

CMT453x服务和特征值的配置操作说明

简介

本文档介绍CMT453x系列蓝牙芯片自定义服务、特征值操作流程，包括修改服务和特征值的UUID、权限等参数以及添加自定义服务和自定义特征值的方法步骤。本文档目的在于让开发者能够快速熟悉自定义蓝牙服务和特征值的修改或添加的方式，以降低开发难度。

免责声明

深圳市华普微电子股份有限公司保留在不另行通知的情况下，更改产品以提升其可靠性、功能或设计的权利。本公司亦不承担因使用此处所述产品或电路而引致的任何责任。

关于涉及生命维持设备的应用

深圳市华普微电子股份有限公司的产品并不适用于生命维持设备、装置或系统，因为这些产品的故障可能会导致人身伤害。使用或销售本产品作上述用途的客户须自行承担风险，并同意就因使用或销售不当而引致的任何损害，向本公司作出全面赔偿。

联系方式

深圳市华普微电子股份有限公司

地址：深圳市南山区西丽街道万科云城三期8栋A座30层

电话：+86-0755-82973805

邮箱：sales@hoperf.com

网址：<http://www.hoperf.cn>

目录

1	修改服务和特征值的UUID和权限	4
1.1	修改服务和特征值的UUID值	4
1.2	修改特征值的权限.....	5
2	添加自定义的特征值.....	9
2.1	添加特征值的UUID值	9
2.2	添加特征值的枚举声明	9
2.3	添加特征值到DB	10
2.4	从机接收数据回调函数	10
2.5	从机发送数据	13
2.6	主机(手机)读取从机数据	14
3	添加自定义服务	16
4	历史版本.....	22

1 修改服务和特征值的UUID和权限

1.1 修改服务和特征值的UUID值

本文档以 *ble_rtds_peripheral* 例程为说明对象，服务和特征值的 UUID 值定义位于 *app_rdtss.h*。原始的服务和特征值的 UUID 值定义如下：

服务的 UUID 宏定义：

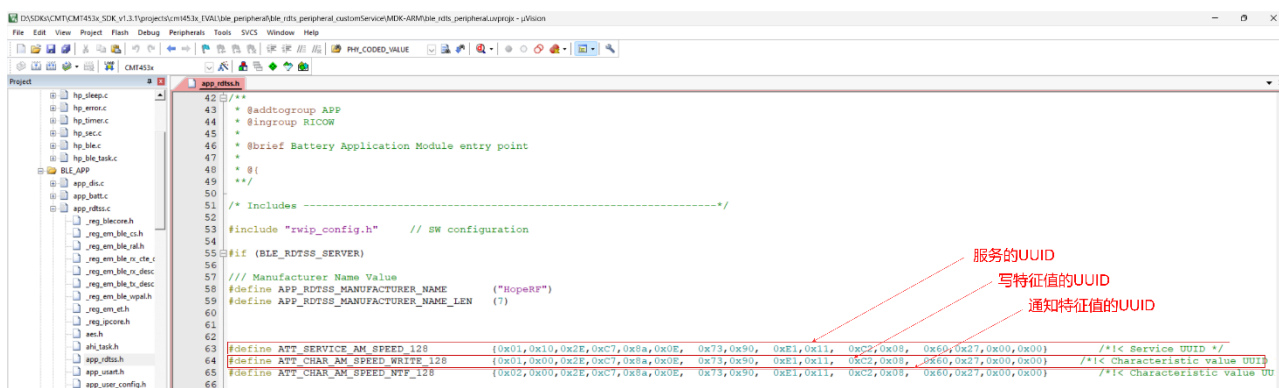
ATT_SERVICE_AM_SPEED_128

服务的写特征值的 UUID 宏定义：

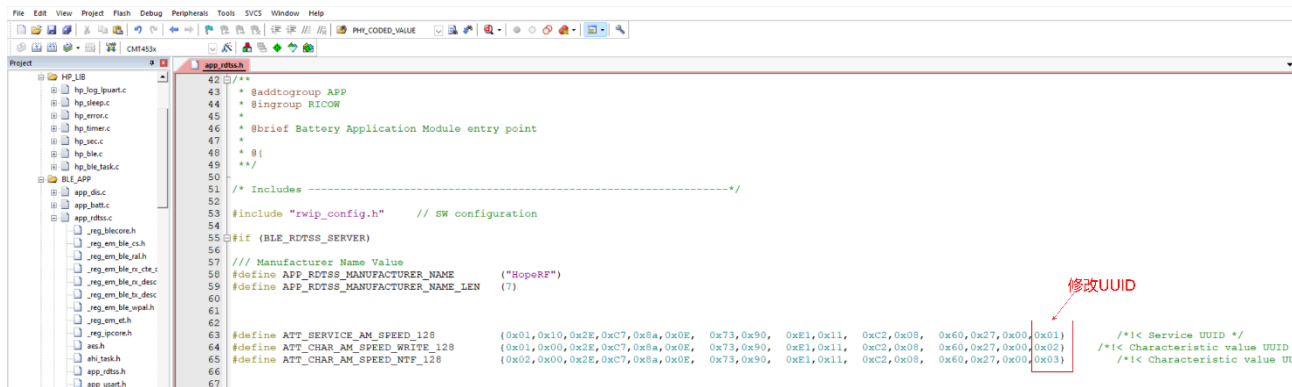
TT_CHAR_AM_SPEED_WRITE_128

服务的通知特征值的 UUID 宏定义：

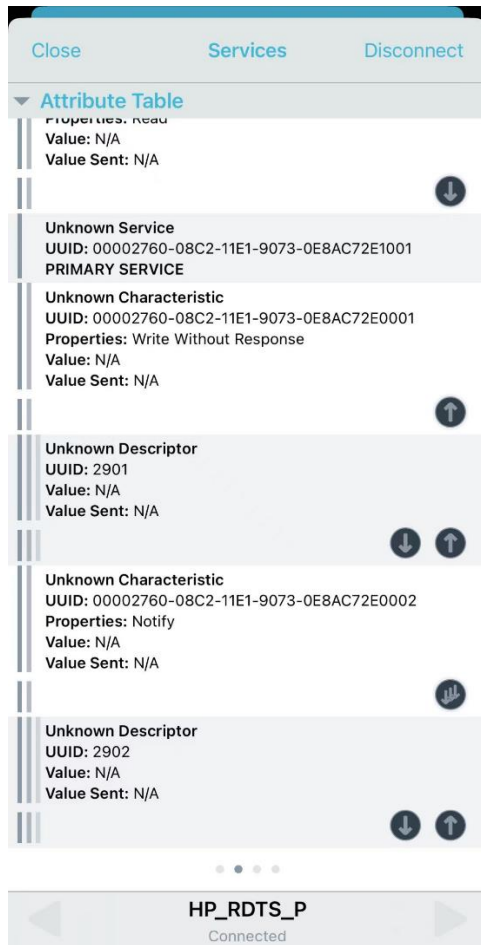
TT_CHAR_AM_SPEED_NTF_128



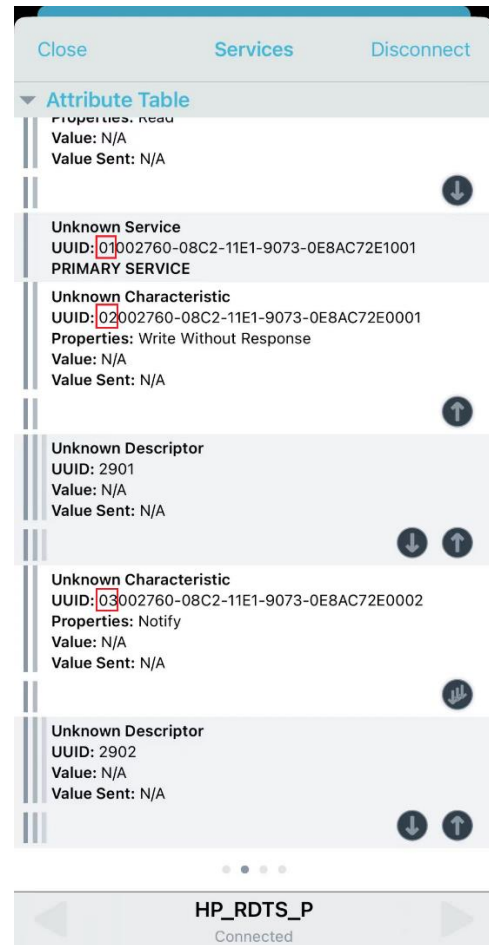
修改服务和特征值的 UUID 值：修改宏定义数组内的成员数值即可修改 UUID 值



