char},
    // Long Value Characteristic Value
    [CUST1_IDX_LONG_VALUE_VAL]          = {CUST1_LONG_VALUE_UUID_128, ATT_UUID_128_LEN, PERM(RD, ENABLE) | PERM(WR,
ENABLE) | PERM(NTF, ENABLE),
                                           DEF_CUST1_LONG_VALUE_CHAR_LEN, 0, NULL},
    // Long Value Client Characteristic Configuration Descriptor
    [CUST1_IDX_LONG_VALUE_NTF_CFG]      = {(uint8_t*)&att_decl_cfg, ATT_UUID_16_LEN, PERM(RD, ENABLE) | PERM(WR,
ENABLE),
                                           sizeof(uint16_t), 0, NULL},
    // Long Value Characteristic User Description
    [CUST1_IDX_LONG_VALUE_USER_DESC]    = { (uint8_t*)&att_decl_user_desc, ATT_UUID_16_LEN, PERM(RD, ENABLE),
                                           sizeof(CUST1_LONG_VALUE_CHAR_USER_DESC) - 1,
sizeof(CUST1_LONG_VALUE_CHAR_USER_DESC) - 1, CUST1_LONG_VALUE_CHAR_USER_DESC},
```

自定义服务内容

按步骤添加属性

TODO 13 – 在用户自定义服务数据库里，添加自定义属性申明，值和注释说明

```
/* @file user_custs_config.h */
```

```
/* copy and paste in code step 13 add your characteristic declaration, value and description in database attributes
*/

    // Your Control Point Characteristic Declaration
    [CUST1_IDX_YOUR_CONTROL_POINT_CHAR]      = {(uint8_t*)&att_decl_char, ATT_UUID_16_LEN, PERM(RD, ENABLE),
                                              sizeof(custs1_your_ctrl_point_char),
sizeof(custs1_your_ctrl_point_char), (uint8_t*)&custs1_your_ctrl_point_char},
    // Your Control Point Characteristic Value
    [CUST1_IDX_YOUR_CONTROL_POINT_VAL]       = {CUST1_YOUR_CTRL_POINT_UUID_128, ATT_UUID_128_LEN, PERM(WR, ENABLE),
                                              DEF_CUST1_YOUR_CTRL_POINT_CHAR_LEN, 0, NULL},
    // Your Control Point Characteristic User Description
    [CUST1_IDX_YOUR_CONTROL_POINT_USER_DESC] = {(uint8_t*)&att_decl_user_desc, ATT_UUID_16_LEN, PERM(RD, ENABLE),
                                              sizeof(CUST1_YOUR_CONTROL_POINT_USER_DESC) - 1,
sizeof(CUST1_YOUR_CONTROL_POINT_USER_DESC) - 1, CUST1_YOUR_CONTROL_POINT_USER_DESC},
};
```

自定义服务内容

按步骤添加属性

TODO 13 – 在用户自定义服务数据库里，添加自定义属性申明，值和注释说明

```
/* @file user_custs_config.h */
```

```
/* copy and paste in code step 13 add your characteristic declaration, value and description in database attributes
*/

    // Your Control Point Characteristic Declaration
    [CUST1_IDX_YOUR_CONTROL_POINT_CHAR]      = {(uint8_t*)&att_decl_char, ATT_UUID_16_LEN, PERM(RD, ENABLE),
                                              sizeof(custs1_your_ctrl_point_char),
sizeof(custs1_your_ctrl_point_char), (uint8_t*)&custs1_your_ctrl_point_char},
    // Your Control Point Characteristic Value
    [CUST1_IDX_YOUR_CONTROL_POINT_VAL]       = {CUST1_YOUR_CTRL_POINT_UUID_128, ATT_UUID_128_LEN, PERM(WR, ENABLE),
                                              DEF_CUST1_YOUR_CTRL_POINT_CHAR_LEN, 0, NULL},
    // Your Control Point Characteristic User Description
    [CUST1_IDX_YOUR_CONTROL_POINT_USER_DESC] = {(uint8_t*)&att_decl_user_desc, ATT_UUID_16_LEN, PERM(RD, ENABLE),
                                              sizeof(CUST1_YOUR_CONTROL_POINT_USER_DESC) - 1,
sizeof(CUST1_YOUR_CONTROL_POINT_USER_DESC) - 1, CUST1_YOUR_CONTROL_POINT_USER_DESC},
};
```

自定义服务内容

按步骤添加GATT命令

TODO 14 – 在文件user_custs1_impl.c 里加入GATT 命令的句柄函数

