CMT453x 数据传输例程应用指南

简介

本文档旨在让开发者快速了解CMT453x系列蓝牙SoC数据传输例程ble_rdts_peripheral和ble_rdts_central的软件框架及工作原理，以减少开发难度。

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 例程介绍 4

2 例程演示 5

2.1 数据传输从机与手机间通信： 5

2.2 数据传输从机与数据传输主机间通信： 7

3 数据传输服务（RDTS Service） 10

3.1 GATT Service 介绍 10

3.2 数据流分析-手机与数据传输从机通信 10

3.3 数据流分析-数据传输主从机通信 12

4 数据传输例程项目结构 14

4.1 Project Target 14

4.2 目录结构如下 14

5 代码流程分析 16

5.1 数据传输从设备代码流程分析 16

5.2 数据传输主设备代码流程分析 19

版本历史 21

1 例程介绍

蓝牙数据传输例程包含central和peripheral两部分，在SDK中分别提供有ble_rdts_central和ble_rdts_peripheral例程。两设备间通过蓝牙建立连接，将设备端经串口所接收到的数据通过蓝牙发送至对端，同时，在收到对端蓝牙所发送的数据后，经串口输出显示。

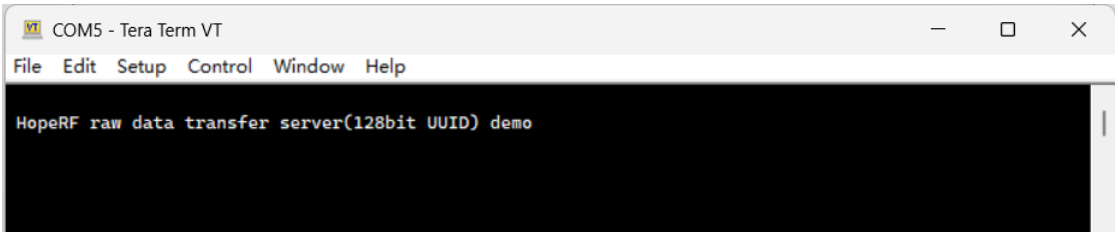
为方便测试，用户也可以使用手机或平板等作为central端设备，扫描并连接ble_rdts_peripheral设备进行数据传输。

2 例程演示

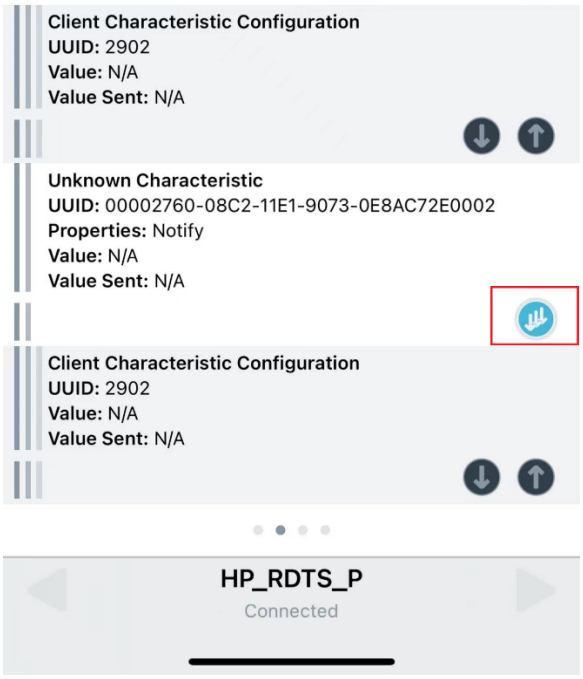
2.1 数据传输从机与手机间通信：

1. 使用标准 J-Link 工具连接 CMT4531 dongle 开发板。
2. 将 dongle 通过 USB 连接至电脑，此时电脑将自动识别 dongle 板上的 USB 转串口设备并将其枚举为标准串口。
3. 插上 TXD/RXD 跳线帽以连接串口。
4. 连接 VDD 跳线帽给 dongle 板上的 HM-BT4531B 供电。
5. 在 PC 端通过串口工具（例如，Tera Term）连接 dongle 板的串口，参数为 115200 8N1。
6. 打开项目：

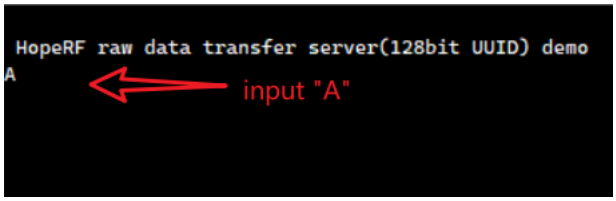
   \projects\cmt453x_EVAL\ble_peripheral\ble_rdts_peripheral\MDK-ARM\ble_rdts_peripheral.uvprojx
7. 编译项目，并将编译生成的固件烧录至开发板。
8. 此时在串口工具上将会收到如下打印信息