修改服务和特征值的 UUID 后的实际效果如下所示：



修改 UUID 前



修改 UUID 后

注意:

1. 一般在手机 App 上看到的 UUID 与程序中配置的顺序是相反的。譬如，手机 APP 上服务 UUID 一般显示为：0x00002760-08c2-11e1-9073-0e8ac72e1001。
2. 16bit UUID 数传例程的宏定义在 `app_rdtss_16bit.h`，修改方式和此方式一样。

1.2 修改特征值的权限

本文档以 `ble_rdtss_peripheral` 例程为说明对象，数传例程蓝牙服务和特征值的权限，定义位于 `app_rdtss.c` 文件的结构体 `const struct attm_desc_128 app_rdtss_att_db[]` 里面。

特征值权限定义结构体：attm_desc_128

```
/// Internal 128bits UUID service description
struct attm_desc_128
{
    /// 128 bits UUID LSB First
    uint8_t uuid[ATT_UUID_128_LEN];
    /// Attribute Permissions (@see enum attm_perm_mask)
    uint16_t perm;
    /// Attribute Extended Permissions (@see enum attm_value_perm_mask)
    uint16_t ext_perm;
    /// Attribute Max Size
    /// note: for characteristic declaration contains handle offset
    /// note: for included service, contains target service handle
    uint16_t max_size;
};
```

说明：第一项是UUID；第二项perm是属性表权限，主要用于定义属性表本身的权限，比如读/写/通知；第三项ext_perm是扩展权限，主要用于定义特征值属性，比如长度；第四项为最大长度值。

服务定义(service definition)需要包含服务声明(service declaration)，同时还可以有包含定义(include definitions)和特征值定义(characteristic definitions)。所有在服务定义中的包含定义和特征值定义都被认为是服务的一部分。一个服务定义中可以有任意多个包含定义和特征值定义。

例程中，服务定义的结构体为app_rdtss_att_db[]

```

47  /* Includes ----- */
48  #include "co_utils.h"
49  #include "app_rdtss.h"           // rdtss Application Module Definitions
50  #include "rdtss_task.h"         // Device Information Profile Functions
51  #include "gapm_task.h"          // GAP Manager Task API
52  #include "hp_ble.h"
53  #include "prf.h"
54  #include "ke_timer.h"
55  #include "stdio.h"
56  #include "rdtss.h"
57  #include "app_usart.h"
58
59  /* Private typedef ----- */
60  /* Private define ----- */
61  /* Private constants ----- */
62  /* Private variables ----- */
63
64  const uint8_t app_rdtss_svc_uuid[16] = ATT_SERVICE_AM_SPEED_128;
65  const struct attm_desc_128 app_rdtss_att_db[RTSS_IDX_NB] =
66  {
67      /* Service Declaration */
68      [0] = {{0x00,0x28},          PERM(RD, ENABLE),          0,          0},
69
70      /* Characteristic Declaration */
71      [1] = {{0x03,0x28},          PERM(RD, ENABLE),          0,          0},
72      /* Characteristic Value */
73      [2] = {ATT_CHAR_AM_SPEED_WRITE_128, PERM(WRITE_COMMAND, ENABLE), PERM(RI, ENABLE) | PERM_VAL(UUID_LEN, 0x02), 0x200},
74      /* Client Characteristic Configuration */
75      [3] = {{0x02,0x29},          PERM(RD, ENABLE) | PERM(WRITE_REQ, ENABLE), PERM(RI, ENABLE),          20},
76
77      /* Characteristic Declaration */
78      [4] = {{0x03,0x28},          PERM(RD, ENABLE),          0,          0},
79      /* Characteristic Value */
80      [5] = {ATT_CHAR_AM_SPEED_NTF_128, PERM(NTF, ENABLE) | PERM(RD, ENABLE) | PERM(WRITE_COMMAND, ENABLE), PERM(RI, ENABLE) | PERM_VAL(UUID_LEN, 0x02), 0x200},
81      /* Client Characteristic Configuration */
82      [6] = {{0x02,0x29},          PERM(RD, ENABLE) | PERM(WRITE_REQ, ENABLE), PERM(RI, ENABLE),          20},
83  };

```

其中每一项的说明如下：

- Service Declaration**
 服务声明(Service declaration)其 UUID 为 0x2800 或 0x2801，分别代表 Primary Service 和 Secondary Service。权限为 Read Only。
- Characteristic Declaration**
 特征声明，其 UUID 为 0x2803，权限为 Read Only。
- Characteristic Value declaration**
 特征值声明包含该特征的值，它是特征声明之后的第一个属性。所有的特征定义都应该有一个特征值声明。特征值可以为 16 位或 128 位 UUID。
- Characteristic User Description**
 特征用户描述声明 UUID 为 0x2901，它是一个可选的特征描述符，定义一个可变大小的 UTF-8 字符串，它是特征值的用户文本描述。
- Client Characteristic Configuration**
 客户端特征配置声明 UUID 为 0x2902，它是一个可选的特性描述符，该描述符定义了特定的客户端如何来配置该特征值。

例程中，通过如下代码给 UUID 为 ATT_CHAR_AM_SPEED_WRITE_128 的特征值定义了写权限，

特征值UUID为128bit长度。

```
[2] = {ATT_CHAR_AM_SPEED_WRITE_128, PERM(WRITE_COMMAND, ENABLE), PERM(RI, ENABLE) | PERM_VAL(UUID_LEN, 0x02), 0x200},
```

例程里通过如下代码给UUID ATT_CHAR_AM_SPEED_NTF_128定义了通知权限，这个特征值UUID为128bit的。

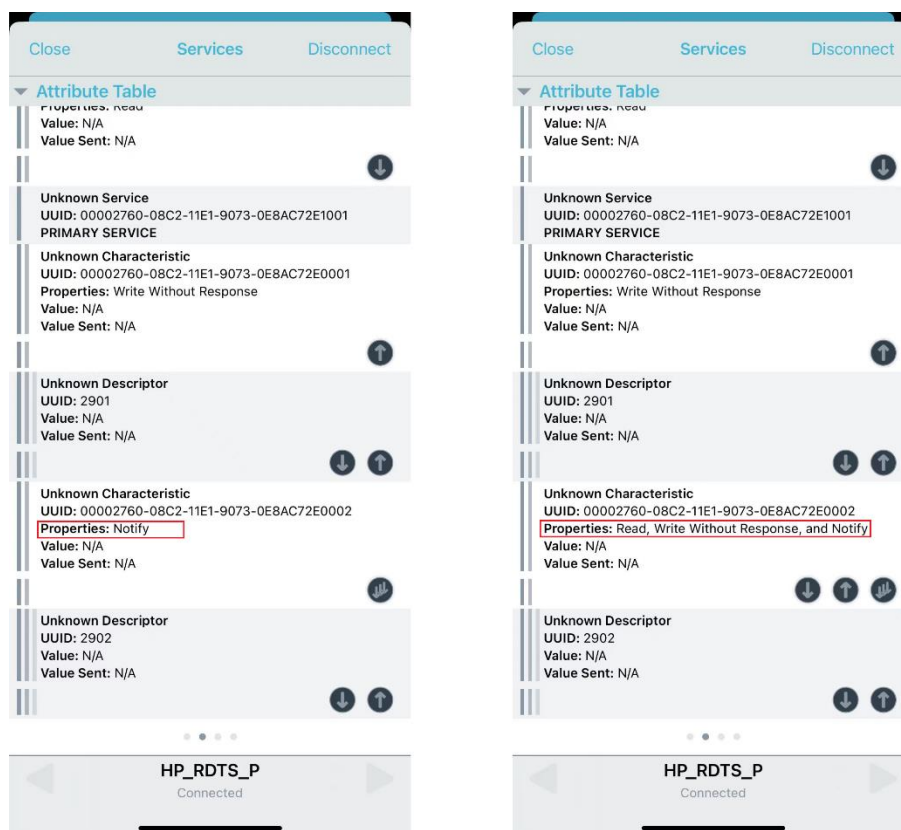
```
[5] = {ATT_CHAR_AM_SPEED_NTF_128, PERM(NTF, ENABLE), PERM(RI, ENABLE) | PERM_VAL(UUID_LEN, 0x02), 0x200 },
```