```
/* @file user_custs1_impl.c */
```

```
/**
 ****************************************************************************************
 * @brief User defined led state value write indication handler.
 * @param[in] msgid   Id of the message received.
 * @param[in] param   Pointer to the parameters of the message.
 * @param[in] dest_id ID of the receiving task instance.
 * @param[in] src_id  ID of the sending task instance.
 * @return void
 ****************************************************************************************
*/
void user_led_wr_ind_handler(ke_msg_id_t const msgid,
                             struct custs1_val_write_ind const *param,
                             ke_task_id_t const dest_id,
                             ke_task_id_t const src_id)
{
    uint8_t led_state = 0;
    memcpy(&led_state, &param->value[0], param->length);

    if (led_state == LED_ON)
        GPIO_SetActive(GPIO_LED_PORT, GPIO_LED_PIN);
    else if (led_state == LED_OFF)
        GPIO_SetInactive(GPIO_LED_PORT, GPIO_LED_PIN);
}
```

自定义服务内容

自定义服务内容

TODO 14 – 在文件user_peripheral.c 里的函数user_catch_rest_hndl() 加入switch case 语句

```
/* @file user_peripheral.c */
```

```
void user_catch_rest_hndl(ke_msg_id_t const msgid,
                          void const *param,
                          ke_task_id_t const dest_id,
                          ke_task_id_t const src_id)
{
    switch(msgid)
    {
        case CUSTS1_VAL_WRITE_IND:
        {
            struct custs1_val_write_ind const *msg_param = (struct custs1_val_write_ind const *)(param);

            switch (msg_param->handle)
            {
                case USER_IDX_LED_STATE_VAL:
                    user_led_wr_ind_handler(msgid, msg_param, dest_id, src_id);
                    break;