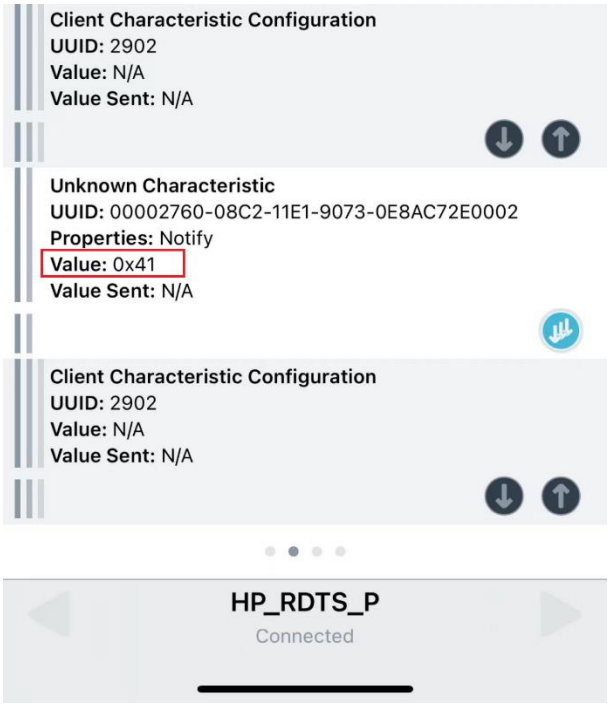


9. 在手机端使用 App 扫描 BLE 设备，连接设备“HP_RDTS_P”。（如果无法扫描到设备，请复位 dongle 板或重新上电）
10. 在 App 上使能特征值 0x02002EC78a0E-7390-E111-C208-60270000 的通知权限。

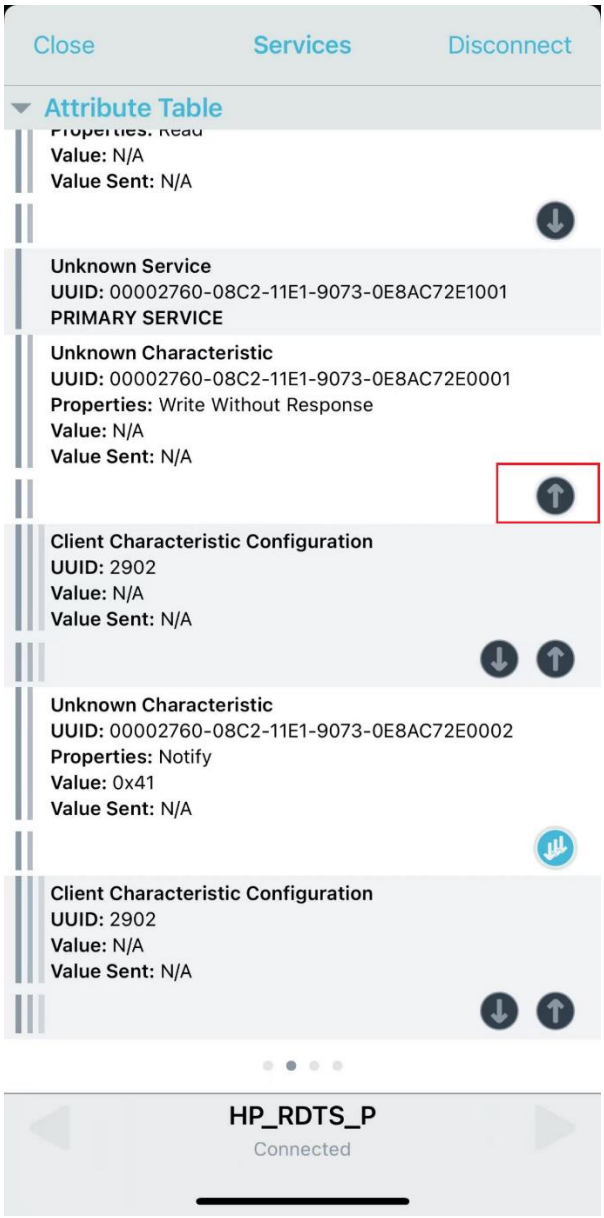


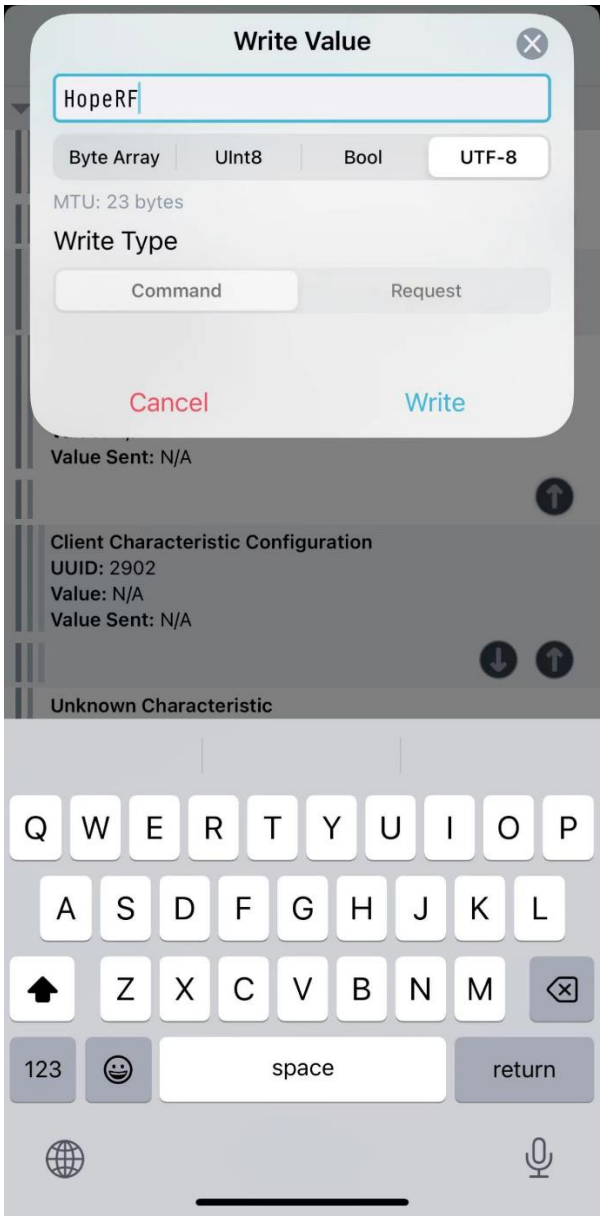
11. 在 PC 串口端输入任意字符或字符串，App 将显示所接收到的数据。





12. 在 App 上向特征值 0x01002EC78a0E-7390-E111-C208-60270000 写入数据，相应的数据将会在串口上显示。





2.2 数据传输从机与数据传输主机间通信：

上述章节演示了如何使用数据传输从机ble_rdts_peripheral与手机进行通信，本节我们将介绍数据传输从机ble_rdts_peripheral与主机ble_rdts_central间的通信。

