
N32WB031 服务和特征值的配置操作说明 V1.0

简介

本文档介绍 N32WB031 系列 32 位 蓝牙芯片（以下简称 N32WB031）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。本文档目的在于让使用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，以减少开发前期的准备时间，降低开发难度。

NATIONS CONFIDENTIAL

目录

1 修改服务和特征值的 UUID 和权限	1
1.1 修改服务和特征值的 UUID 值.....	1
1.2 修改特征值的权限	3
2 添加自定义的特征值	5
2.1 添加特征值的 UUID 值.....	5
2.2 添加特征值的枚举声明	5
2.3 添加特征值到 DB.....	6
2.4 从机接收数据回调函数	6
2.5 从机发送数据	9
2.6 主机(手机)读取从机数据.....	10
3 添加自定义服务	11
4 历史版本	21
5 声明	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 RDTSS 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：ATT_CHAR_AM_SPEED_WRITE_128

服务的通知特征值的 UUID 宏定义：ATT_CHAR_AM_SPEED_NTF_128

PRELIMINARY

```

37 #ifndef APP_RDTSS_H
38 #define APP_RDTSS_H
39
40 /**
41  * @addtogroup APP
42  * @brief Battery Application Module entry point
43  * @{}
44  * @{}
45  */
46
47 /* Includes -----*/
48
49 #include "rwip_config.h" // SW configuration
50
51 #if (BLE_RDTSS_SERVER)
52
53 // Manufacturer Name Value
54 #define APP_RDTSS_MANUFACTURER_NAME ("Nations")
55 #define APP_RDTSS_MANUFACTURER_NAME_LEN (7)
56
57 #define ATT_SERVICE_AM_SPEED_128 {0x01,0x10,0x2E,0xC7,0x8A,0x0E, 0x73,0x90, 0xE1,0x11, 0xC2,0x08, 0x60,0x27,0x00,0x00} /*!< Service UUID */
58 #define ATT_CHAR_AM_SPEED_WRITE_128 {0x01,0x00,0x2E,0xC7,0x8A,0x0E, 0x73,0x90, 0xE1,0x11, 0xC2,0x08, 0x60,0x27,0x00,0x00} /*!< Characteristic value UUID */
59 #define ATT_CHAR_AM_SPEED_NTF_128 {0x02,0x00,0x2E,0xC7,0x8A,0x0E, 0x73,0x90, 0xE1,0x11, 0xC2,0x08, 0x60,0x27,0x00,0x00} /*!< Characteristic value UUID */
60
61
62
63
64
65
66
67
68
69
70

```

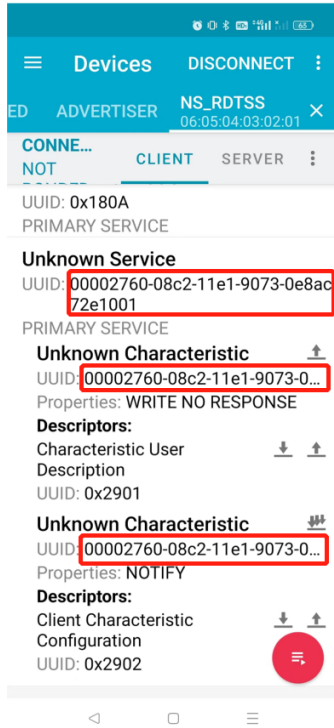
修改服务和特征值的 UUID 值：修改宏定义数组内的成员数值即可修改 UUID 值

```

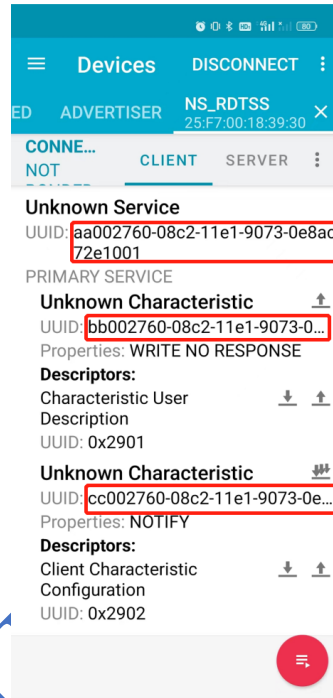
42 * @ingroup RICOV
43 *
44 * @brief Battery Application Module entry point
45 *
46 * @{}
47 **/
48
49 /* Includes -----*/
50
51 #include "rwip_config.h" // SW configuration
52
53 #if (BLE_RDTSS_SERVER)
54
55 // Manufacturer Name Value
56 #define APP_RDTSS_MANUFACTURER_NAME ("Nations")
57 #define APP_RDTSS_MANUFACTURER_NAME_LEN (7)
58
59 #define ATT_SERVICE_AM_SPEED_128 {0x01,0x10,0x2E,0xC7,0x8A,0x0E, 0x73,0x90, 0xE1,0x11, 0xC2,0x08, 0x60,0x27,0x00,0xaa} /*!< Service UUID */
60 #define ATT_CHAR_AM_SPEED_WRITE_128 {0x01,0x00,0x2E,0xC7,0x8A,0x0E, 0x73,0x90, 0xE1,0x11, 0xC2,0x08, 0x60,0x27,0x00,0xbb} /*!< Characteristic value UUID */
61 #define ATT_CHAR_AM_SPEED_NTF_128 {0x02,0x00,0x2E,0xC7,0x8A,0x0E, 0x73,0x90, 0xE1,0x11, 0xC2,0x08, 0x60,0x27,0x00,0xcc} /*!< Characteristic value UUID */
62
63
64
65
66
67
68
69
70

```

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, 比如服务手机APP上UUID一般显示为: 0x0000276008c211e190730e8ac72e1001。用户如果想改变服务或者特征值的UUID的值, 修改如上宏定义的值就可以进行UUID的变更。

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rdtss例程为说明对象，数传例程蓝牙服务和特征值的的权限，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

```

78 };
79
80
81 /// Internal 128bits UUID service description
82 struct attm_desc_128
83 {
84     /// 128 bits UUID LSB First
85     uint8_t uuid[ATT_UUID_128_LEN];
86     /// Attribute Permissions (@see enum attm_perm_mask)
87     uint16_t perm;
88     /// Attribute Extended Permissions (@see enum attm_value_perm_mask)
89     uint16_t ext_perm;
90     /// Attribute Max Size //属性最大长度
91     /// note: for characteristic declaration contains handle offset 注意：对于特性声明包含手柄偏移
92     /// note: for included service, contains target service handle 注：对于包含的服务，包含目标服务手柄
93     uint16_t max_size;
94 };
95
96

```

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。

特征值权限配置结构体：app_rdtss_att_db[]

```

52 #include "ke_timer.h"
53 #include "stdio.h"
54 #include "rdtss.h"
55 #include "app_usart.h"
56
57 /* Private typedef -----*/
58 /* Private define -----*/
59 /* Private constants -----*/
60 /* Private variables -----*/
61 const uint8_t app_rdtss_svc_uuid[16] = ATT_SERVICE_AM_SPEED_128;
62 const struct attm_desc_128 app_rdtss_att_db[RDRTSS_IDX_NB] =
63 {
64     /* Service Declaration */
65     [0] = { {0x00, 0x28}, PERM(RD, ENABLE), 0, 0 },
66
67     /* Characteristic Declaration */
68     [1] = { {0x03, 0x28}, PERM(RD, ENABLE) | PERM(WRITE_REQ, ENABLE), 0, 0 },
69     /* Characteristic Value */
70     [2] = { ATT_CHAR_AM_SPEED_WRITE_128, PERM(WRITE_COMMAND, ENABLE), PERM(RI, ENABLE) | PERM_VAL(UUID_LEN, 0x02), 0x200 },
71     /* Client Characteristic Configuration descriptor */
72     [3] = { {0x01, 0x29}, PERM(RD, ENABLE) | PERM(WRITE_REQ, ENABLE), PERM(RI, ENABLE), 20 },
73
74     /* Characteristic Declaration */
75     [4] = { {0x03, 0x28}, PERM(RD, ENABLE) | PERM(WRITE_REQ, ENABLE), 0, 0 },
76     /* Characteristic Value */
77     [5] = { ATT_CHAR_AM_SPEED_NTF_128, PERM(NTF, ENABLE), PERM(RI, ENABLE) | PERM_VAL(UUID_LEN, 0x02), 0x200 },
78     /* Client Characteristic Configuration descriptor */
79     [6] = { {0x02, 0x29}, PERM(RD, ENABLE) | PERM(WRITE_REQ, ENABLE), PERM(RI, ENABLE), 20 },
80 };
81
82
83
84