- 我们可以使用的权限有：RD（读），WRITE_COMMAND（不带回应写），WRITE_REQ（带回应写），NTF（通知），IND（带回应通知），添加方式为：在特征值的属性上添加调用函数：PERM(RD, ENABLE)，RD表示的是读权限。
- 例如：我们希望给上面的特征值 ATT_CHAR_AM_SPEED_NTF_128，在原有的通知选项再加上读写权限，我们应该按如下修改：

```
[5] = {ATT_CHAR_AM_SPEED_NTF_128, PERM(NTF, ENABLE) | PERM(RD, ENABLE) | PERM(WRITE_COMMAND, ENABLE), PERM(RI, ENABLE) | PERM_VAL(UUID_LEN, 0x02), 0x200},
```

```
63
64 const uint8_t app_rdtss_svc_uuid[16] = ATT_SERVICE_AM_SPEED_128;
65 const struct attm_desc_128 app_rdtss_att_db[RTSS_IDX_NB] =
66 {
67     /* Service Declaration */
68     [0] = {{0x00, 0x28}, PERM(RD, ENABLE), 0, 0},
69
70     /* Characteristic Declaration */
71     [1] = {{0x03, 0x28}, PERM(RD, ENABLE), 0, 0},
72     /* Characteristic Value */
73     [2] = {ATT_CHAR_AM_SPEED_WRITE_128, PERM(WRITE_COMMAND, ENABLE), PERM(RI, ENABLE) | PERM_VAL(UUID_LEN, 0x02), 0x200},
74     /* Client Characteristic Configuration */
75     [3] = {{0x02, 0x29}, PERM(RD, ENABLE) | PERM(WRITE_REQ, ENABLE), PERM(RI, ENABLE), 20},
76
77     /* Characteristic Declaration */
78     [4] = {{0x03, 0x28}, PERM(RD, ENABLE), 0, 0},
79     /* Characteristic Value */
80     [5] = {ATT_CHAR_AM_SPEED_NTF_128, PERM(NTF, ENABLE) | PERM(RD, ENABLE) | PERM(WRITE_COMMAND, ENABLE), PERM(RI, ENABLE) | PERM_VAL(UUID_LEN, 0x02), 0x200},
81     /* Client Characteristic Configuration */
82     [6] = {{0x02, 0x29}, PERM(RD, ENABLE) | PERM(WRITE_REQ, ENABLE), PERM(RI, ENABLE), 20},
83 }
```

特征值添加权限前、后的实现效果如下：



- 16bit UUID 的特征值权限定义和128bit的基本一致，具体参考结构体struct attm_desc_16的定义。

2 添加自定义的特征值

`ble_rdtss_peripheral` 数传例程蓝牙服务的权限定义位于 `app_rdtss.c` 的结构体数组 `struct attm_desc_128 app_rdtss_att_db[]` 里面。用户可以修改该数组来添加新的特征值。

譬如，我们就在这个服务里添加一个 128bit UUID 的特征值，并且配置其有读、写和通知三种权限。

2.1 添加特征值的UUID值

在原来基础上，添加一个用户自定义的特征值，其UUID为：0x0400276008c211e190730e8ac72e1002

```
#define ATT_SERVICE_AM_SPEED_128      {0x01,0x10,0x2E,0xC7,0x8a,0x0E, 0x73,0x90, 0xE1,0x11, 0xC2,0x08, 0x60,0x27,0x00,0x01}  
#define ATT_CHAR_AM_SPEED_WRITE_128  {0x01,0x00,0x2E,0xC7,0x8a,0x0E, 0x73,0x90, 0xE1,0x11, 0xC2,0x08, 0x60,0x27,0x00,0x02}  
#define ATT_CHAR_AM_SPEED_NTF_128    {0x02,0x00,0x2E,0xC7,0x8a,0x0E, 0x73,0x90, 0xE1,0x11, 0xC2,0x08, 0x60,0x27,0x00,0x03}  
#define ATT_CHAR_AM_SPEED_USER1_128  {0x02,0x00,0x2E,0xC7,0x8a,0x0E, 0x73,0x90, 0xE1,0x11, 0xC2,0x08, 0x60,0x27,0x00,0x04}
```

2.2 添加特征值的枚举声明

在 `app_rdtss.h` 中给新添加的特征值在枚举中添加新的枚举声明：

添加特征声明： `RTDSS_IDX_USER_CHAR`

添加特征值： `RTDSS_IDX_USER_VAL`

添加特征配置描述符： `RTDSS_IDX_USER_CFG`

```
/// rdtss Service Attributes Indexes  
enum  
{  
    RTDSS_IDX_SVC,  
  
    RTDSS_IDX_WRITE_CHAR,  
    RTDSS_IDX_WRITE_VAL,  
    RTDSS_IDX_WRITE_CFG,  
  
    RTDSS_IDX_NTF_CHAR,  
    RTDSS_IDX_NTF_VAL,  
    RTDSS_IDX_NTF_CFG,  
  
    RTDSS_IDX_USER1_CHAR, // Characteristic Declaration  
    RTDSS_IDX_USER1_VAL,  // Characteristic Value  
    RTDSS_IDX_USER1_CFG,  // Client Characteristic Configuration  
  
    RTDSS_IDX_NB,  
};
```

说明：

1. 例程中，使用 UUID 为 `ATT_CHAR_AM_SPEED_WRITE_128` 的特征值对应的写标号为

`RTDSS_IDX_WRITE_VAL`，接收数据时判断 `handle` 句柄为此标号时即为手机 App（主机）向（从机）此特征值写入数据，即下发的数据。

- 通过定义 RDTSS_IDX_NTF_VAL 当作 UUID 为 ATT_CHAR_AM_SPEED_NTF_128 的特征值通知标号，通过通知操作上发数据时，在 handle 填入 RDTSS_IDX_NTF_VAL 即可识别为向此特征值上发数据。
- 所以在添加枚举声明的时候和后续添加特征值到 GATT database 时，添加的顺序一定要一一对应，否则就会导致相应功能的标号错位。