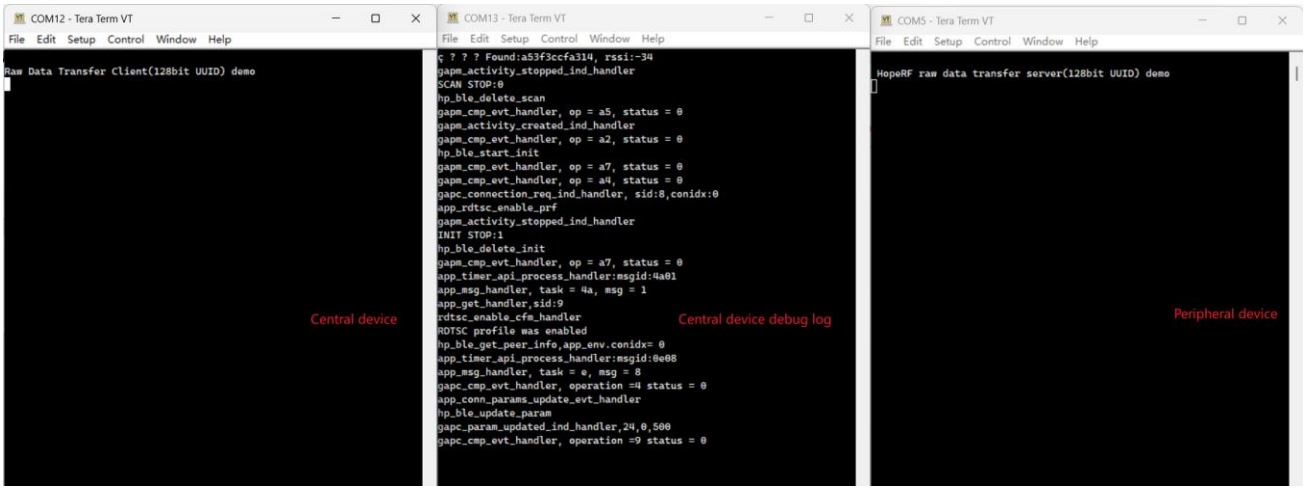
关于数据传输从机设备的演示搭建，请参考2.1节的步骤1至7。

以下步骤演示如何搭建数据传输主机设备。

1. 使用标准 J-Link 工具连接 CMT4531 dongle 开发板。
2. 将 dongle 通过 USB 连接至电脑，此时电脑将自动识别 dongle 板上的 USB 转串口设备并将其枚举为标准串口。
3. 插上 TXD/RXD 跳线帽以连接串口。
4. 连接 VDD 跳线帽给 dongle 板上的 HM-BT4531B 供电。
5. 在 PC 端通过串口工具（例如，Tera Term）连接 dongle 板的串口，参数为 115200 8N1。
6. 打开项目：

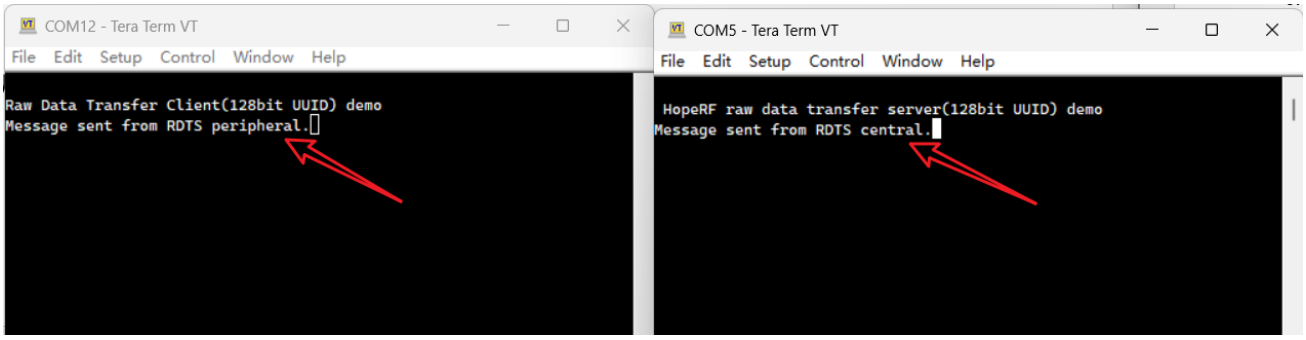
   \projects\cmt453x_EVAL\ble_central\ble_rdts_central\MDK-ARM\ble_rdts_central.uvprojx

7. 编译项目，并将编译生成的固件烧录至开发板。
8. 保持 ble_rdts_peripheral 设备正常上电状态,此时 ble_rdts_central 设备将会自动扫描并查找设备名为“HP_RDTS_P”的从设备，然后建立连接。注意，CMT4531 dongle 开发板上的串口已经预留用于数据传输，如果需要查看代码运行过程中的调试信息，请监控 PB1 上的串口打印输出。



9. 连接建立之后，在 ble_rdts_central 和 ble_rdts_peripheral 数据传输串口端输入任意数据，数据将会通过蓝牙传递至对端，并通过串口输出显示在终端上。

   在 central 端串口输入“Message sent from RDTS central.”，相应的信息将会出现在 peripheral 端设备的串口终端上。反之亦然。


COM12 - Tera Term VT
File Edit Setup Control Window Help
Raw Data Transfer Client(128bit UUID) demo
Message sent from RDTS peripheral.
COM5 - Tera Term VT
File Edit Setup Control Window Help
HopeRF raw data transfer server(128bit UUID) demo
Message sent from RDTS central.