```

例程里通过如下代码给 UUID 为 ATT_CHAR_AM_SPEED_WRITE_128 的特征值定义了写权限，和这个特征值是 128bit 的长度。

```
[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，和这个特征值是128bit 的。

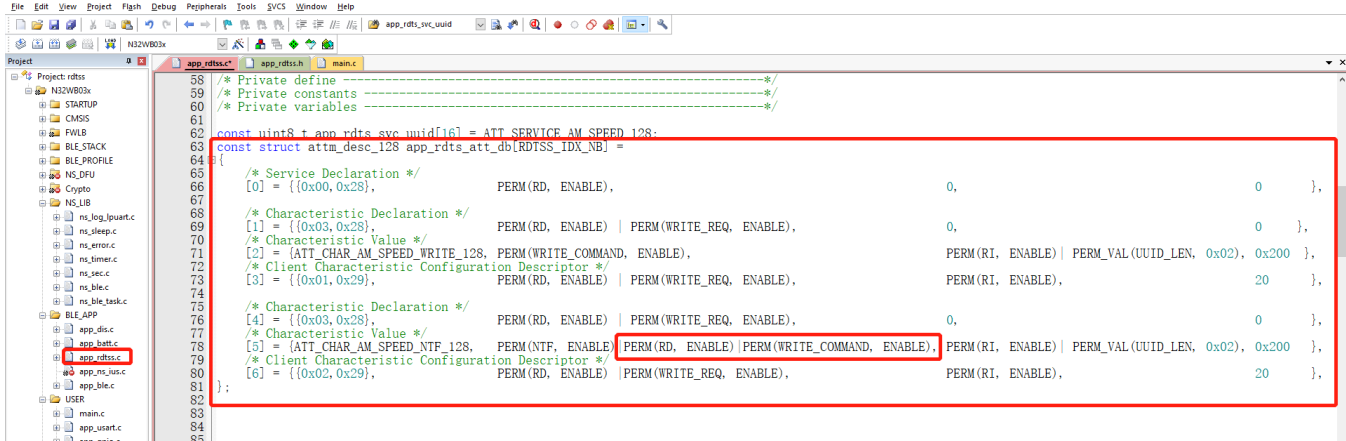
```
[5] = {ATT_CHAR_AM_SPEED_NTF_128, PERM(NTF, ENABLE), PERM(RI, ENABLE) | PERM_VAL(UUID_LEN, 0x02), 0x200 },
```

- 我们可以使用的权限有：RD（读），WRITE_COMMAND（不带回应写），WRITE_REQ（带回应

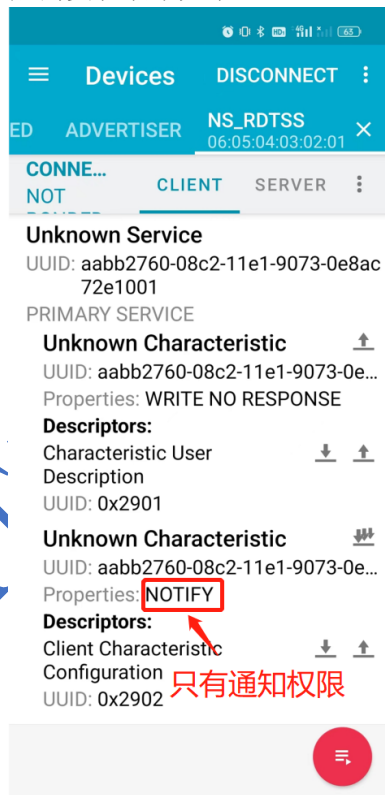
写), NTF (通知), IND (带回应通知), 添加方式为: 在特征值的属性上添加调用函数: PERM(RD, ENABLE), RD表示的是读权限。

- 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:

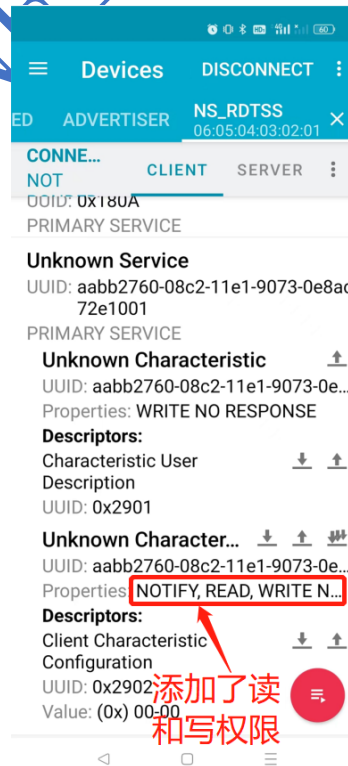
```
[5] = {ATT_CHAR_AM_SPEED_NTF_128, PERM(NTF, ENABLE) | PERM(RD, ENABLE) | PERM(WRITE_COMMAND, ENABLE), PERM(RI, ENABLE) | PERM_VAL(UUID_LEN, 0x02), 0x200},
```



特征值添加权限后的实现效果如下:



修改前



修改后

- 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

RTSS 数传例程蓝牙服务的权限定义位于 app_rdtss.c 的结构体 struct attm_desc_128 rdtss_att_db 里面。比如我们就在这个服务里添加一个 128bit UUID 为 0xFFFF1 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

在原来的基础上，添加一个新的用户自定义的特征值的 UUID：0xdd00276008c211e190730e8ac72e1001。

```

50
51 #include "rwip_config.h" // SW configuration
52
53 #if (BLE_RDTSS_SERVER)
54
55 // Manufacturer Name Value
56 #define APP_RDTSS_MANUFACTURER_NAME ("Nations")
57 #define APP_RDTSS_MANUFACTURER_NAME_LEN (7)
58
59
60
61 #define ATT_SERVICE_AM_SPEED_128 {0x01, 0x10, 0x2E, 0xc7, 0x8a, 0x0E, 0x73, 0x90, 0xE1, 0x11, 0xC2, 0x08, 0x60, 0x27, 0x00, 0xaa} /*< Service UUID */
62 #define ATT_CHAR_AM_SPEED_WRITE_128 {0x01, 0x00, 0x2E, 0xc7, 0x8a, 0x0E, 0x73, 0x90, 0xE1, 0x11, 0xC2, 0x08, 0x60, 0x27, 0x00, 0xbb} /*< Characteristic value UUID */
63
64 #define ATT_CHAR_AM_SPEED_NTF_128 {0x02, 0x00, 0x2E, 0xc7, 0x8a, 0x0E, 0x73, 0x90, 0xE1, 0x11, 0xC2, 0x08, 0x60, 0x27, 0x00, 0xcc} /*< Characteristic value UUID */
65
66 #define ATT_CHAR_AM_SPEED_UAER1_128 {0x03, 0x00, 0x2E, 0xc7, 0x8a, 0x0E, 0x73, 0x90, 0xE1, 0x11, 0xC2, 0x08, 0x60, 0x27, 0x00, 0xdd} /*< 添加一个用户特征值 */
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94

```

2.2 添加特征值的枚举声明

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：RDTSS_IDX_USER_CHAR

添加特征值：RDTSS_IDX_USER_VAL

添加特征配置描述符：RDTSS_IDX_USER_CFG