2.3 添加特征值到DB

修改数据 app_rdtss_att_db [RDTSS2_IDX_NB]，以添加新的特征值。

```
const uint8_t app_rdtss_svc_uuid[16] = ATT_SERVICE_AM_SPEED_128;
const struct attm_desc_128 app_rdtss_att_db[RDTSS_IDX_NB] =
{
    /* Service Declaration */
    [0] = {{0x00,0x28}, PERM(RD, ENABLE), 0, 0},
    /* Characteristic Declaration */
    [1] = {{0x03,0x28}, PERM(RD, ENABLE), 0, 0},
    /* Characteristic Value */
    [2] = {ATT_CHAR_AM_SPEED_WRITE_128, PERM(WRITE_COMMAND, ENABLE), PERM(RI, ENABLE) | PERM_VAL(UUID_LEN, 0x02), 0x200},
    /* Client Characteristic Configuration */
    [3] = {{0x02,0x29}, PERM(RD, ENABLE) | PERM(WRITE_REQ, ENABLE), PERM(RI, ENABLE), 20},
    /* Characteristic Declaration */
    [4] = {{0x03,0x28}, PERM(RD, ENABLE), 0, 0},
    /* Characteristic Value */
    [5] = {ATT_CHAR_AM_SPEED_NTF_128, PERM(NTF, ENABLE) | PERM(RD, ENABLE) | PERM(WRITE_COMMAND, ENABLE), PERM(RI, ENABLE) | PERM_VAL(UUID_LEN, 0x02), 0x200},
    /* Client Characteristic Configuration */
    [6] = {{0x02,0x29}, PERM(RD, ENABLE) | PERM(WRITE_REQ, ENABLE), PERM(RI, ENABLE), 20},
    /* Characteristic Declaration */
    [7] = {{0x03,0x28}, PERM(RD, ENABLE), 0, 0},
    /* Characteristic Value */
    [8] = {ATT_CHAR_AM_SPEED_USER1_128, PERM(NTF, ENABLE) | PERM(RD, ENABLE) | PERM(WRITE_COMMAND, ENABLE), PERM(RI, ENABLE) | PERM_VAL(UUID_LEN, 0x02), 0x200},
    /* Client Characteristic Configuration */
    [9] = {{0x02,0x29}, PERM(RD, ENABLE) | PERM(WRITE_REQ, ENABLE), PERM(RI, ENABLE), 20},
};
```

2.4 从机接收数据回调函数

标号 RDTSS_IDX_USER1_CFG 为新特征值 ATT_CHAR_AM_SPEED_UAER1_128 的配置描述标号，App 使能特征值的通知权限后，从机程序会进入特征值对应的标号进行处理。

在 app_rdtss.c 的 rdtss_val_write_ind_handler() 函数内，为新特征值添加通知权限处理代码：

```
uint16_t handle = ind_value->handle;
uint16_t length = ind_value->length;

switch (handle)
{
    case RDTSS_IDX_NTF_CFG:
        if(length == 2)
        {
            uint16_t cfg_value = ind_value->value[0] + ind_value->value[1];

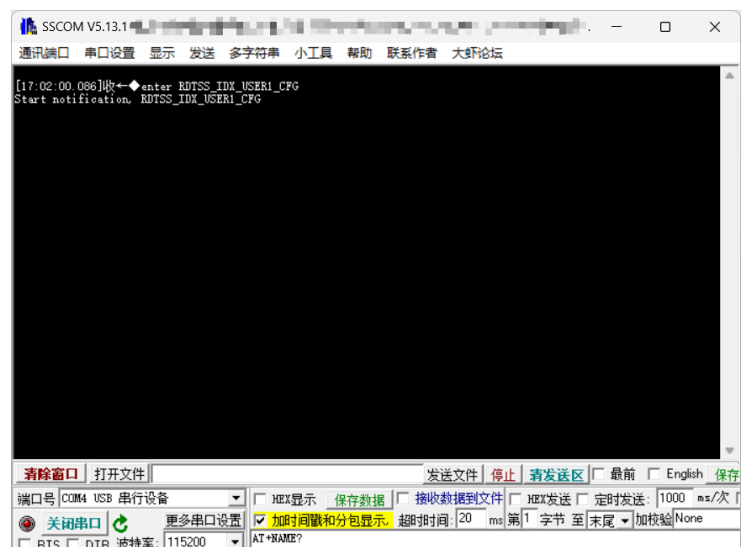
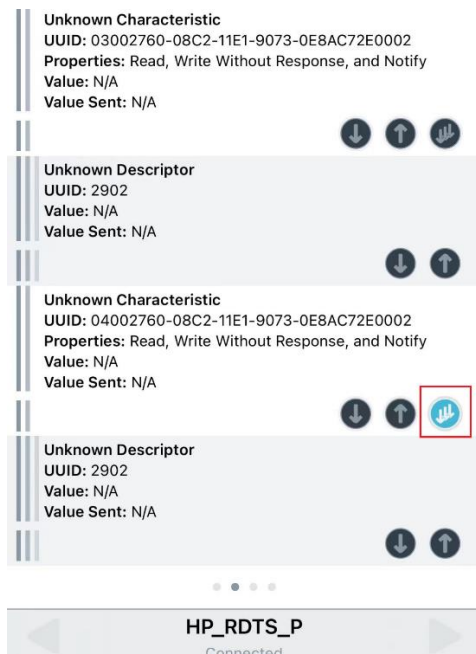
            if(cfg_value == PRF_CLI_START_NTF)
            {
                //enabled notify
            }
            else if(cfg_value == PRF_CLI_STOP_NTFFIND)
            {
            }
        }

        break;

    case RDTSS_IDX_USER1_CFG:
        HP_LOG_INFO("enter RDTSS_IDX_USER1_CFG\r\n");
        if(length == 2)
        {
            uint16_t cfg_value = ind_value->value[0] + ind_value->value[1];
            if(cfg_value == PRF_CLI_START_NTF)
            {
                // Start notification
                HP_LOG_INFO("Start notification, RDTSS_IDX_USER1_CFG\r\n");
            }
            else if(cfg_value == PRF_CLI_STOP_NTFFIND)
            {
                // Stop notification/indication
                HP_LOG_INFO("Stop notification/indication, RDTSS_IDX_USER1_CFG\r\n");
            }
        }
        break;

    case RDTSS_IDX_WRITE_VAL:
        //ble data receive
}
```