3 数据传输服务（RDTS Service)

3.1 GATT Service介绍

数据传输服务（RDTS Service, raw data transfer service）用于收发BLE数据，其服务和特征值都使用自定义的128bit UUID，具体如下表格所示：

Service/Characteristic	UUID	Properties
Service	01102EC78a0E-7390-E111-C208-60270000	ATT_CHAR_PROP_RD
Characteristic RX data	01002EC78a0E-7390-E111-C208-60270000	ATT_CHAR_PROP_WR_NO_RESP
Characteristic TX data	02002EC78a0E-7390-E111-C208-60270000	ATT_CHAR_PROP_NTF

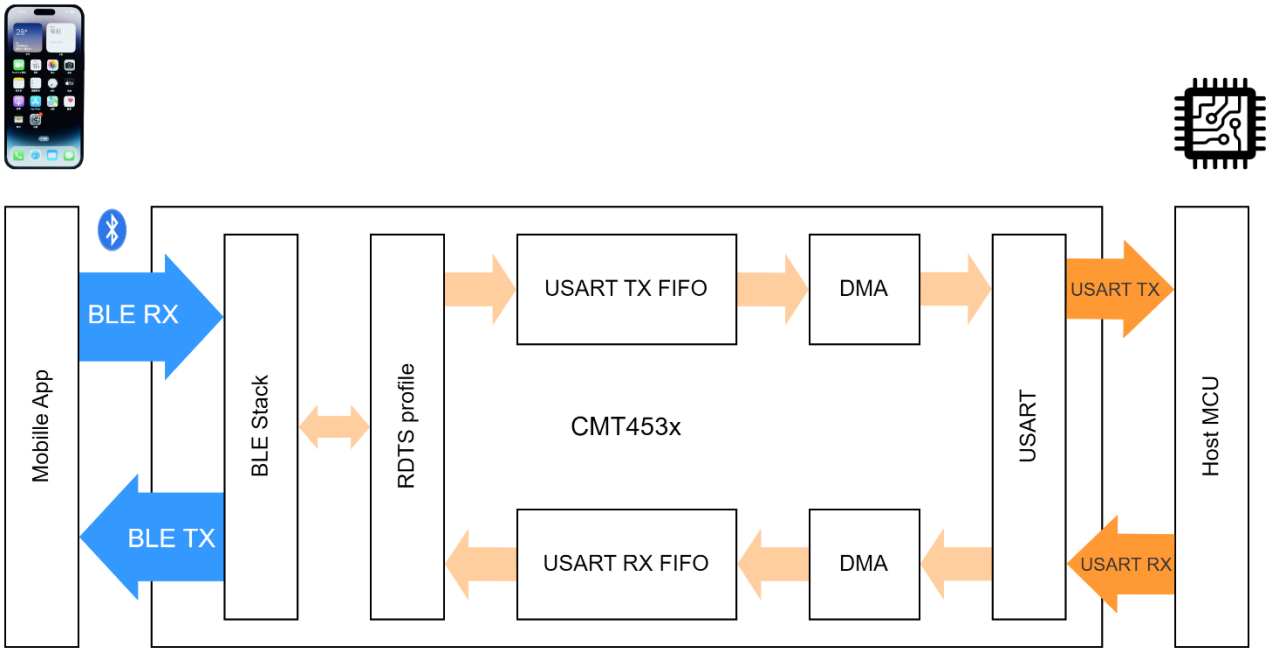
注意：

手机App通过“write with no respond”向从设备下发数据，从设备通过“Notify”上报数据到手机App。

3.2 数据流分析-手机与数据传输从机通信

这里以手机作为主设备与从机建立连接，并通过执行Write without response操作向从设备写数据。从设备通过串口（USART）和外部的上位机通信，上位机可以是host MCU，也可以是支持串口功能的PC。从设备经串口接收到上位机传递的数据之后，通过Notification操作向主设备上报数据。在示例代码中，为了提高数据传输的效率，减小CPU的工作量，我们使用了硬件DMA来进行数据操作。

数据传输的路径如下图所示：



- **BLE 接收**

  在回调函数 `rdtss_val_write_ind_handler()`下的 `RDTSS_IDX_WRITE_VAL` 消息获取 BLE 下发的数据，在此调用 `app_usart_tx_fifo_enter()`函数把数据存入 USART TX FIFO（`usart_tx_fifo_buf`），后续步骤参考串口发送相关描述。

- **串口发送**

  从设备通过 BLE 接收到数据之后，调用 `app_usart_tx_fifo_enter()`将数据存入 USART TX FIFO, 并通过 `ke_timer_set()`创建一个 50ms 后执行的事件用于发送串口数据。如果在 50ms 内有通过 BLE 接收到新的数据，该定时器会被取消并重新创建。因此，只有在 BLE 接收数据断流（50ms 没有新数据）后，会调用 `app_usart_tx_process` 函数把 USART TX FIFO (`usart_tx_fifo_buf`)中的数据通过串口 DMA 发出至上位机 MCU 或 PC 端。在接收到 DMA 发送完成的中断后，在中断服务函数 `DMA_Channel1_2_3_4_IRQHandler()`中再次调用串口发送函数，直到 USART TX FIFO 所有的数据发送完成。