                default:
                    break;
            }
        } break;
```

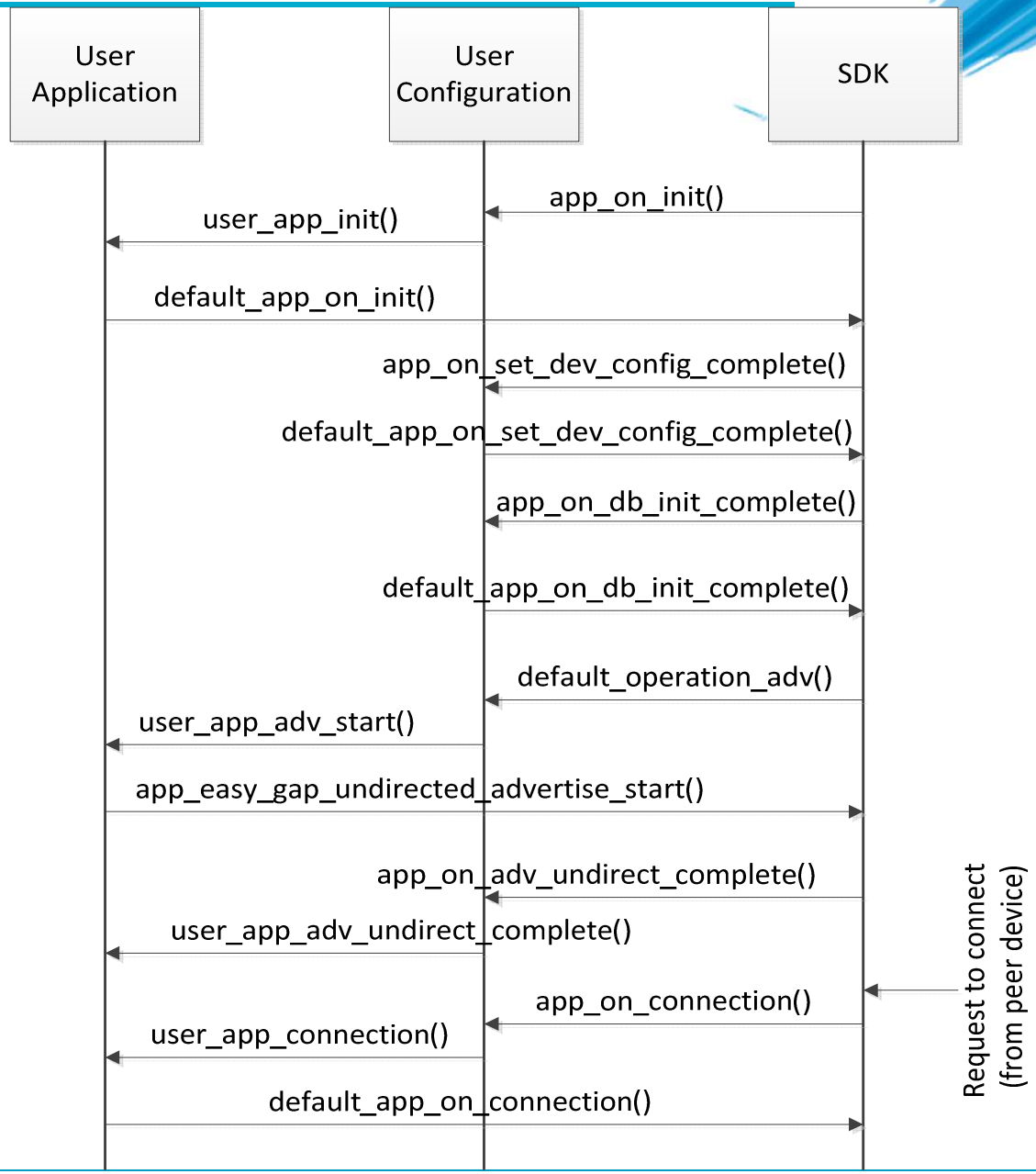
自定义服务内容

工作机理?

- 在整个BLE应用交互周期内，会有一些事件，需要用特定的方式来处理
- 在处理特定事件时，SDK可以很灵活的调用默认的或者用户自定义的接口来处理特定事件（**`user_catch_rest_hndl`**）
- 在处理机制上，SDK提供注册callback函数的接口，来处理每个消息或操作
- 在用户空间的的头文件，“user_callback_config.h”，记录callback函数的注册接口.

自定义服务内容

抽象代码流程



自定义服务内容

user_callback_config.h 重要函数讨论

```
static const  struct arch_main_loop_callbacks user_app_main_loop_callbacks = {
    .app_on_init            = user_app_init,
    .app_on_ble_powered     = NULL,
    .app_on_sytem_powered   = NULL,
    .app_before_sleep       = NULL,
    .app_validate_sleep     = NULL,
    .app_going_to_sleep     = NULL,
    .app_resume_from_sleep  = NULL,
};
```

```
void user_app_init(void)
{
    // Initialize Manufacturer Specific Data
    mnf_data_init();
    // Initialize default services and set sleep mode
    default_app_on_init();
}
```

自定义服务内容

概览user_callback_config.h

```
static const struct app_callbacks user_app_callbacks = {
    // Handle connection request indication, if no connection has been established restart advertising
    .app_on_connection              = user_app_connection,
    .app_on_disconnect              = user_app_disconnect, // Restart Advertising
    /* Add the first required service in the database
       if database initialized then
       No service to add in the DB -> Start Advertising */
    .app_on_set_dev_config_complete = default_app_on_set_dev_config_complete,
    /* If advertising was canceled for any reason other then connection establishment
       then update advertising data and start advertising again */
    .app_on_adv_undirect_complete   = user_app_adv_undirect_complete,
    // database initialization is completed, then set the initial values of service characteristics programmatically
    .app_on_db_init_complete        = default_app_on_db_init_complete,
    .app_on_scanning_completed      = NULL, // NULL indicated this indication will not be handled by Dialog SDK;
    .app_on_adv_report_ind          = NULL, // either implement it or use the existing code based on your requirement
};

// Handles the messages that are not handled by the SDK internal mechanisms.
static const catch_rest_event_func_t app_process_catch_rest_cb = (catch_rest_event_func_t)user_catch_rest_hndl;
```

自定义服务内容

user_custs_config.h

添加custom1 server函数的callback参数列表.