当APP使能新特征值的通知权限时，程序进入此特征值对应的特征配置描述符标号判断里面：

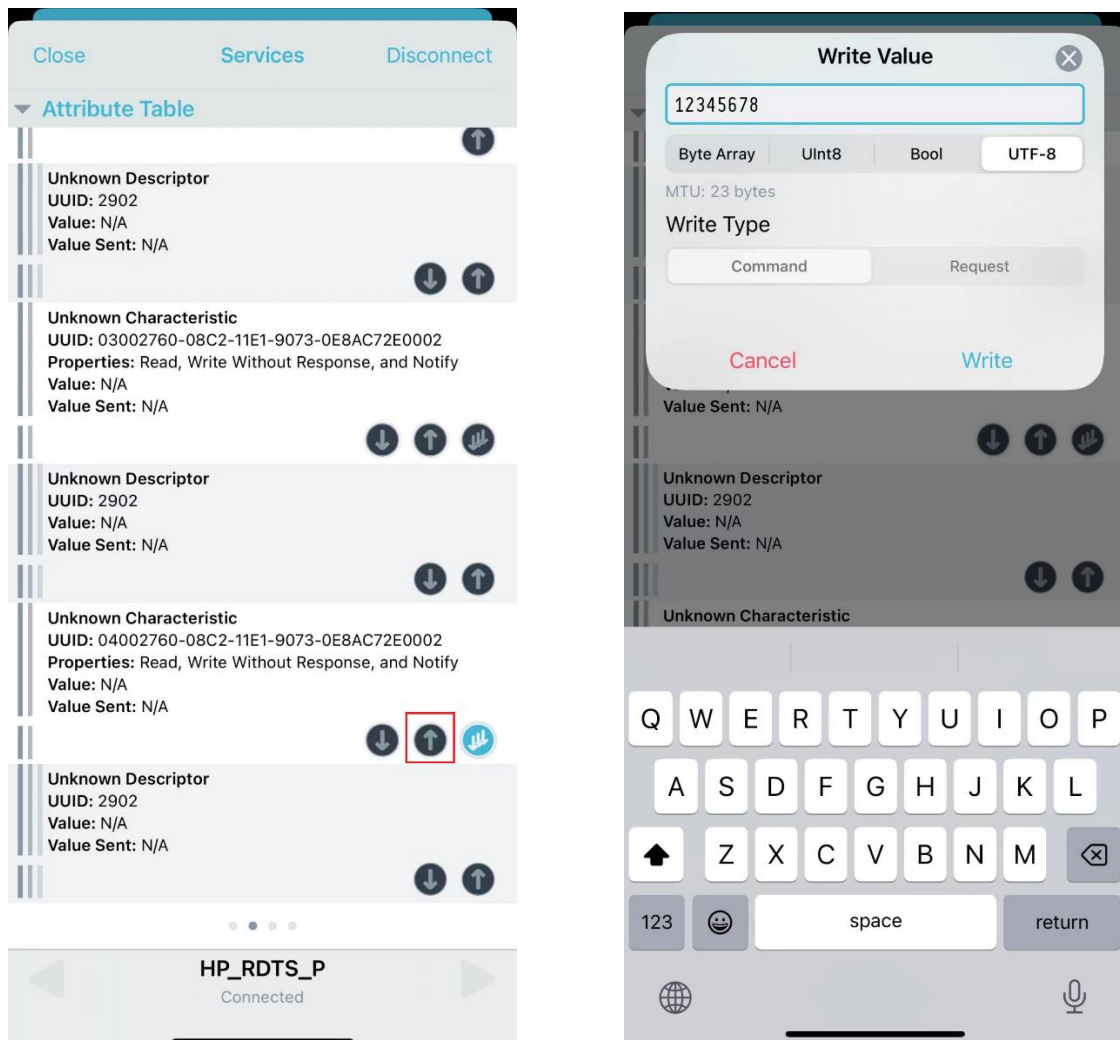


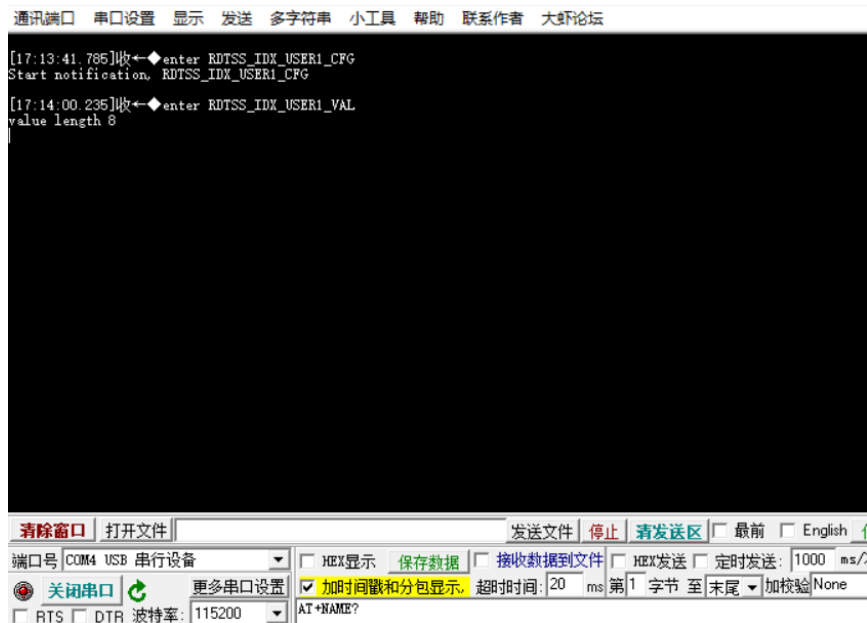
标号RDTSS_IDX_USER1_VAL为新特征值ATT_CHAR_AM_SPEED_UAER1_128的特征值标号，我们将可以通过它执行特征值的读，写和通知操作。

在 `app_rdtss.c` 的 `rdtss_val_write_ind_handler()`函数内，为新特征值添加接收数据处理代码，用户在此函数内获取蓝牙数据：

```
case RDTSS_IDX_WRITE_VAL:  
    //ble data receive  
    app_uart_tx_fifo_enter(ind_value->value, ind_value->length);  
    break;  
case RDTSS_IDX_USER1_VAL:  
    HP_LOG_INFO("enter RDTSS_IDX_USER1_VAL\r\n");  
    HP_LOG_INFO("value length %d \r\n", ind_value->length);  
    break;  
default:  
    break;
```

例如，手机APP通过新特征值下发数据12345678给从机，从机接收数据如下：



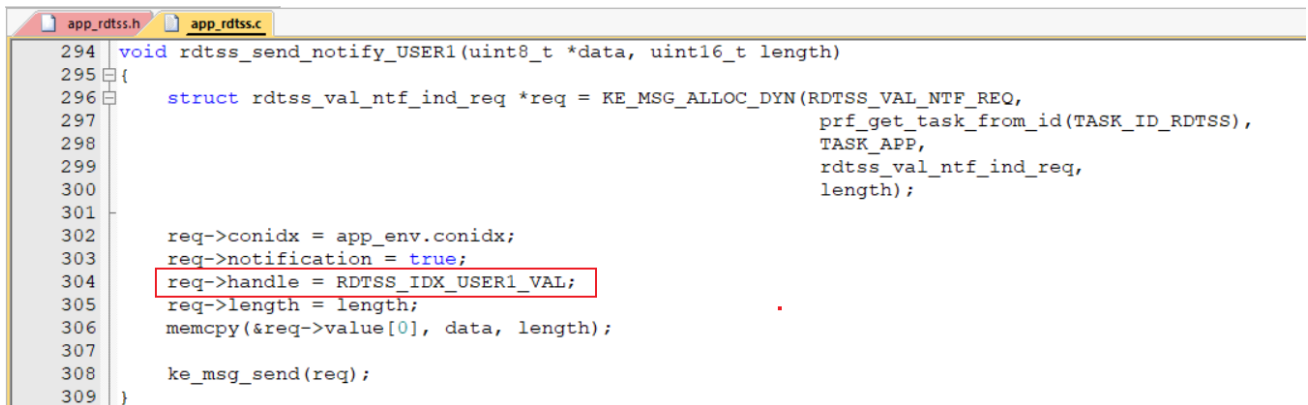


2.5 从机发送数据

标号RDTSS_IDX_USER1_VAL为新特征值ATT_CHAR_AM_SPEED_UAER1_128的标号，我们可以通过它对特征值执行读，写和通知操作。

为新特征值ATT_CHAR_AM_SPEED_UAER1_128添加发送数据的函数：

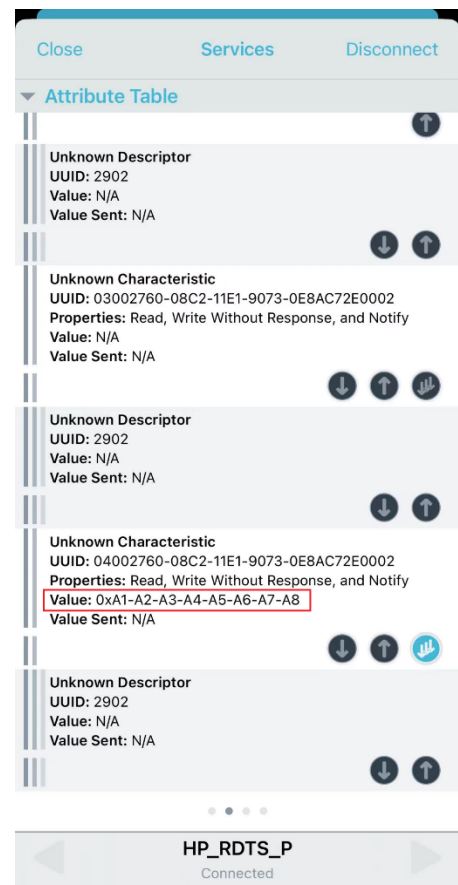
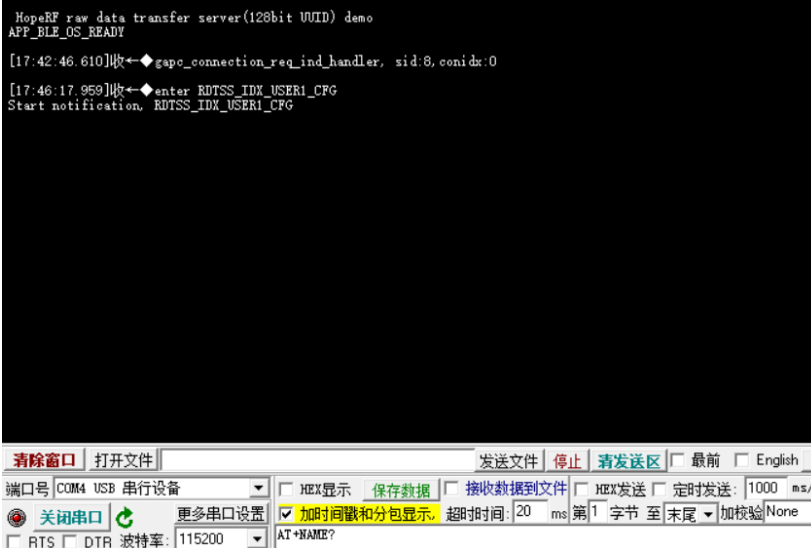
rdtss_send_notify_USER1()