- **串口接收**

  串口通过 DMA 回环模式接收，并使用三个中断服务函数 (DMA_TC, DMA_HT, USART_IDLE) 调用 `usart_rx_check_in_irq()`函数检查 DMA 已传输完成的数据，并通过 `app_usart_rx_data_fifo_enter()`函数把已收到的数据转移到 USART RX FIFO (`usart_rx_fifo_buf`)，后续步骤参考 BLE 发送描述。

- **BLE 发送**

  在 USART 接收数据断流（10ms 没有新数据）后，调用 `usart_forward_to_ble_loop()`函数检查 USART RX FIFO(`usart_rx_fifo_buf`)，然后把数据通过 `rdtss_send_notify()`函数发送到手机

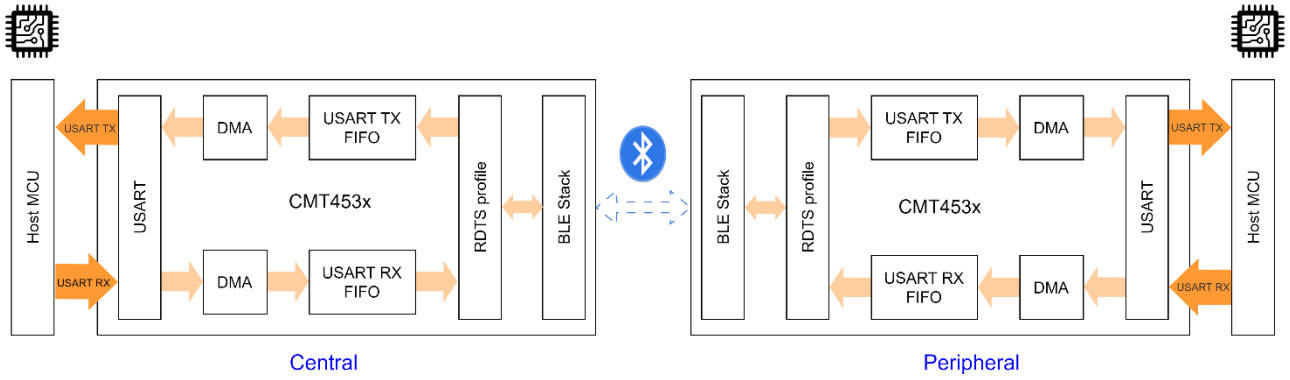
端。在每一包数据发送完成的回调函数 rdtss_val_ntf_cfm_handler 中再次调用 usart_forward_to_ble_loop()函数直到所有数据发送完成。

3.3 数据流分析-数据传输主从机通信

接下来，我们来讲解数据传输主机ble_rdts_central与数据传输从机ble_rdts_peripheral间通信过程的数据流。

数据传输的路径如下图所示：

注意：在此章节中我们主要讨论数据流，有关主、从设备间如何建立蓝牙连接请查看第5章节代码流程分析。同时，从设备的数据流分析请参考上一节所述。



主设备端数据流的处理同样分为四个阶段。

- **BLE 接收**

  在回调函数 rdtsc_data_rx_ind_handler()中获取 BLE 从设备所上报的数据，在此调用 app_usart_tx_fifo_enter()函数把数据存入 USART TX FIFO（usart_tx_fifo_buf），后续步骤参考串口发送相关描述。

- **串口发送**

  主设备通过 BLE 接收到从设备上报的数据后，调用 app_usart_tx_fifo_enter()将数据存入 USART TX FIFO 并通过 ke_timer_set()创建一个 50ms 后执行的事件用于发送串口数据。如果在 50ms 内有通过 BLE 接收到新的数据，该定时器会被取消并重新创建。因此，只有在 BLE 接收数据断流（50ms 没有新数据）后，会调用 app_usart_tx_process()函数将 USART TX FIFO (usart_tx_fifo_buf)中的数据通过串口 DMA 发出至上位机 MCU 或 PC 端。

- **串口接收**

  串口通过 DMA 回环模式接收，并使用三个中断服务函数 (DMA_TC, DMA_HT, USART_IDLE) 调用 usart_rx_check_in_irq()函数检查 DMA 已传输完成的数据，并通过 app_usart_rx_data_fifo_enter()函数把已收到的数据转移到 USART RX FIFO (usart_rx_fifo_buf)，后续步骤参考 BLE 发送描述。