```
.
/// Custom1/2 server function callback table this is linking point of your database and DA1458x SDK5.0.3
static const struct cust_prf_func_callbacks cust_prf_funcs[] =
{
#if (BLE_CUSTOM1_SERVER)
   {  TASK_CUSTS1,
      custs1_att_db,
      CUST1_IDX_NB,
      #if (BLE_APP_PRESENT)
      app_custs1_create_db, app_custs1_enable,
      #else
      NULL, NULL,
      #endif
      custs1_init, NULL
   },
#endif
#if (BLE_CUSTOM2_SERVER)
   {  TASK_CUSTS2,
      NULL,
      0,
      #if (BLE_APP_PRESENT)
      app_custs2_create_db, app_custs2_enable,
      #else
      NULL, NULL,
      #endif
      custs2_init, NULL
   },
#endif
   {TASK_NONE, NULL, 0, NULL, NULL, NULL, NULL},   // DO NOT MOVE. Must always be last
};
```

```
/// Structure of custom profile call back function table.
struct cust_prf_func_callbacks
{
    /// Profile Task ID.
    enum KE_TASK_TYPE       task_id;
    /// pointer to the custom database table defined by user
    const struct attm_desc_128 *att_db;
    /// max number of attributes in custom database
    const uint8_t max_nb_att;
    /// Pointer to the custom database create function defined by
user
    prf_func_void_t         db_create_func;