例如：在*rdtss_val_write_ind_handler()*函数中，通过新特征值ATT_CHAR_AM_SPEED_UAER1_128的RDTSS_IDX_USER1_VAL标号给App发数据。在App使能新特征值的通知权限时，将自动通过notification发送数据0xA1 A2 A3 A4 A5 A6 A7 A8给App端。

```
case RDTSS_IDX_USER1_CFG:
    HP_LOG_INFO("enter RDTSS_IDX_USER1_CFG\r\n");
    if(length ==2)
    {
        uint16_t cfg_value = ind_value->value[0] + ind_value->value[1];
        if(cfg_value == PRF_CLI_START_NTF)
        {
            // Start notification
            HP_LOG_INFO("Start notification, RDTSS_IDX_USER1_CFG\r\n");
            uint8_t ntf_val[8] = {0xA1, 0xA2, 0xA3, 0xA4, 0xA5, 0xA6, 0xA7, 0xA8};
            rdtss_send_notify_USER1(ntf_val, sizeof(ntf_val));
        }
        else if(cfg_value == PRF_CLI_STOP_NTFIND)
        {
            // Stop notification/indication
            HP_LOG_INFO("Stop notification/indication, RDTSS_IDX_USER1_CFG\r\n");
        }
    }
    break;
```

通讯端口 串口设置 显示 发送 多字符串 小工具 帮助 联系作者 大虾论坛



2.6 主机(手机)读取从机数据

当主机App读取从机数据时，程序进入读数据请求回调函数，用户可以在此处把需要上传的数据发送给主机。在例程中，我们默认将manufacturer name发送给主机。

```
static int rdtss_value_req_ind_handler(ke_msg_id_t const msgid,
                                      struct rdtss_value_req_ind const *req_value,
                                      ke_task_id_t const dest_id,
                                      ke_task_id_t const src_id)
{
    HP_LOG_DEBUG("%s\r\n", __func__);

    // Initialize length
    uint8_t len = 0;
    // Pointer to the data
    uint8_t *data = NULL;

    len = APP_RDTSS_MANUFACTURER_NAME_LEN;
    data = (uint8_t *)APP_RDTSS_MANUFACTURER_NAME;

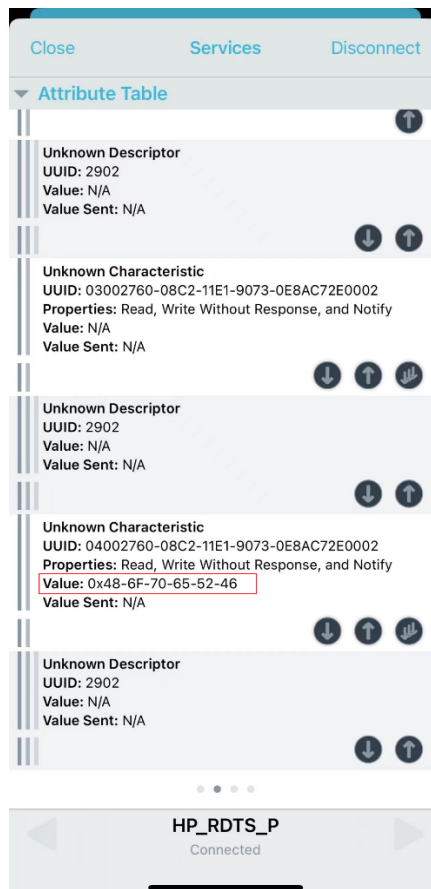
    // Allocate confirmation to send the value
    struct rdtss_value_req_rsp *rsp_value = KE_MSG_ALLOC_DYN(RDTSS_VALUE_REQ_RSP,
                                                             src_id, dest_id,
                                                             rdtss_value_req_rsp,
                                                             len);

    rsp_value->length = len;
    rsp_value->att_idx = req_value->att_idx;
    if (len)
    {
        // Copy data
        memcpy(&rsp_value->value, data, len);
    }
    // Send message
    ke_msg_send(rsp_value);

    return (KE_MSG_CONSUMED);
}
```

```
/// Manufacturer Name Value
#define APP_RDTSS_MANUFACTURER_NAME ("Hoperf")
#define APP_RDTSS_MANUFACTURER_NAME_LEN (6)
```

下图所示，为App读取到的数据：



3 添加自定义服务

上述章节中，我们介绍了如何来添加一个特征值，修改特征值的权限。在本节，我们将讨论如何来添加一个新的服务。为了方便用户理解，我们以`ble_rdtss_peripheral`例程中的`rdtss`服务为模板，添加一个`rdtss2`新服务。

在此之前我们可以通过全局搜索 `BLE_RDTSS_SERVER` 这个宏来了解这一个服务所包含的文件和代码。

1) 在 `app_user_config.h` 定义宏 `CFG_PRF_RDTSS2` 的值为 1 使能这个新的服务。

```
#define CFG_PRF_RDTSS2 1

/* profiles config */
#define CFG_APP_DIS 1
#define CFG_PRF_DIS 1
// #define CFG_APP_BATT 1
// #define CFG_PRF_BASS 1
#define CFG_PRF_RDTSS 1

#define CFG_PRF_RDTSS2 1 // Add a new service

#ifdef BLE_OTA_ENABLE
#define CFG_APP_HP_IUS 1
#endif
```

2) 在 `rwprf_config.h` 文件添加 `profile` 层的宏控制代码，并且修改 `BLE_RDTSS_ENABLE` 宏的值;

```
#if defined(CFG_PRF_RDTSS2)
#define BLE_RDTSS2_SERVER 1
#else
#define BLE_RDTSS2_SERVER 0
#endif // defined(CFG_PRF_RDTSS2)
```

```
#define BLE_RDTSS_ENABLE (BLE_RDTSS_SERVER || BLE_RDTSS_16BIT_SERVER || BLE_RDTSS2_SERVER)
```

3) 在 `rdts_common.c` 参照 `#if (BLE_RDTSS_SERVER)` 使能内容添加头文件;

```
#if (BLE_RDTSS2_SERVER)
#include "rdtss2.h"
#endif // (BLE_RDTSS2_SERVER)
```

4) 在 `rdts_common.c` 参照 `#if (BLE_RDTSS_SERVER)` 使能内容添加两个函数实现。

```
#if (BLE_RDTSS2_SERVER)
uint16_t rdtss2_get_att_handle(uint8_t att_idx)
{
    struct rdtss2_env_tag *rdtss2_env = PRF_ENV_GET(RDTSS2, rdtss2);
    uint16_t handle = ATT_INVALID_HDL;

    if (att_idx < rdtss2_env->max_nb_att)
```



```

{
    handle = rdtss2_env->shd1 + att_idx;
}

return handle;
}

uint8_t rdtss2_get_att_idx(uint16_t handle, uint8_t *att_idx)
{
    struct rdtss2_env_tag *rdtss2_env = PRF_ENV_GET(RDTSS2, rdtss2);
    uint8_t status = PRF_APP_ERROR;

    if ((handle >= rdtss2_env->shd1) && (handle < rdtss2_env->shd1 + rdtss2_env->max_nb_att))
    {
        *att_idx = handle - rdtss2_env->shd1;
        status = ATT_ERR_NO_ERROR;
    }

    return status;
}
#endif // (BLE_RDTSS2_SERVER)