```

74 // rdtss Service Attributes Indexes
75 enum
76 {
77     RDTSS_IDX_SVC,
78
79     RDTSS_IDX_WRITE_CHAR,
80     RDTSS_IDX_WRITE_VAL,
81     RDTSS_IDX_WRITE_CFG,
82
83     RDTSS_IDX_NTF_CHAR,
84     RDTSS_IDX_NTF_VAL,
85     RDTSS_IDX_NTF_CFG,
86
87     RDTSS_IDX_USER1_CHAR, //添加特征声明
88     RDTSS_IDX_USER1_VAL, //添加特征值
89     RDTSS_IDX_USER1_CFG, //添加特征配置描述符
90
91     RDTSS_IDX_NB,
92 };
93
94

```

说明:

- 1、例程中使用 RDTSS_IDX_WRITE_VAL 当作 UUID 为 ATT_CHAR_AM_SPEED_WRITE_128 的特征值写标号，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。
- 2、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，通过通知操作上传数据时，在 handle 填入 RDTSS_IDX_NTF_VAL 即可识别为向此特征值上传数据。
- 3、所以在添加枚举声明的时候和后续添加特征值到 DB 使，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

app_rdtss_att_db2[RDTSS2_IDX_NB]

```

61
62 const uint8_t app_rdtss_svc_uuid[16] = ATT_SERVICE_AM_SPEED_128;
63 const struct attm_desc_128 app_rdtss_att_db[RDTSS_IDX_NB] =
64 {
65     /* Service Declaration */
66     [0] = {{0x00, 0x28}, PERM(RD, ENABLE), 0, 0 },
67     /* Characteristic Declaration */
68     [1] = {{0x03, 0x28}, PERM(RD, ENABLE) | PERM(WRITE_REQ, ENABLE), 0, 0 },
69     /* Characteristic Value */
70     [2] = {ATT_CHAR_AM_SPEED_WRITE_128, PERM(WRITE_COMMAND, ENABLE), PERM(RI, ENABLE) | PERM_VAL(UUID_LEN, 0x02), 0x200 },
71     /* Client Characteristic Configuration Descriptor */
72     [3] = {{0x01, 0x29}, PERM(RD, ENABLE) | PERM(WRITE_REQ, ENABLE), PERM(RI, ENABLE), 20 },
73     /* Characteristic Declaration */
74     [4] = {{0x03, 0x28}, PERM(RD, ENABLE) | PERM(WRITE_REQ, ENABLE), 0, 0 },
75     /* Characteristic Value */
76     [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 },
77     /* Client Characteristic Configuration Descriptor */
78     [6] = {{0x02, 0x29}, PERM(RD, ENABLE) | PERM(WRITE_REQ, ENABLE), PERM(RI, ENABLE), 20 },
79     /* USER1 Characteristic Declaration */
80     [7] = {{0x03, 0x28}, PERM(RD, ENABLE) | PERM(WRITE_REQ, ENABLE), 0, 0 },
81     /* USER1 Characteristic Value */
82     [8] = {ATT_CHAR_AM_SPEED_UAER1_128, PERM(NTF, ENABLE) | PERM(RD, ENABLE) | PERM(WRITE_COMMAND, ENABLE), PERM(RI, ENABLE) | PERM_VAL(UUID_LEN, 0x02), 0x200 },
83     /* USER1 Client Characteristic Configuration Descriptor */
84     [9] = {{0x02, 0x29}, PERM(RD, ENABLE) | PERM(WRITE_REQ, ENABLE), PERM(RI, ENABLE), 20 },
85 }

```

修改序号 添加特征值UUID 配置特征值权限

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：


```

196 rdtss_send_notify_USER1((uint8_t *)ind_value->value, ind_value->length); //通过ATT_CHAR_AM_SPEED_UAER1_128特征值的RDTSS_IDX_USER1_VAL
197
198 uint16_t handle = ind_value->handle;
199 uint16_t length = ind_value->length;
200 switch (handle)
201 {
202     case RDTSS_IDX_NTF_CFG: //APP打开ATT_CHAR_AM_SPEED_NTF_128特征值的通知权限, 从机通过此标号接收
203         NS_LOG_INFO("enter RDTSS_IDX_NTF_CFG\r\n");
204         if(length == 2)
205         {
206             uint16_t cfg_value = ind_value->value[0] + ind_value->value[1];
207
208             if(cfg_value == PRF_CLI_START_NTF)//APP打开了通知权限
209             {
210                 //enabled notify
211             }
212             else if(cfg_value == PRF_CLI_STOP_NTFIND)//APP关闭了通知权限
213             {
214             }
215         }
216         break;
217     case RDTSS_IDX_USER1_CFG: //APP打开ATT_CHAR_AM_SPEED_UAER1_128新特征值的通知权限, 从机通过此标号接收
218         NS_LOG_INFO("enter RDTSS_IDX_USER1_CFG\r\n");
219         if(length == 2)
220         {
221             uint16_t cfg_value = ind_value->value[0] + ind_value->value[1];
222
223             if(cfg_value == PRF_CLI_START_NTF)//APP打开了通知权限
224             {
225                 //enabled notify
226             }
227             else if(cfg_value == PRF_CLI_STOP_NTFIND)//APP关闭了通知权限
228             {
229             }
230         }
231         break;
232     case RDTSS_IDX_WRITE_VAL: //APP通过ATT_CHAR_AM_SPEED_WRITE_128的写权限下发数据, 从机通过此标号接收
233         NS_LOG_INFO("enter RDTSS_IDX_WRITE_VAL\r\n");
234

```

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：

1. 在APP中打开新特征值的通知权限，APP即可以接收从机通过此特征值上发的数据

2. APP打开新特征值的通知权限，会进入新特征值的RDTSS_IDX_USER1_CFG标号进行相应的操作

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：

```

226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
    else if(cfg_value == PRF_CLI_STOP_NTFIND)//APP关闭了通知权限
    {
    }
    break;
}
case RDTSS_IDX_WRITE_VAL: //APP通过ATT_CHAR_AM_SPEED_WRITE_128的写权限下发数据, 从机通过此标号接收
NS_LOG_INFO("enter RDTSS_IDX_WRITE_VAL\r\n");
//ble data receive
app_uart_tx_fifo_enter(ind_value->value, ind_value->length); //从机把接收到的APP下发的数据, 通过USART1发送给主控MCU
break;
case RDTSS_IDX_NTF_VAL: //APP通过ATT_CHAR_AM_SPEED_NTF_128的写权限下发数据, 从机通过此标号接收
NS_LOG_INFO("enter RDTSS_IDX_NTF_VAL\r\n");
//ble data receive
app_uart_tx_fifo_enter(ind_value->value, ind_value->length); //从机把接收到的APP下发的数据, 通过USART1发送给主控MCU
break;
case RDTSS_IDX_USER1_VAL: //APP通过ATT_CHAR_AM_SPEED_UAER1_128新特征值的写权限下发数据, 从机通过此标号接收
NS_LOG_INFO("enter RDTSS_IDX_USER1_VAL\r\n");
//ble data receive
app_uart_tx_fifo_enter(ind_value->value, ind_value->length); //从机把接收到的APP下发的数据, 通过USART1发送给主控MCU
break;
default:
break;
}
return (KE_MSG_CONSUMED);
}
    
```

用户可以在RDTSS_IDX_USER1_VAL标号下接收
获取APP通过新特征值下发的数

例如APP通过新特征值下发数据888888给从机，从机接收数据如下：

2. 从机接收到数据之后, 可以进入新特征值的 RDTSS_IDX_USER1_VAL标号进行蓝牙数据的接收

1. APP通过新特征值的写权限给从机发送数值: 888888

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的新特征值标号，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：rdtss2_send_notify_USER1()

```

250 */
251 void rdtss_send_notify(uint8_t *data, uint16_t length)
252 {
253     struct rdtss_val_ntf_ind_req *req = KE_MSG_ALLOC_DYN(RDTSS_VAL_NTF_REQ,
254                                                         prf_get_task_from_id(TASK_ID_RDTSS),
255                                                         TASK_APP,
256                                                         rdtss_val_ntf_ind_req,
257                                                         length);
258
259     req->conid = app_env.conid;
260     req->notification = true;
261     req->handle = RDTSS_IDX_NTF_VAL;
262     req->length = length;
263     memcpy(&req->value[0], data, length);
264
265     ke_msg_send(req);
266 }
267
268 void rdtss_send_notify_USER1(uint8_t *data, uint16_t length) //给新添加的特征值添加一个发送蓝牙数据的函数
269 {
270     struct rdtss_val_ntf_ind_req *req = KE_MSG_ALLOC_DYN(RDTSS_VAL_NTF_REQ,
271                                                         prf_get_task_from_id(TASK_ID_RDTSS),
272                                                         TASK_APP,
273                                                         rdtss_val_ntf_ind_req,
274                                                         length);
275
276     req->conid = app_env.conid;
277     req->notification = true;
278     req->handle = RDTSS_IDX_USER1_VAL; //修改为新特征值对应的上发数据的标号
279     req->length = length;
280     memcpy(&req->value[0], data, length);
281
282     ke_msg_send(req);
283 }
    