    /// Pointer to the custom profile enable function defined by user
    prf_func_uint16_t       enable_func;
    /// Pointer to the custom profile initialization function
    prf_func_void_t         init_func;
    /// Pointer to the validation function defined by user
    prf_func_validate_t     value_wr_validation_func;
};
```

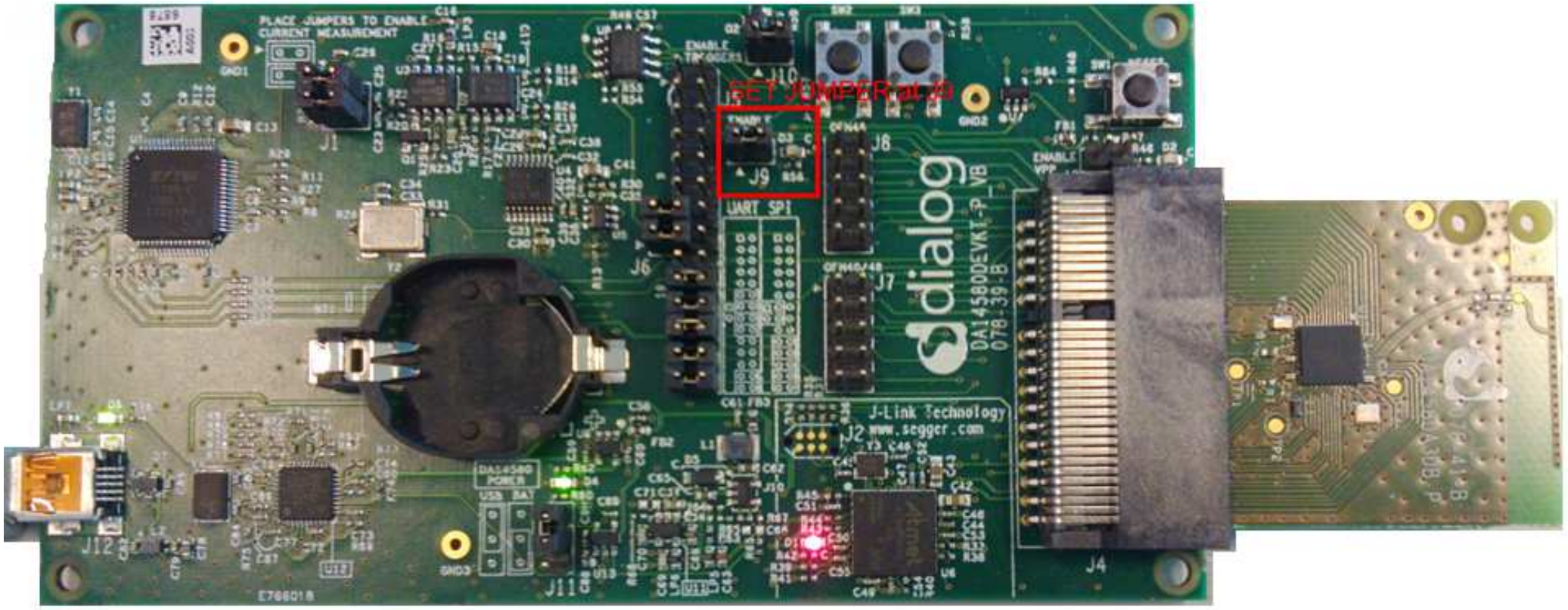
BLE内容

你会看到输出内容

- The LightBlue iOS 应用可以用来连接 iPad/iPod/iPhone 设备，其中iPad/iPod/iPhone作为BLE主控，而应用设备则作为BLE外设，名字内容由“USER_DEVICE_NAME”字段定义.
- 设备信息服务（DISS）默认需要被添加。在一些扫描设备里，这个服务会被作为名字服务，有特定的UUID字段（0A 18）.
- 连接建上之后，设备的服务属性可以被手机访问提取.