```

5) 在 *rdts_common.h* 参照 `#if (BLE_RDTSS_SERVER)`, 声明两个函数。

```

#if (BLE_RDTSS2_SERVER)
uint16_t rdtss2_get_att_handle(uint8_t att_idx);
uint8_t rdtss2_get_att_idx(uint16_t handle, uint8_t *att_idx);
#endif // (BLE_RDTSS2_SERVER)

```

6) 在 *rwip_task.h* 添加这个服务的任务号, 注意不要与其他任务号重复。这个例程最小的任务号为 76。






```
TASK_ID_RDTSS2 = 76,
```

7) 添加 `profile` 层库文件

在 *middlewares\HopeRF\ble_library\hp_ble_profile\rdts\rdtss* 目录下, 复制 *rdtss.c*, *rdtss.h*, *rdtss_task.c*, *rdtss_task.h* 四个文件并命名为 *rdtss2.c*, *rdtss2.h*, *rdtss2_task.c*, *rdtss2_task.h*。替换新文件中的 RDTSS 为 RDTSS2, 替换 *rdtss* 为 *rdtss2*, 避免重名声明。







rdtss2.c 和 *rdtss2_task.c* 文件路径如下:

middlewares > HopeRF > ble_library > hp_ble_profile > rdtss > rdtss > src

名称	修改日期	类型
 rdtss.c	2023/5/15 17:38	C 源文件
 rdtss_16bit.c	2023/5/15 17:38	C 源文件
 rdtss_16bit_task.c	2023/5/15 17:39	C 源文件
 rdtss_task.c	2023/5/15 17:39	C 源文件
 rdtss2.c	2023/6/1 10:16	C 源文件
 rdtss2_task.c	2023/6/1 10:18	C 源文件

*rdtss2.h*和*rdtss2_task.h*文件路径如下

middlewares > HopeRF > ble_library > hp_ble_profile > rdtss > rdtss > api

名称	修改日期	类型
 rdtss.h	2023/5/15 17:38	H 文件
 rdtss_16bit.h	2023/5/15 17:38	H 文件
 rdtss_16bit_task.h	2023/5/15 17:38	H 文件
 rdtss_task.h	2023/5/15 17:39	H 文件
 rdtss2.h	2023/6/1 10:17	H 文件
 rdtss2_task.h	2023/6/1 10:19	H 文件

8) 添加 profile 应用层代码

在例程的 *app_profile* 目录下复制 *app_rdtss.c* 和 *app_rdtss.h* 为 *app_rdtss2.c* 和 *app_rdtss2.h*，参考上述方法，替换文件中的 RDTSS 为 RDTSS2，替换 rdtss 为 rdtss2。

*app_rdtss2.c*文件路径如下：

projects > cmt453x_EVAL > ble_peripheral > ble_rdtss_peripheral_customService > src > app_profile

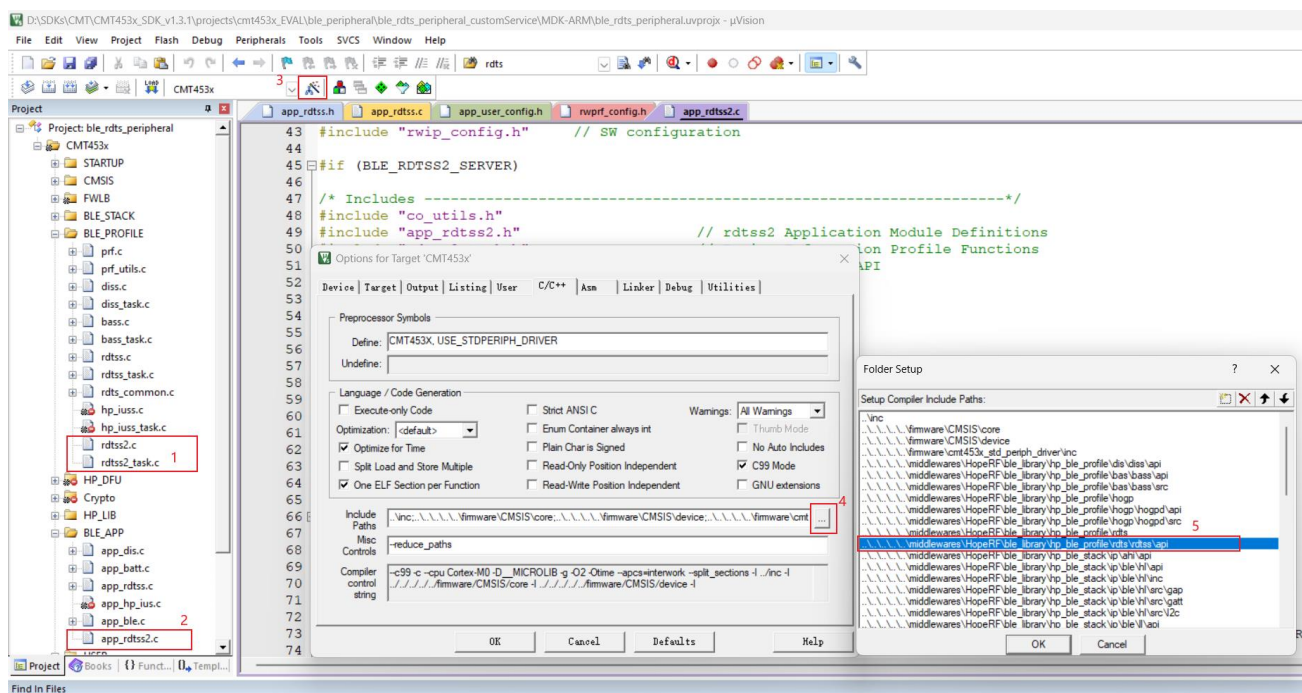
名称	修改日期	类型	大小
 app_batt.c	2023/5/15 17:38	C 源文件	9 KB
 app_dis.c	2023/5/15 17:38	C 源文件	8 KB
 app_hp_ius.c	2023/5/15 17:49	C 源文件	8 KB
 app_rdtss.c	2023/5/31 17:40	C 源文件	13 KB
 app_rdtss2.c	2023/6/1 10:34	C 源文件	13 KB

*app_rdtss2.h*文件路径如下：

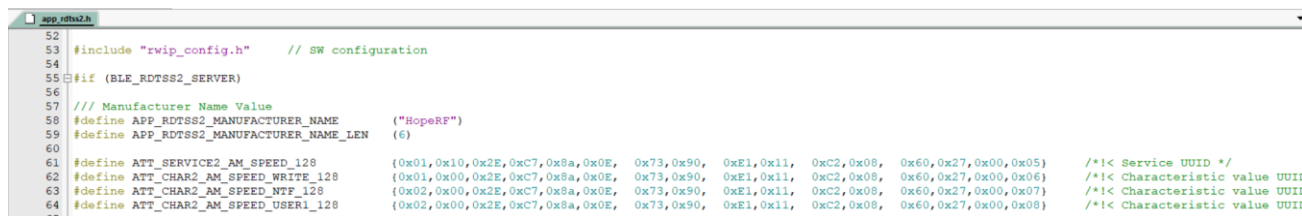
projects > cmt453x_EVAL > ble_peripheral > ble_rdtss_peripheral_customService > inc > app_profile