```

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：

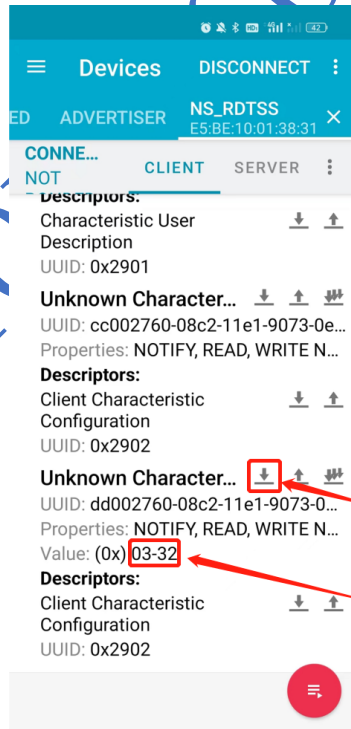
2.6 主机(手机)读取从机数据

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。

```

141 uint8_t ADC_data[] = {0x03, 0x32};
142
143 static int rdtss_value_req_ind_handler(ke_msg_id_t const msgid, //主机APP读取数据回调函数，应用：可以通过此函数上传ADC、温湿度等值
144                                     struct rdtss_value_req_ind const *req_value,
145                                     ke_task_id_t const dest_id,
146                                     ke_task_id_t const src_id)
147 {
148     NS_LOG_DEBUG("%s\r\n", __func__);
149
150     // Initialize length
151     uint8_t len = 0;
152     // Pointer to the data
153     uint8_t *data = NULL;
154
155     // len = APP_RDTSS_MANUFACTURER_NAME_LEN;
156     data = (uint8_t *)APP_RDTSS_MANUFACTURER_NAME;
157
158     len = sizeof(ADC_data);
159     data = (uint8_t *)ADC_data;
160
161     NS_LOG_INFO("\r\n");
162     for(uint8_t i=0; i<len; i++)
163     {
164         NS_LOG_INFO("0x%x ", ADC_data[i]);
165     }
166     NS_LOG_INFO("\r\n");
167     // Allocate confirmation to send the value
168     struct rdtss_value_req_rsp *rsp_value = KE_MSG_ALLOC_DYN(RDTSS_VALUE_REQ_RSP,
169                                                             src_id, dest_id,
170                                                             rdtss_value_req_rsp,
171                                                             len);
172
173     rsp_value->length = len;
174     rsp_value->att_idx = req_value->att_idx;
175     if (len)
176     {
177         // Copy data
178         memcpy(&rsp_value->value, data, len);
179     }
180 }
    
```

如主机读取从机数据：



1、APP 点击读取从机数据按钮

2、从机响应主机读取数据请求，并上传数据

3 添加自定义服务

以上面已经添加了 1 个特征值和修改了特征值权限的 rdtss 服务为模板(如果不需要修改特征值和权限, 可以找一个原始的 rdtss 工程做模板), 添加一个命名为 rdtss2 的新服务。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

```

81 #define SBC_PARAM_KEY_SIZE 16 /**< Minimum encryption key size. 7 to 16 */
82 #define SBC_PARAM_BOND 1 /**< Perform bonding. */
83 #define SBC_PARAM_MITM 1 /**< Man In The Middle protection not required. */
84 #define SBC_PARAM_LESC 0 /**< LE Secure Connections not enabled. */
85 #define SBC_PARAM_KEYPRESS 0 /**< Keypress notifications not enabled. */
86 #define SBC_PARAM_IKEY GAP_RDIST_NONE /**< Initiator Key Distribution. (@enum gap_kdist) */
87 #define SBC_PARAM_IKEY GAP_RDIST_ENCKEY /**< Responder Key Distribution. (@enum gap_kdist) */
88 #define SBC_PARAM_SEC_MODE_LEVEL GAP_NO_SEC /**< Device security requirements (minimum security level). (@enum see gap_sec_req) */
89
90 //bond config
91 #define MAX_BOND_PEER 5
92 #define BOND_STORE_ENABLE 0
93 #define BOND_DATA_BASE_ADDR 0x01020000
94
95 /* profiles config */
96 #define CFG_APP_DIS 1
97 #define CFG_PRF_DIS 1
98 // #define CFG_APP_BATT 1
99 // #define CFG_PRF_BASS 1
100 #define CFG_PRF_RDTSS 1
101
102 #define CFG_PRF_RDTSS2 1 //添加一个新服务
103
104 #ifndef BLE_OTA_ENABLE
105 #define CFG_APP_NS_IUS 1
106 #endif
107
108 /* User config */
109
110 #define NS_LOG_ERROR_ENABLE 1
111 #define NS_LOG_WARNING_ENABLE 1
112 #define NS_LOG_INFO_ENABLE 1
113 #define NS_LOG_DEBUG_ENABLE 0
114
115 #define NS_LOG_LPUART_ENABLE 1
116
117 #define NS_TIMER_ENABLE 1

```

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)

```

```

434 #else
435 #define BLE_RDTSS_16BIT_SERVER 0
436 #endif // defined(CFG_PRF_RDTSS_16BIT)
437
438 // Apple Notification Center Service Client Role
439 #if defined(CFG_PRF_ANCC)
440 #define BLE_ANC_CLIENT 1
441 #else
442 #define BLE_ANC_CLIENT 0
443 #endif //defined(CFG_PRF_ANCC)
444
445 // #define BLE_RDTSS_ENABLE (BLE_RDTSS_SERVER || BLE_RDTSS_16BIT_SERVER )
446
447
448 #if defined(CFG_PRF_RDTSS2) //添加服务2的宏定义
449 #define BLE_RDTSS2_SERVER 1
450 #else
451 #define BLE_RDTSS2_SERVER 0
452 #endif // defined(CFG_PRF_RDTSS2)
453
454 #define BLE_RDTSS_ENABLE (BLE_RDTSS_SERVER || BLE_RDTSS_16BIT_SERVER || BLE_RDTSS2_SERVER)
455
456
457 // BLE_CLIENT_PRF indicates if at least one client profile is present
458 #if (BLE_PROX_MONITOR || BLE_FINDME_LOCATOR || BLE_HT_COLLECTOR || BLE_BP_COLLECTOR \
459     BLE_HR_COLLECTOR || BLE_DIS_CLIENT || BLE_TIP_CLIENT || BLE_SP_CLIENT \
460     BLE_BATT_CLIENT || BLE_GL_COLLECTOR || BLE_HID_BOOT_HOST || BLE_HID_REPORT_HOST \
461     BLE_RSC_COLLECTOR || BLE_CSC_COLLECTOR || BLE_CP_COLLECTOR || BLE_LN_COLLECTOR || BLE_AN_CLIENT \
462     BLE_PAS_CLIENT || BLE_IPS_CLIENT || BLE_ENV_CLIENT || BLE_WSC_CLIENT \
463     BLE_UDS_CLIENT || BLE_BCS_CLIENT || BLE_WPT_CLIENT || BLE_PLX_CLIENT \
464     BLE_COM_CLIENT || BLE_DBG_THPP || BLE_RDTSS_CLIENT || BLE_ANC_CLIENT)
465 #define BLE_CLIENT_PRF 1
466 #else
467 #define BLE_CLIENT_PRF 0
468 #endif // (BLE_PROX_MONITOR || BLE_FINDME_LOCATOR ...)
469
470 // BLE_SERVER_PRF indicates if at least one server profile is present
471 #if (BLE_PROX_REPORTER || BLE_FINDME_TARGET || BLE_HT_THERMOM || BLE_BP_SENSOR \
472     BLE_TIP_SENSOR || BLE_IP_SENSOR || BLE_RSC_SERVER || BLE_BP_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