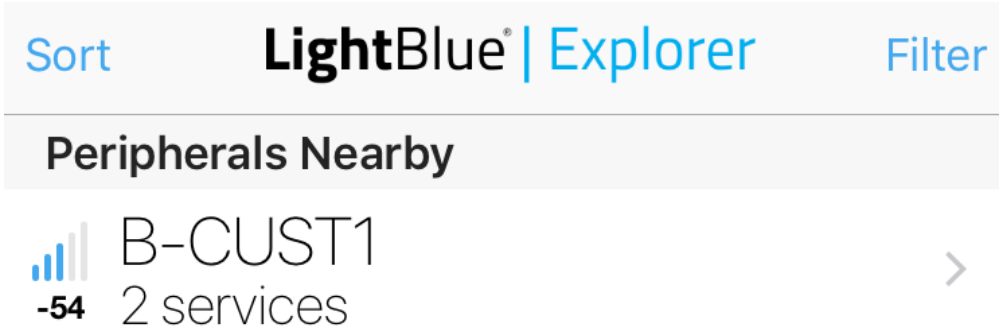
BLE内容

DA14580 DK-Pro Configuration

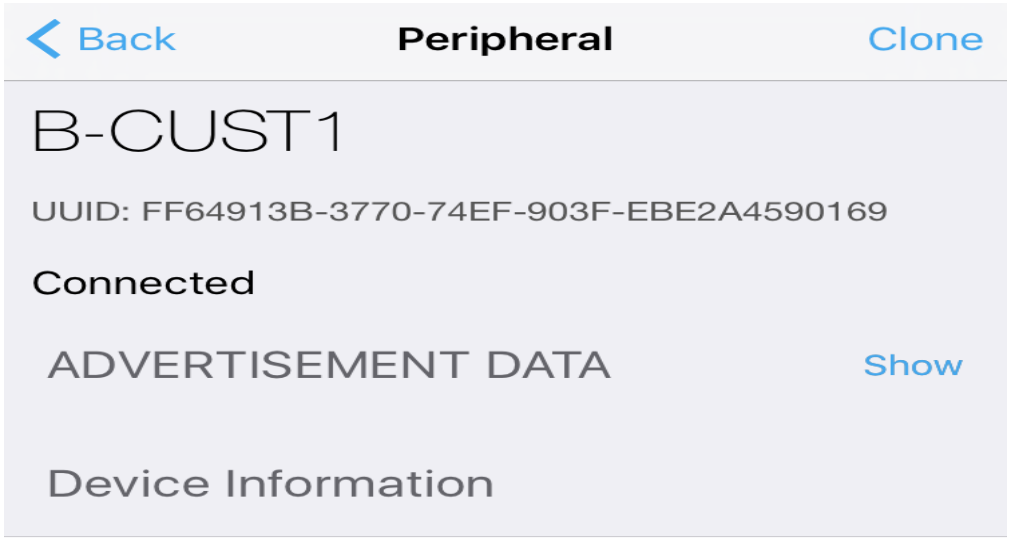


BLE内容

你将会看到的输出结果...



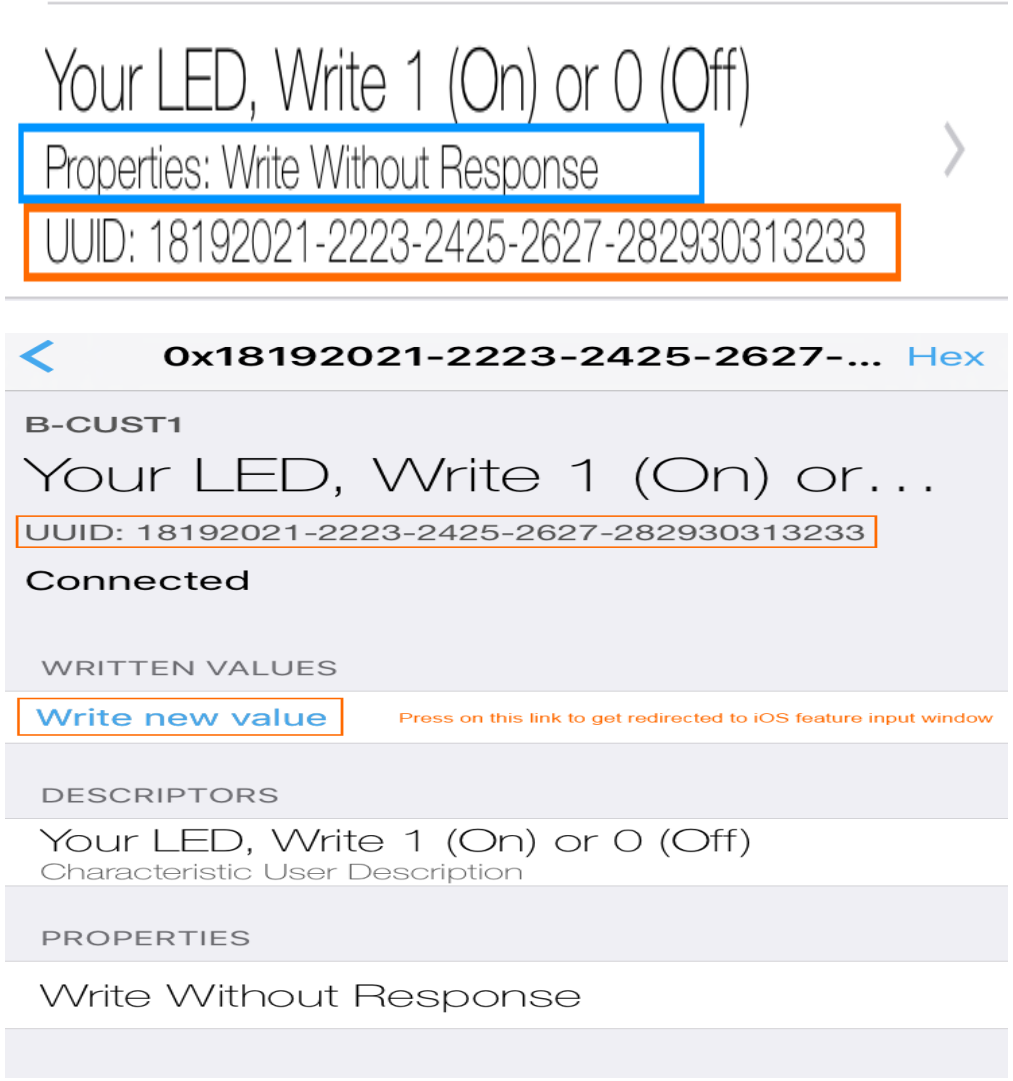
1. 你的设备在广播



2. 你的设备连接成功

BLE内容

你将会看到的输出结果...

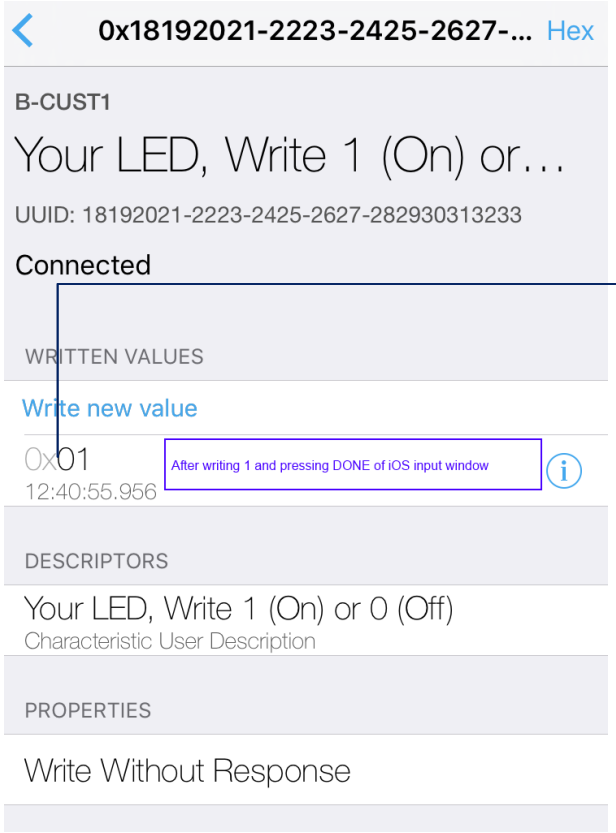


3. 你的LED状态属性

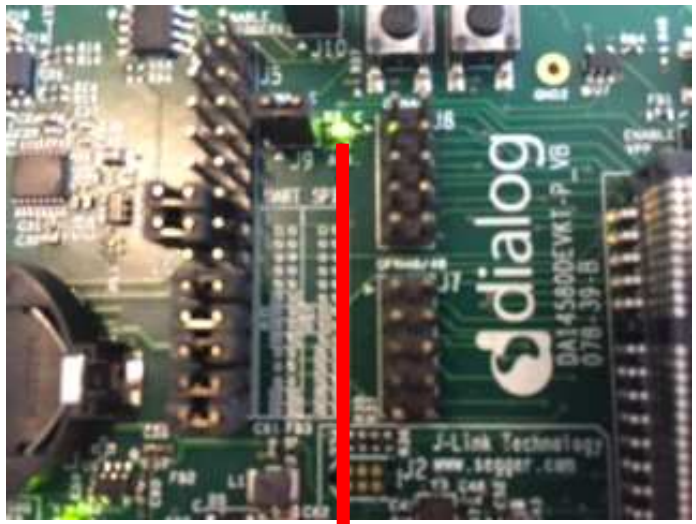
4. 橙色行高亮处说明“可以写入新的属性”

BLE内容

你将会看到的输出结果...



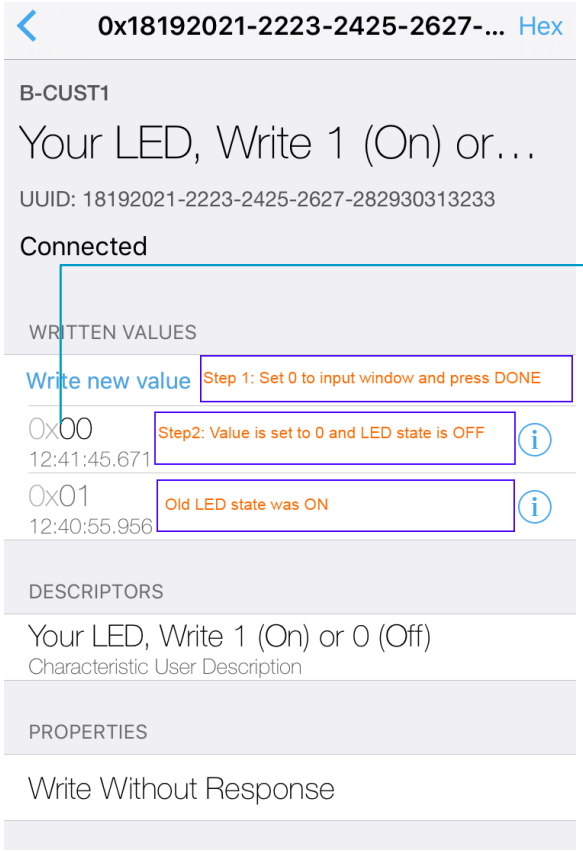
5. 确认在IOS APP里写入了0x01



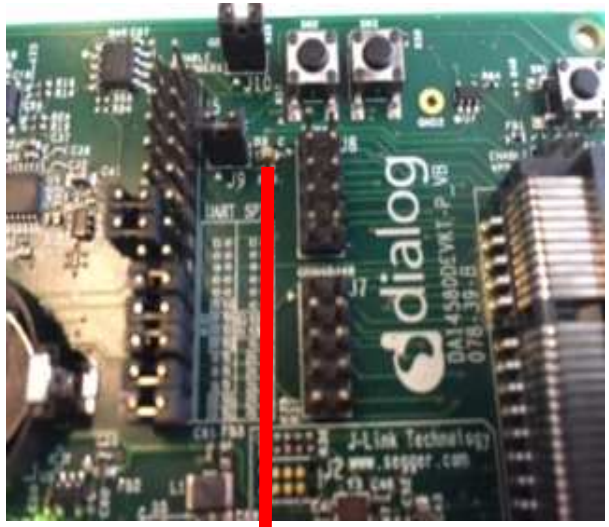
6. 在开发板中确认LED状态

BLE内容

你将会看到的输出结果...



7.确认在IOS APP里写入了0x00



8.在开发板中确认LED状态

BLE 内容

输出内容