名称	修改日期	类型	大小
app_batt.h	2023/5/15 17:38	H 文件	4 KB
app_dis.h	2023/5/15 17:38	H 文件	5 KB
app_hp_ius.h	2023/5/15 17:49	H 文件	4 KB
app_rdtss.h	2023/5/31 18:22	H 文件	4 KB
app_rdtss2.h	2023/6/1 10:37	H 文件	4 KB
rwapp_config.h	2023/5/15 17:38	H 文件	4 KB

- 9) 把上述修改的 profile 层库文件和应用层的.c 源文件添加到工程相应的目录，即 BLE_PROFILE 和 BLE_APP 目录，并参照上述头文件路径，把.h 文件添加到工程中（默认已添加）。



- 10) 请参考第 1.1 节的内容修改 UUID，如需更改权限请参考 1.2 节的内容。请注意修改服务和特征值的宏定义和 UUID 值，避免重复定义：



在修改服务和特征值的宏定义及UUID值之后，请注意同步修改app_rdtss2.c中的相关代码。

```

app_rdtss2.c
62  /* Private variables ----- */
63
64  const uint8_t app_rdtss2_svc_uuid[16] = ATT_SERVICE2_AM_SPEED_128;
65  const struct attm_desc_128 app_rdtss2_att_db[RTSS2_IDX_NB] =
66  {
67      /* Service Declaration */
68      [0] = {{0x00, 0x28},          PERM(RD, ENABLE),          0,          0},
69
70      /* Characteristic Declaration */
71      [1] = {{0x03, 0x28},          PERM(RD, ENABLE),          0,          0},
72      /* Characteristic Value */
73      [2] = {ATT_CHAR2_AM_SPEED_WRITE_128, PERM(WRITE_COMMAND, ENABLE), PERM_VAL(UUID_LEN, 0x02), 0x200},
74      /* Client Characteristic Configuration */
75      [3] = {{0x02, 0x29},          PERM(RD, ENABLE) | PERM(WRITE_REQ, ENABLE), PERM(RI, ENABLE), 20},
76
77      /* Characteristic Declaration */
78      [4] = {{0x03, 0x28},          PERM(RD, ENABLE),          0,          0},
79      /* Characteristic Value */
80      [5] = {ATT_CHAR2_AM_SPEED_NTF_128, PERM(NTF, ENABLE) | PERM(RD, ENABLE) | PERM(WRITE_COMMAND, ENABLE), PERM(RI, ENABLE) | PERM_VAL(UUID_LEN, 0x02), 0x200},
81      /* Client Characteristic Configuration */
82      [6] = {{0x02, 0x29},          PERM(RD, ENABLE) | PERM(WRITE_REQ, ENABLE), PERM(RI, ENABLE), 20},
83
84      /* Characteristic Declaration */
85      [7] = {{0x03, 0x28},          PERM(RD, ENABLE),          0,          0},
86      /* Characteristic Value */
87      [8] = {ATT_CHAR2_AM_SPEED_USER1_128, PERM(NTF, ENABLE) | PERM(RD, ENABLE) | PERM(WRITE_COMMAND, ENABLE), PERM(RI, ENABLE) | PERM_VAL(UUID_LEN, 0x02), 0x200},
88      /* Client Characteristic Configuration */
89      [9] = {{0x02, 0x29},          PERM(RD, ENABLE) | PERM(WRITE_REQ, ENABLE), PERM(RI, ENABLE), 20},
90
91  };

```

- 11) 先在 `app_ble.c` 文件中添加头文件 `#include "app_rdtss2.h"`，然后在 `app_ble_prf_init()` 函数里调用 `hp_ble_add_prf_func_register(app_rdtss2_add_rdtss)` 注册新增服务。

```

void app_ble_prf_init(void)
{
    #if (BLE_APP_DIS)
    //add device informaiton server
    hp_ble_add_prf_func_register(app_dis_add_dis);
    #endif //BLE_APP_DIS
    #if (BLE_APP_BATT)
    //add battery level server
    hp_ble_add_prf_func_register(app_batt_add_bas);
    #endif //BLE_APP_BATT
    #if (BLE_APP_HP_IUS)
    hp_ble_add_prf_func_register(app_hp_ius_add_hp_ius);
    #endif //BLE_APP_HP_IUS

    //add raw data transmit server(rdtss)
    hp_ble_add_prf_func_register(app_rdtss_add_rdtss);

    hp_ble_add_prf_func_register(app_rdtss2_add_rdtss);
}

```

- 12) 至此服务已经添加完成，后续可以在 `rdtss2_val_write_ind_handler()` 的 `RTSS2_IDX_USER1_CFG` 事件回调获取主机配置从机的状态，通过 `RTSS2_IDX_WRITE_VAL` 事件回调获取从机接收到的数据，通过 `rdtss2_send_notify()` 函数向主机发送数据。

例如，在主机使能新特征值的通知权限时，从机自动向主机发送 `0xC1 C2 C3 C4 C5 C6 C7 C8` 字符串。

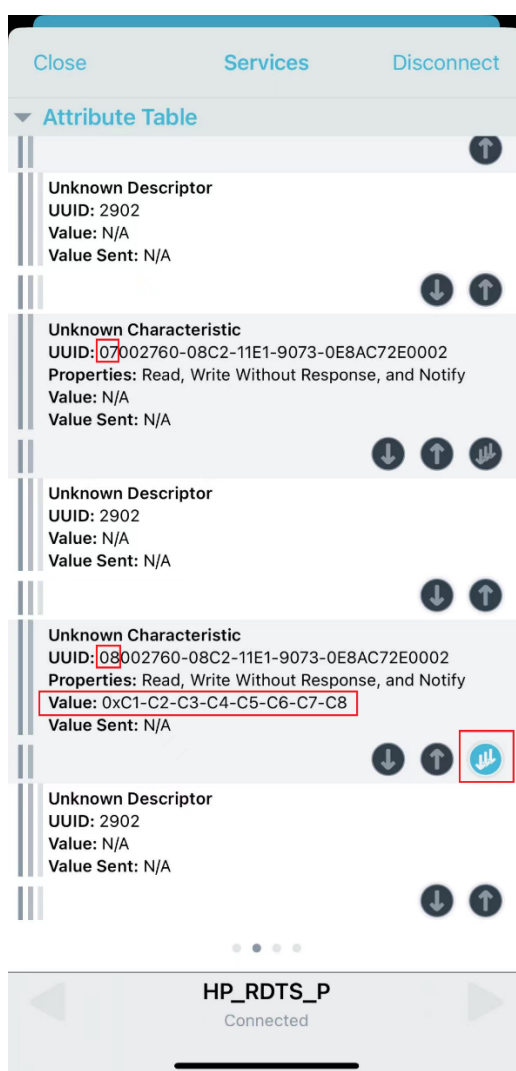
```

case RTSS2_IDX_USER1_CFG:
    HP_LOG_INFO("enter RTSS2_IDX_USER1_CFG\r\n");
    if(length == 2)
    {
        uint16_t cfg_value = ind_value->value[0] + ind_value->value[1];
        if(cfg_value == PRF_CLI_START_NTF)
        {
            // Start notification
            HP_LOG_INFO("Start notification, RTSS2_IDX_USER1_CFG\r\n");
            uint8_t ntf_val[8] = {0xC1, 0xC2, 0xC3, 0xC4, 0xC5, 0xC6, 0xC7, 0xC8};
            rdtss2_send_notify_USER1(ntf_val, sizeof(ntf_val));
        }
    }

```

```
}  
else if(cfg_value == PRF_CLI_STOP_NTFFIND)  
{  
    // Stop notification/indication  
    HP_LOG_INFO("Stop notification/indication, RDTSS2_IDX_USER1_CFG\r\n");  
}  
}  
break;
```

如下图所示，手机连接之后，可以发现新添加的服务及特征值，使能新特征值的通知权限，将收到相应的数据。



4 历史版本

版本	日期	备注
V1.0	2023.05.26	新建文档