```

```

31 * @version v1.0.0
32 *
33 * @copyright Copyright (c) 2019, Nations Technologies Inc. All rights reserved.
34 */
35
36 /* Includes -----*/
37 #include "rwprf_config.h" // SW configuration
38
39 #if (BLE_RDTSS_ENABLE)
40 #include "rdts_common.h"
41 #endif
42 #if (BLE_RDTSS_SERVER)
43 #include "rdtss.h"
44 #endif // (BLE_RDTSS_SERVER)
45
46 #if (BLE_RDTSS2_SERVER) //添加服务2的头文件
47 #include "rdtss2.h"
48 #endif
49
50
51 #if (BLE_RDTSS_16BIT_SERVER)
52 #include "rdtss_16bit.h"
53 #endif // (BLE_RDTSS_16BIT_SERVER)
54
55 #include "gattc_task.h"
56 #include "att.h"
57 #include "attm_db.h"
58 #include "prf_types.h"
59 #include "prf_utils.h"
60 /* Private define -----*/

```

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->shdl + att_idx;
    }
}

```

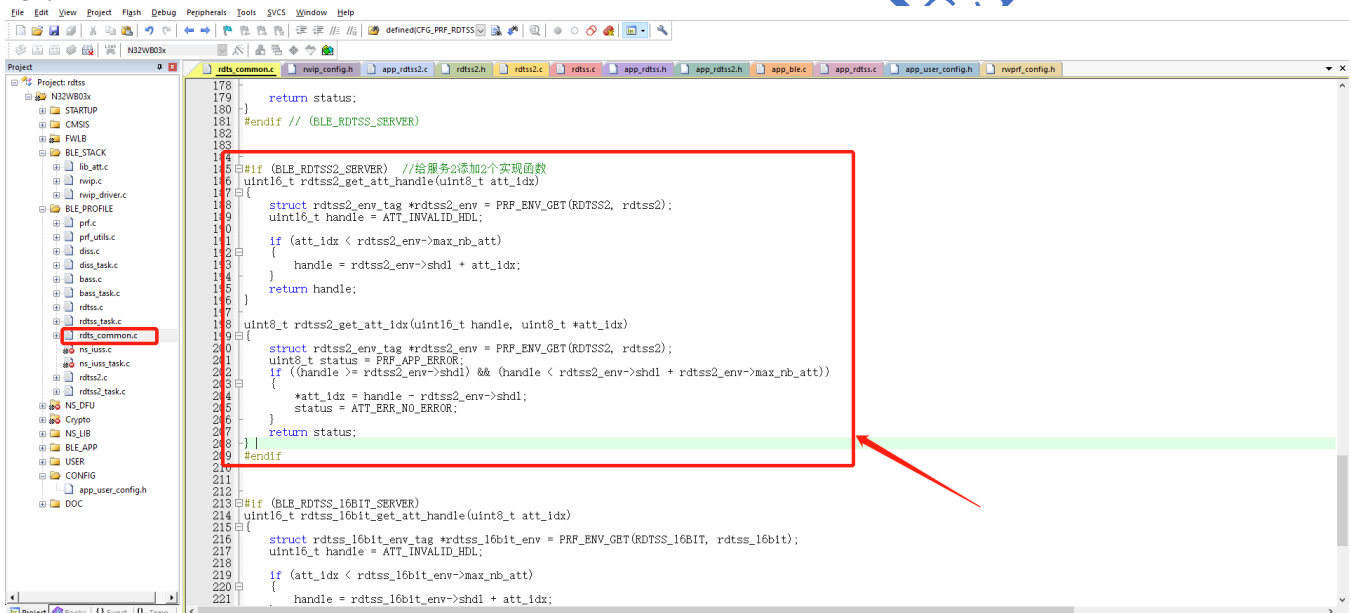
```

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

```

CONFIDENTIAL



5) 在rdts_common.h 参照#if (BLE_RDTSS_SERVER)使能,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)

```

```

76 * @param[out] att_idx attribute index
77 * @return high layer error code
78 */
79 uint8_t rdtss_get_att_idx(uint16_t handle, uint8_t *att_idx);
80 #endif // (BLE_RDTSS_SERVER)
81
82
83 #if (BLE_RDTSS2_SERVER) //给服务2添加两个函数声明
84 uint16_t rdtss2_get_att_handle(uint8_t att_idx);
85
86 uint8_t rdtss2_get_att_idx(uint16_t handle, uint8_t *att_idx);
87 #endif // (BLE_RDTSS2_SERVER)
88
89
90
91
92 #if (BLE_RDTSS16BIT_SERVER)
93 /**
94 * @brief Compute the handle of a given attribute based on its index
95 * @details Specific to raw data transfer server in 16bit uuid
96 * @param[in] att_idx attribute index
97 * @return the corresponding handle

```

6) 在rwip_task.h添加这个服务的任务号, 注意应不要重复, 这个例程最小的任务号为76。

TASK_ID_RDTSS2 = 76,

```

179
180 TASK_ID_UDSS = 61, // User Data Service Server Task
181 TASK_ID_UDSC = 62, // User Data Service Client Task
182
183 TASK_ID_BCSS = 63, // Body Composition Server Task
184 TASK_ID_BCSL = 64, // Body Composition Client Task
185
186 TASK_ID_WPTS = 65, // Wireless Power Transfer Profile Server Task
187 TASK_ID_WPTC = 66, // Wireless Power Transfer Profile Client Task
188
189 TASK_ID_PLXS = 67, // Pulse Oximeter Profile Server Task
190 TASK_ID_PLXC = 68, // Pulse Oximeter Profile Client Task
191
192 TASK_ID_CGMS = 69, // Continuous Glucose Monitoring Server Task
193 TASK_ID_CGMC = 70, // Continuous Glucose Monitoring Client Task
194
195 TASK_ID_RDTSS = 71, // Raw data transfer server task
196 TASK_ID_RDTSS2 = 76, // 给服务2添加服务的任务号, 不能重复, 所以写76
197 TASK_ID_RDTSS16BIT = 72, // Raw data transfer server in 16bit uuid task
198
199 TASK_ID_NS_IUS = 73,
200 TASK_ID_RDTSC = 74, // Raw data transfer client task
201
202 TASK_ID_ANCC = 75, // Apple Notification Center Service Client Task
203
204 TASK_ID_MESH = 200, // Mesh Task
205
206 /* 240 -> 241 reserved for Audio Mode 0 */
207 TASK_ID_AMO = 240, // BLE Audio Mode 0 Task
208 TASK_ID_AMO_HAS = 241, // BLE Audio Mode 0 Hearing Aid Service Task
209
210
211
212
213
214 TASK_ID_THPP = 242, // Throughput profile tester used for debugging
215
216 TASK_ID_INVALID = 0xFF // Invalid Task Identification

```

7) 添加 profile 层库文件, middlewares\Nationstech\ble_library\ns_ble_profile\rdts\rdtss 目录下, 复制 rdtss.c, rdtss.h, rdtss_task.c, rdtss_task.h 四个文件为 rdtss2.c, rdtss2.h, rdtss2_task.c, rdtss2_task.h, 修改里面 RDTSS/rdtss 相关字段为 RDTSS2/rdtss2, 避免重名声明。(注意检查是否替换彻底)

rdtss2.c 和 rdtss2_task.c 文件路径: middlewares\Nationstech\ble_library\ns_ble_profile\rdts\rdtss\src

middlewares > Nationstech > ble_library > ns_ble_profile > rdts > rdtss > src

名称	修改日期	类型	大小
rdtss.c	2022/5/13 14:51	C 源文件	8 KB
rdtss_16bit.c	2022/5/13 14:51	C 源文件	9 KB
rdtss_16bit_task.c	2022/5/13 14:51	C 源文件	29 KB
rdtss_task.c	2022/5/13 14:51	C 源文件	28 KB
rdtss2.c	2022/11/14 11:02	C 源文件	8 KB
rdtss2_task.c	2022/11/14 11:03	C 源文件	28 KB

rdtss2.h 和 rdtss2_task.h 文件路径: middlewares\Nationstech\ble_library\ns_ble_profile\rdts\rdtss\api

middlewares > Nationstech > ble library > ns_ble_profile > rdts > rdtss > api				
名称	修改日期	类型	大小	
rdtss.h	2022/5/13 14:51	C Header 源文件	4 KB	
rdtss_16bit.h	2022/5/13 14:51	C Header 源文件	4 KB	
rdtss_16bit_task.h	2022/5/13 14:51	C Header 源文件	9 KB	
rdtss_task.h	2022/5/13 14:51	C Header 源文件	9 KB	
rdtss2.h	2022/11/14 11:01	C Header 源文件	4 KB	
rdtss2_task.h	2022/11/14 11:00	C Header 源文件	9 KB	