- 注意: 在当前或者之后的实例里，设备可以被连接。设备被连上之后意味着别的扫描设备将无法定位当前设备—建议你只连接自己的设备.

- **注意**: 一些扫描设备（特别是Apple设备）也许不会在显示端更新获得的名字-为了修正这一点，有必要重启蓝牙功能。

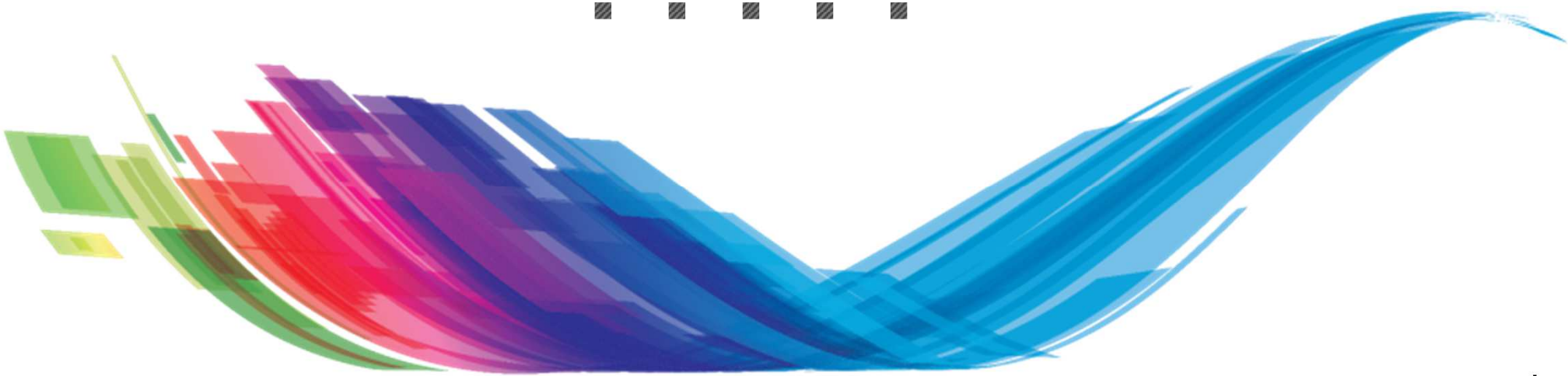
BLE 内容

参考文档

- https://developer.bluetooth.org/gatt/Pages/default.aspx
- https://www.bluetooth.com/specifications/adopted-specifications
- http://support.dialog-semiconductor.com/connectivity
- https://www.wikiwand.com/en/Universally_unique_identifier
- **Register with Dialog semiconductor to get enormous development support**
  - http://support.dialog-semiconductor.com/user/register

The Power To Be…

?????



…personal
…portable
…connected