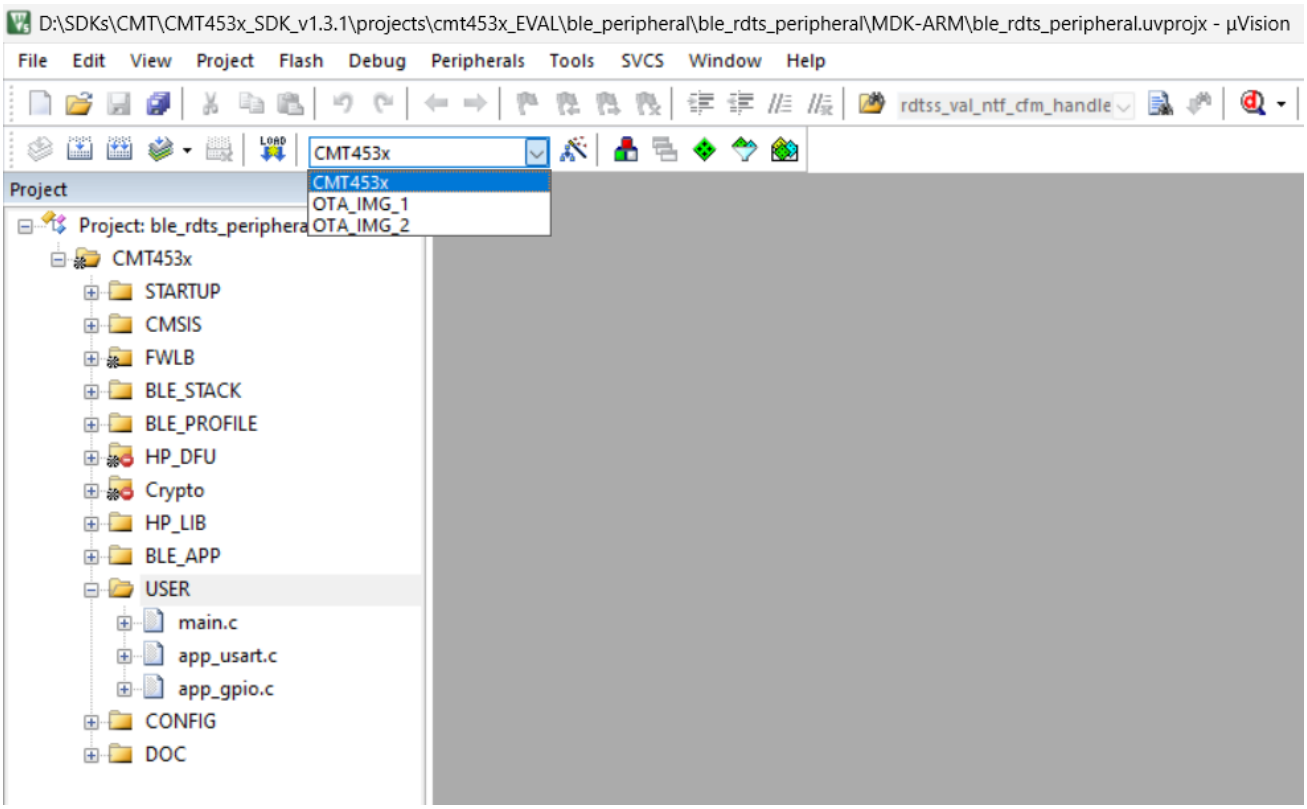
- **BLE 发送**

  在USART接收数据断流（10ms没有新数据）后，调用usart_forward_to_ble_loop()函数检查USART RX FIFO(usart_rx_fifo_buf)，然后把数据通过user_send_ble_data()接口并最终调用rdtsc_data_tx_req_handler()以GATTC_WRITE_NO_RESPONSE的方式将数据传送至从设备端。

4 数据传输例程项目结构

4.1 Project Target

在ble_rdts_peripheral工程中，为了方便用户使用DFU功能，如下图所示，我们创建了三个project target，用户可以在使用DFU配置或不使用DFU配置的情况下灵活切换。



- CMT453x: 一般蓝牙Target，不带DFU配置，一般ble项目只有这个target
- OTA_IMG_1：带蓝牙OTA的Target，配置为Bank1地址
- OTA_IMG_2：带蓝牙OTA的Target，配置为Bank2地址

4.2 目录结构

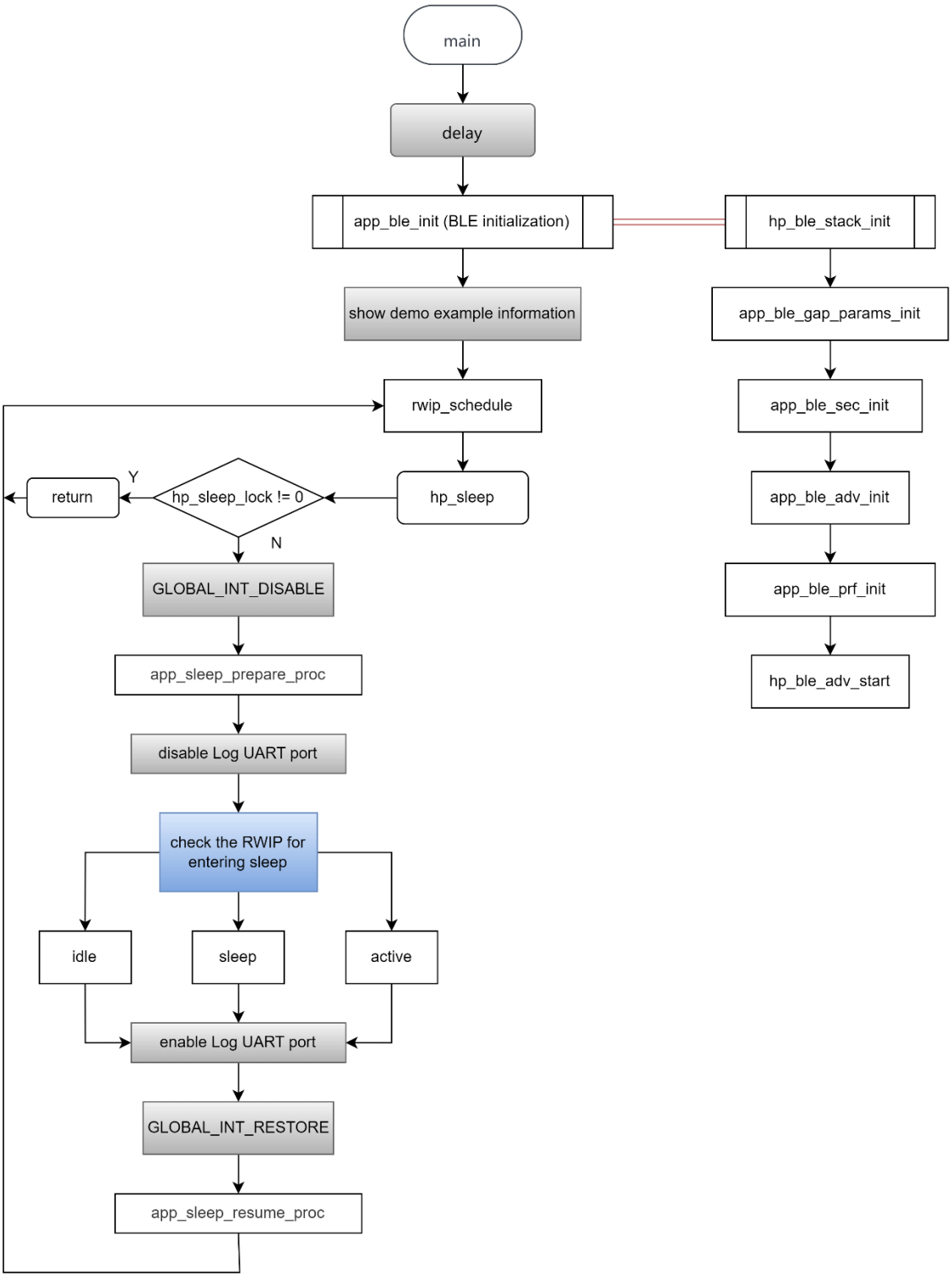
- STARTUP：芯片启动文件
- CMSIS：芯片内核配置
- FWLB：芯片外设驱动库
- BLE_STACK：BLE 协议栈
- BLE_PROFILE：BLE profile