参考以下方式, 把文件里面的 RDTSS/rdtss 相关字段替换为 RDTSS2/rdtss2
用 RDTSS2 替换文件中的 RDTSS:



1、打开文件

2、找到要替换的文本

3、选中要替换的文本

4、替换为 RDTSS2

5、确认替换

```

1  // *****
2  * Copyright (c) 2019, Nations Technologies Inc.
3  * All rights reserved.
4  *
5  *
6  *
7  * Redistribution and use in source and binary forms, with or without
8  * modification, are permitted provided that the following conditions are met:
9  *
10 * - Redistributions of source code must retain the above copyright notice,
11 * this list of conditions and the disclaimer below.
12 *
13 * Nations' name may not be used to endorse or promote products derived from
14 * this software without specific prior written permission.
15 *
16 * DISCLAIMER: THIS SOFTWARE IS PROVIDED BY NATIONS "AS IS" WITHOUT ANY
17 * IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
18 * OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
19 * IN NO EVENT SHALL NATIONS BE LIABLE FOR ANY DAMAGES, INCLUDING, BUT NOT
20 * LIMITED TO, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES,
21 * INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES,
22 * LOSS OF PROFITS, OR BUSINESS INTERRUPTION, HOWEVER CAUSED AND, WHETHER IN
23 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
24 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
25 * POSSIBILITY OF SUCH DAMAGE.
26 * *****
27
28 // **
29 * @file rdtss2.h
30 * @author Nations Firmware Team
31 * @version v1.0.1
32 *
33 * @copyright Copyright (c) 2019, Nations Technologies Inc. All rights reserved.
34 *
35 #ifndef RDTSS2_H_
36 #define RDTSS2_H_
37
38
39 #include "rwip_config.h" // SW configuration
40
41 #if (BLE_RDTSS2_SERVER)
42
43 /* Includes -----*/
44 #include "catdint.h"
45 #include "prf_types.h"
46 #include "prf.h"
47 #include "atm.h"
48
49 #include "rdts_common.h"
50
51 #define RDTSS2_IDX_MAX (1)
52
53 /* Public typedef -----*/
54 // Parameters for the database creation

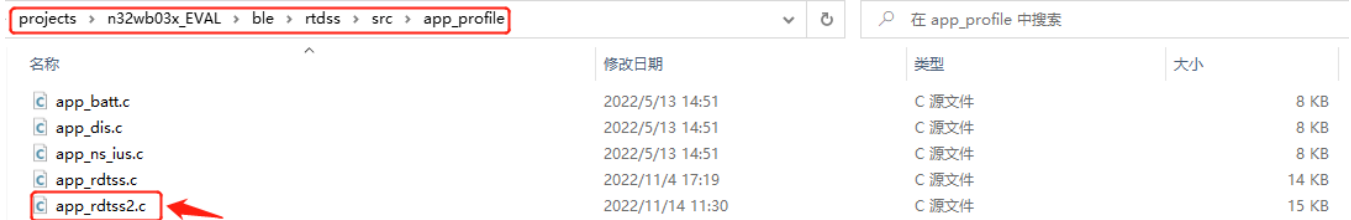
```

用 rdtss2 替换文件中的 rdtss:

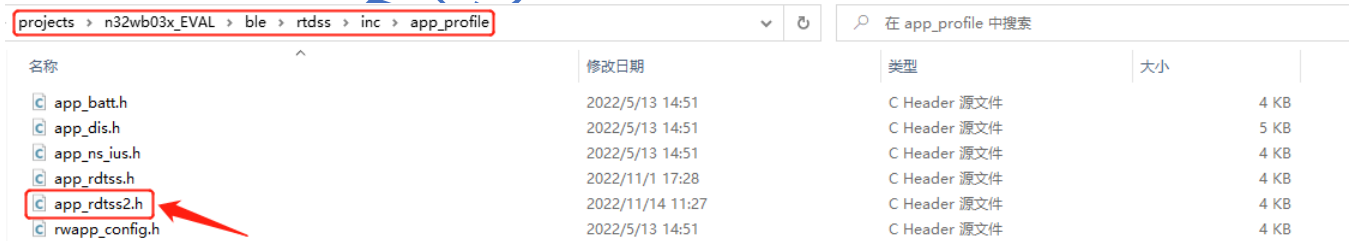


8) 添加profile应用层代码，在例程的app_profile目录下复制app_rdtss.c和app_rdtss.h为app_rdtss2.c和app_rdtss2.h，参考上述方式改里面RDTSS/rdtss相关字段为RDTSS2/rdtss2，避免重名声明。(注意检查是否替换彻底)

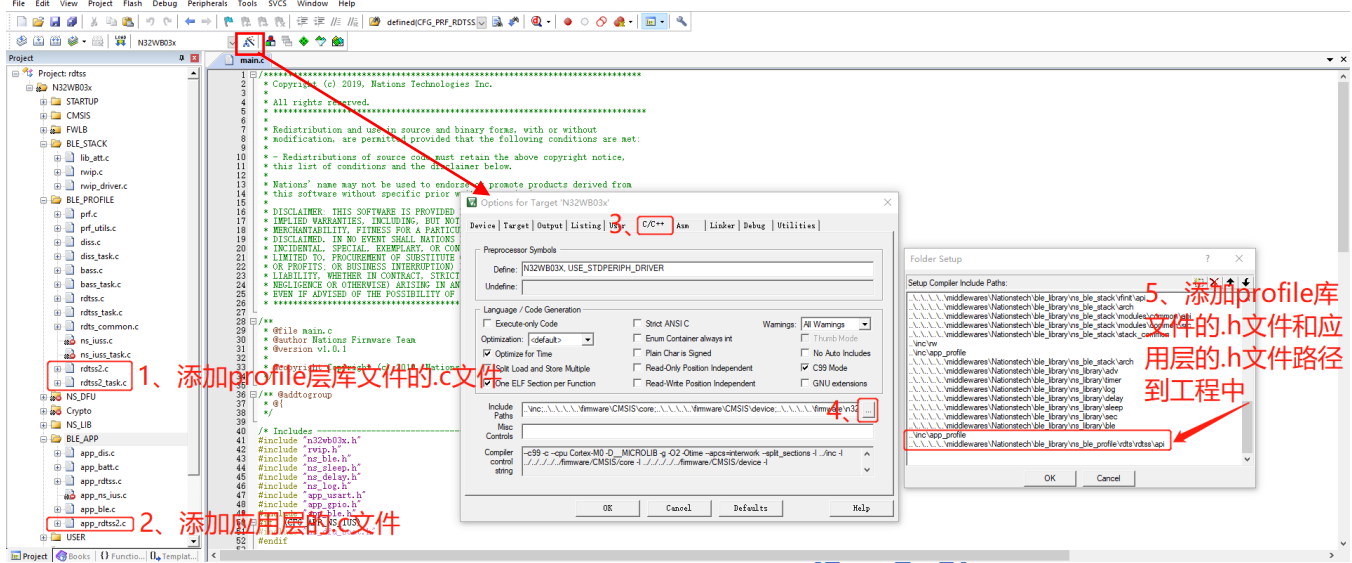
app_rdtss2.c文件路径: projects\n32wb03x_EVAL\ble\rdtss\src\app_profile



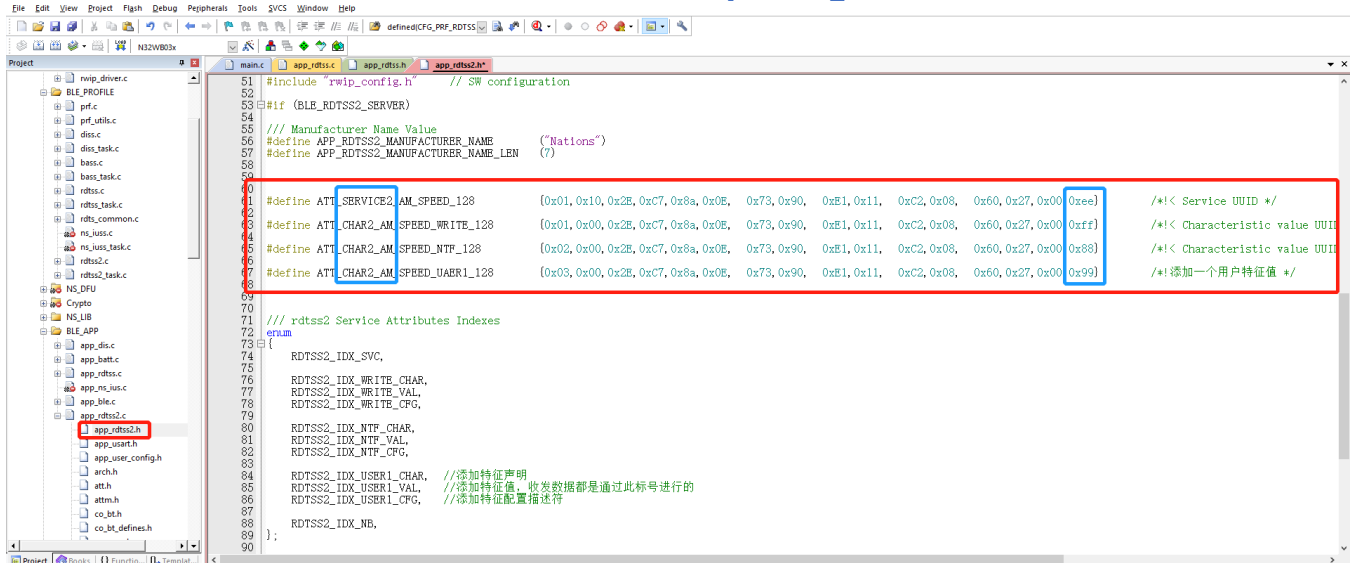
app_rdtss2.h 文件路径: projects\n32wb03x_EVAL\ble\rdtss\inc\app_profile



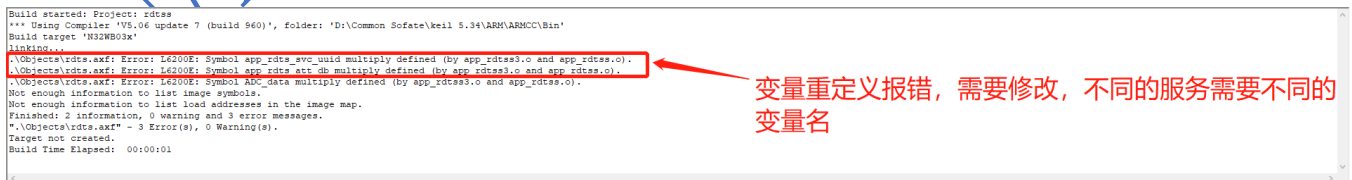
9) 把上述修改的profile层库文件和应用层的.c源文件添加到工程相应的目录，即BLE_PROFILE和BLE_APP目录，并参照上述头文件路径，把.h文件添加到工程中。



10) 请参考第1节的内容修改UUID，如需更改权限请参考第2节的内容。修改服务2的服务和特征值的宏定义和UUID值，避免和服务1重复定义：



添加好.c文件和.h文件之后，进行编译，出现变量重定义报错，需要进行变量名修改，避免重定义。



修改服务 2 相关的配置数据和赋值数据：在 app_rdtss2.c 文件中包含头文件#include "app_rdtss2.h"，再修改相应变量名称为：app_rdtss2_svc2_uuid[16]和 app_rdtss2_att_db[RDTSS2_IDX_NB]：

```

61 void rdtss2_send_notify_USER1(uint8_t *data, uint16_t length);
62
63 #STARTUP
64 const uint8_t app_rdtss2_svc2_uuid[16] = ATT_SERVICE2_AM_SPEED_128;
65 const struct att_db_desc_t app_rdtss2_att_db[RDTSS2_IDX_NB] =
66 {
67     /* Service Declaration */
68     {0} = {0x00, 0x00}, PERM(0D, ENABLE), PERM(WRITE_REQ, ENABLE), 0, 0, 1,
69     /* Characteristic Declaration */
70     {1} = {0x03, 0x28}, PERM(0D, ENABLE) | PERM(WRITE_REQ, ENABLE), 0, 0, 1,
71     /* Characteristic Declaration */
72     {2} = {ATT_CHAR2_AM_SPEED_WRITE_128, PERM(WRITE_COMMAND, ENABLE), PERM(RI, ENABLE) | PERM_VAL(UUID_LEN, 0x02), 0x200, 1},
73     /* Client Characteristic Configuration Descriptor */
74     {3} = {0x01, 0x00}, PERM(0D, ENABLE) | PERM(WRITE_REQ, ENABLE), PERM(RI, ENABLE), 20, 1,
75     /* Characteristic Declaration */
76     {4} = {0x03, 0x28}, PERM(0D, ENABLE) | PERM(WRITE_REQ, ENABLE), 0, 0, 1,
77     /* Characteristic Declaration */
78     {5} = {ATT_CHAR2_AM_SPEED_WRITE_128, PERM(WRITE_COMMAND, ENABLE), PERM(RI, ENABLE) | PERM_VAL(UUID_LEN, 0x02), 0x200, 1},
79     /* Client Characteristic Configuration Descriptor */
80     {6} = {0x01, 0x00}, PERM(0D, ENABLE) | PERM(WRITE_REQ, ENABLE), PERM(RI, ENABLE), 20, 1,
81     /* Characteristic Declaration */
82     {7} = {ATT_CHAR2_AM_SPEED_WRITE_128, PERM(WRITE_COMMAND, ENABLE), PERM(RI, ENABLE) | PERM_VAL(UUID_LEN, 0x02), 0x200, 1},
83     /* Client Characteristic Configuration Descriptor */
84     {8} = {0x01, 0x00}, PERM(0D, ENABLE) | PERM(WRITE_REQ, ENABLE), PERM(RI, ENABLE), 20, 1,
85     /* USER1 Characteristic Declaration */
86     {9} = {0x03, 0x28}, PERM(0D, ENABLE) | PERM(WRITE_REQ, ENABLE), 0, 0, 1,
87     /* USER1 Characteristic Declaration */
88     {10} = {ATT_CHAR2_AM_SPEED_WRITE_128, PERM(WRITE_COMMAND, ENABLE), PERM(RI, ENABLE) | PERM_VAL(UUID_LEN, 0x02), 0x200, 1},
89     /* Client Characteristic Configuration Descriptor */
90     {11} = {0x01, 0x00}, PERM(0D, ENABLE) | PERM(WRITE_REQ, ENABLE), PERM(RI, ENABLE), 20, 1,
91 };
92
93 void app_rdtss2_add_rdtss(void)
94 {
95     struct rdtss2_db_cfg *db_cfg;
96     struct gatt_profile_task_add_cmd *req = KE_MSG_ALLOC_DYN(GAPM_PROFILE_TASK_ADD_CMD,
97     TASK_APP,
98     gatt_profile_task_add_cmd,
99     sizeof(struct rdtss2_db_cfg));
100
101     // Fill message
102     req->operation = GAPM_PROFILE_TASK_ADD;
103     req->sec_lvl = PERM(CYC, AUTH, NO_AUTH);
104     req->prf_task_id = TASK_ID_RDTSS2;
105     req->app_task = TASK_APP;
106     req->att_hdl = 0;
107
108     // Set parameters
109     db_cfg = (struct rdtss2_db_cfg *) req->param;
110     // Assignable table, in case the handle offset needs to be saved
111     db_cfg->att_tbl = &app_rdtss2_att_db[0];
112     db_cfg->svc_uuid = &app_rdtss2_svc2_uuid[0];
113     db_cfg->max_att = RDTSS2_IDX_NB;
114
115     // Send the message
116     ke_msg_send(req);
117
118     app_rdtss2_init();
119 }
120
121 //++
122 // @brief rdtss2 value require indicate handler
123 // @param
124 // @return