- HP_DUF（可选）：蓝牙 OTA 固件升级相关库
- Crypto（可选）：蓝牙 OTA 固件升级使用的加密相关库
- HP_LIB：蓝牙应用相关库
- BLE_APP：蓝牙应用代码
  - app_ble.c 蓝牙应用代码，比如协议栈和广播初始化，开启广播，广播、连接和断开状态等消息回调。
  - app_dis.c 设备信息服务的应用代码
  - app_batt.c 电池电量信息服务的应用代码
  - app_rdtss.c 数传服务的应用代码
  - app_hp_ius.c OTA 空中升级服务的应用代码
- USER：用户应用代码
  - main.c 主函数，用户应用代码
  - app_usart.c 串口驱动和数据缓存应用代码
- CONFIG：配置文件
  - app_user_coonfig.h 用户配置
- DOC：说明文档
  - readme.txt

5 代码流程分析

5.1 数据传输从设备代码流程分析

数据传输从设备ble_rdts_peripheral的代码流程如下：



主要功能模块的描述如下：

delay:

数据传输从设备在上电之后，会默认delay 2s以防止设备上电或复位后立即进入低功耗模式，而无法连接上J-Link调试工具。

app_ble_init()

初始化蓝牙协议栈。

预设置蓝牙GAP相关的参数，包括设备蓝牙地址、角色等信息，连接间隔、超时时间等。

初始化蓝牙安全相关的功能，包括pin code，IO capabilities等。

初始化蓝牙广播数据，包括广播包内容，广播间隔等。

初始化蓝牙profile，例如，ble_rdts_peripheral中会初始化device information server profile和raw data transfer server profile。

创建广播事件并开始进行广播。

rwip_schedule()

协议栈所提供的接口，调度并处理所有未处理的事件。

hp_sleep()

进入sleep或idle状态

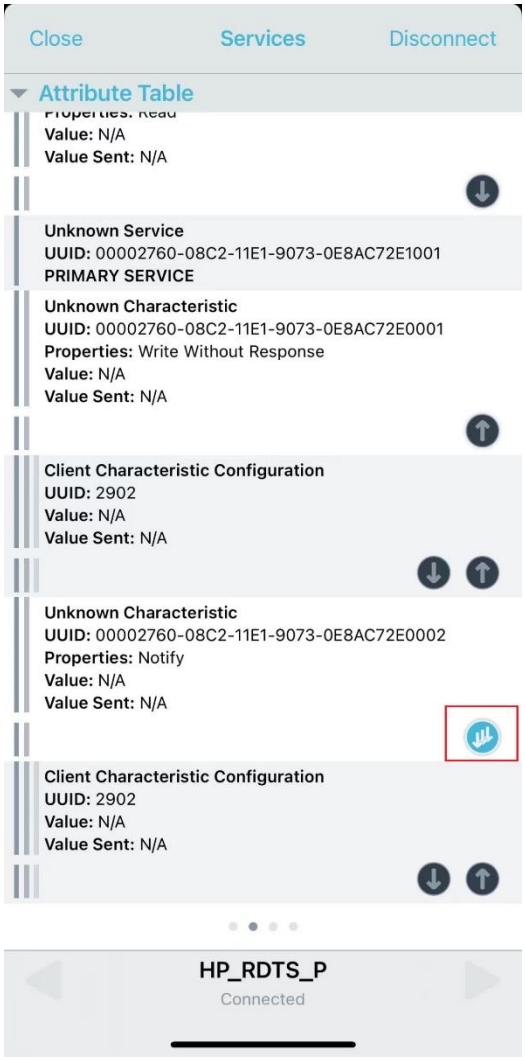
在profile初始化过程中，调用`hp_ble_add_prf_func_register()`将相应的profile添加到列表中，相应的message handler在系统初始化过程中也会注册至协议栈。