```

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 ns_ble_add_prf_func_register(app_rdtss2_add_rdtss)注册新增服务。

```

241 ns_sec_init(&sec_init);
242
243
244 void app_ble_prf_init(void)
245 {
246     #if (BLE_APP_DIS)
247         //add device informaiton server
248         ns_ble_add_prf_func_register(app_dis_add_dis);
249     #endif //BLE_APP_DIS
250     #if (BLE_APP_BATT)
251         //add battery level server
252         ns_ble_add_prf_func_register(app_batt_add_bas);
253     #endif //BLE_APP_BATT
254     #if (BLE_APP_NS_IUS)
255         ns_ble_add_prf_func_register(app_ns_ius_add_ns_ius);
256     #endif //BLE_APP_NS_IUS
257
258     //add raw data transmit server(rdtss)
259     ns_ble_add_prf_func_register(app_rdtss_add_rdtss);
260
261     ns_ble_add_prf_func_register(app_rdtss2_add_rdtss); //为服务2注册一个新的服务
262
263
264
265
266 //++

```

NA

12) 至此服务已经添加完成，后续可以在rdtss2_val_write_ind_handler的RDTSS2_IDX_WRITE_VAL事件回调获取从机接收到的数据和通过rdtss2_send_notify函数向主机发送数据。

通过RDTSS2_IDX_WRITE_VAL标号接收主机下发的数据

```

178 static int rdtss2_val_write_ind_handler(ke_msg_id_t const msgid,
179 struct rdtss2_val_write_ind const *ind_value,
180 ke_task_id_t const dest_id,
181 ke_task_id_t const src_id)
182 {
183     NS_LOG_DEBUG("%s.write handle = %x,length = %x\r\n",__func__,ind_value->handle, ind_value->length);
184     for(uint16_t i=0; i<ind_value->length; i++)
185     {
186         NS_LOG_INFO("%x ",ind_value->value[i]);
187     }
188     NS_LOG_INFO("\r\n");
189     rdtss2_send_notify((uint8_t *)ind_value->value,ind_value->length); //通过ATT_CHAR2_AM_SPEED_NTF_128特征值的RDTSS2_IDX_VAL标号给APP发数据
190     rdtss2_send_notify_USER1((uint8_t *)ind_value->value,ind_value->length); //通过ATT_CHAR2_AM_SPEED_UAER1_128特征值的RDTSS2_IDX_USER1_VAL标号给APP发数据
191     uint16_t handle = ind_value->handle;
192     uint16_t length = ind_value->length;
193     switch (handle)
194     {
195     case RDTSS2_IDX_NTF_CFG: //APP打开ATT_CHAR2_AM_SPEED_NTF_128特征值的通知权限，从机通过此标号接收
196         NS_LOG_INFO("enter RDTSS2_IDX_NTF_CFG\r\n");
197         if(length == 2)
198         {
199             uint16_t cfg_value = ind_value->value[0] + ind_value->value[1];
200             if(cfg_value == PRF_CLI_START_NTF)//APP打开了通知权限
201             {
202                 //enabled notify
203             }
204             else if(cfg_value == PRF_CLI_STOP_NTFIND)//APP关闭了通知权限
205             {
206             }
207         }
208         break;
209     case RDTSS2_IDX_USER1_CFG: //APP打开ATT_CHAR2_AM_SPEED_UAER1_128新特征值的通知权限，从机通过此标号接收
210         NS_LOG_INFO("enter RDTSS2_IDX_USER1_CFG\r\n");
211         if(length == 2)
212         {
213             uint16_t cfg_value = ind_value->value[0] + ind_value->value[1];
214             if(cfg_value == PRF_CLI_START_NTF)//APP打开了通知权限
215             {
216                 //enabled notify
217             }
218             else if(cfg_value == PRF_CLI_STOP_NTFIND)//APP关闭了通知权限
219             {
220             }
221         }
222         break;
223     case RDTSS2_IDX_WRITE_VAL: //APP通过ATT_CHAR2_AM_SPEED_WRITE_128的写权限下发数据，从机通过此标号接收
224         NS_LOG_INFO("enter RDTSS2_IDX_WRITE_VAL\r\n");
225         //ble data receive
226         app_uart_tx_fifo_enter(ind_value->value,ind_value->length); //从机接收到的APP下发的数据，通过USART1发送给主控MCU
227         break;
228     }
229 }
    
```

接收主机下发的数据

数据发送，例如：调用 rdtss2_send_notify()函数向主机发送数据

```

177 * @note
178 *
179 uint8_t str_adc[] = {0x22,0x33}; // 用户数据
180 static int rdtss2_val_write_ind_handler(ke_msg_id_t const msgid,
181 struct rdtss2_val_write_ind const *ind_value,
182 ke_task_id_t const dest_id,
183 ke_task_id_t const src_id)
184 {
185     NS_LOG_DEBUG("%s.write handle = %x,length = %x\r\n",__func__,ind_value->handle, ind_value->length);
186     for(uint16_t i=0; i<ind_value->length; i++)
187     {
188         NS_LOG_INFO("%x ",ind_value->value[i]);
189     }
190     NS_LOG_INFO("\r\n");
191     rdtss2_send_notify((uint8_t *)ind_value->value,ind_value->length); //通过ATT_CHAR2_AM_SPEED_NTF_128特征值的RDTSS2_IDX_NTF_VAL标号给APP发数据
192     rdtss2_send_notify_USER1((uint8_t *)str_adc,sizeof(str_adc)); //通过ATT_CHAR2_AM_SPEED_UAER1_128特征值的RDTSS2_IDX_USER1_VAL标号给APP发数据
193     uint16_t handle = ind_value->handle;
194     uint16_t length = ind_value->length;
195     switch (handle)
196     {
197     case RDTSS2_IDX_NTF_CFG: //APP打开ATT_CHAR2_AM_SPEED_NTF_128特征值的通知权限，从机通过此标号接收
198         NS_LOG_INFO("enter RDTSS2_IDX_NTF_CFG\r\n");
199         if(length == 2)
200         {
201             uint16_t cfg_value = ind_value->value[0] + ind_value->value[1];
202             if(cfg_value == PRF_CLI_START_NTF)//APP打开了通知权限
203             {
204                 //enabled notify
205             }
206             else if(cfg_value == PRF_CLI_STOP_NTFIND)//APP关闭了通知权限
207             {
208             }
209         }
210         break;
211     case RDTSS2_IDX_USER1_CFG: //APP打开ATT_CHAR2_AM_SPEED_UAER1_128新特征值的通知权限，从机通过此标号接收
212         NS_LOG_INFO("enter RDTSS2_IDX_USER1_CFG\r\n");
213         if(length == 2)
214         {
215             uint16_t cfg_value = ind_value->value[0] + ind_value->value[1];
216             if(cfg_value == PRF_CLI_START_NTF)//APP打开了通知权限
217             {
218                 //enabled notify
219             }
220             else if(cfg_value == PRF_CLI_STOP_NTFIND)//APP关闭了通知权限
221             {
222             }
223         }
224         break;
225     }
226 }
    
```

例如：使用服务2的第1个特征值把接收到的数据回发给主机

例如：使用服务2的第2个特征值把用户数据上发给主机

4 服务添加成功后实际效果



5 历史版本

版本	日期	备注
V1.0	2022.11.14	新建文档

NATIONS CONFIDENTIAL

6 声明

国民技术股份有限公司（以下简称国民技术）保有在不事先通知而修改这份文档的权利。国民技术认为提供的信息是准确可信的。尽管这样，国民技术对文档中可能出现的错误不承担任何责任。在购买前请联系国民技术获取该器件说明的最新版本。对于使用该器件引起的专利纠纷及第三方侵权国民技术不承担任何责任。另外，国民技术的产品不建议应用于生命相关的设备和系统，在使用该器件中因为设备或系统运转失灵而导致的损失国民技术不承担任何责任。国民技术对本手册拥有版权等知识产权，受法律保护。未经国民技术许可，任何单位及个人不得以任何方式或理由对本手册进行使用、复制、修改、抄录、传播等。

NATIONS CONFIDENTIAL