例如，在ble_rdts_peripheral例程中，raw data transfer sever (rdtss) profile中定义有以下三个message handler，分别用于client设备对server进行读操作时的处理，client设备完成对server设备写操作时的处理，notification发送完成后的相应处理。

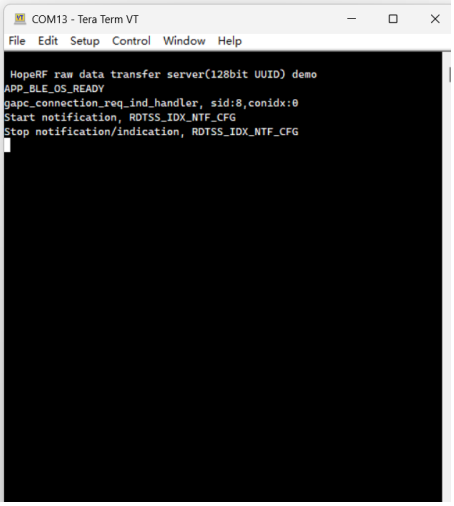
在发生相应操作时，协议栈将会自动触发message handler，用户程序可以在message handler中作相应处理。

```
const struct ke_msg_handler app_rdtss_msg_handler_list[] =
{
    {RDTSS_VALUE_REQ_IND,                (ke_msg_func_t)rdtss_value_req_ind_handler},
    {RDTSS_VAL_WRITE_IND,                (ke_msg_func_t)rdtss_val_write_ind_handler},
    {RDTSS_VAL_NTF_CFM,                  (ke_msg_func_t)rdtss_val_ntf_cfm_handler},
};
```

当通过手机App连接并打开UUID为0x00002760-08C2-11E1-9073-0E8AC72E0002特征值的通知权限时，协议栈将会触发app_rdtss.c 中的 `rdtss_val_write_ind_handler()`函数。用户可以在该message handler中通过判断UUID的handle来作相应的操作。

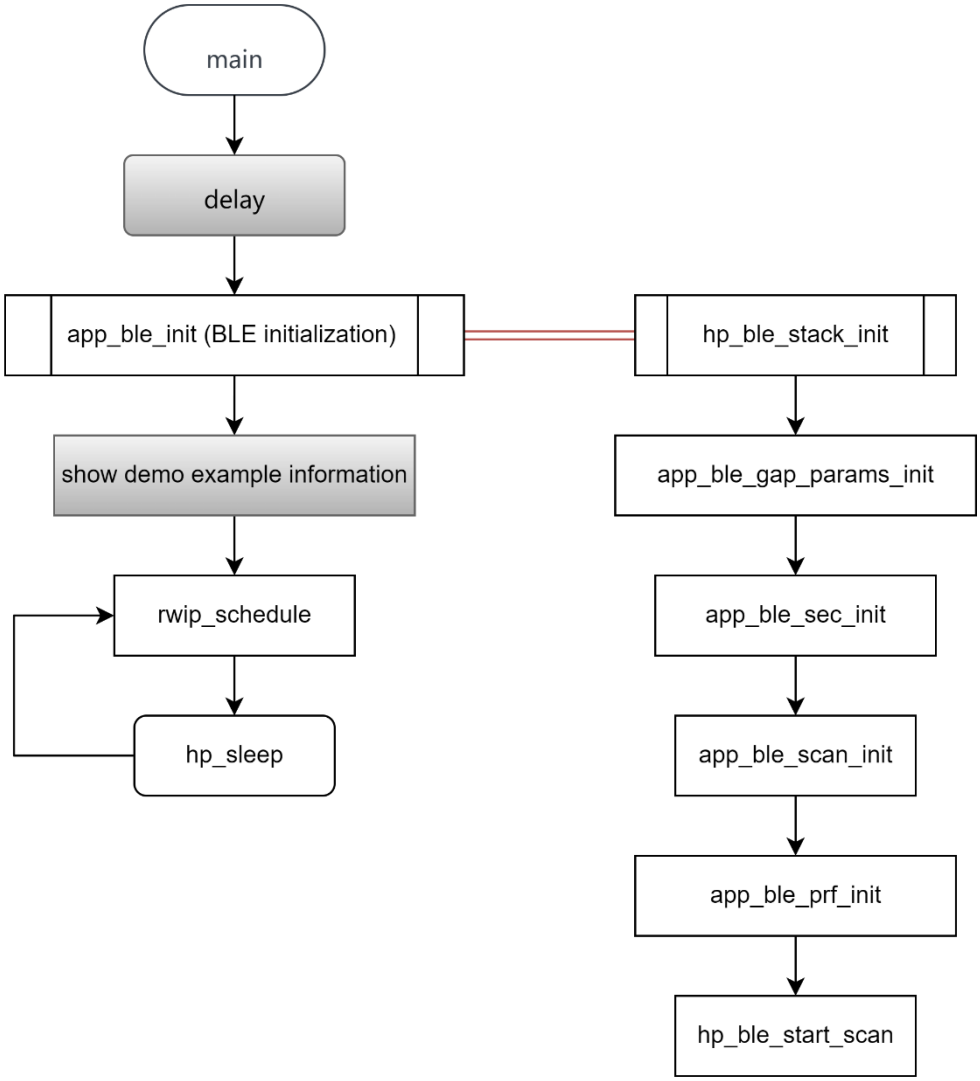



```
169  */
170 static int rdtss_val_write_ind_handler(ke_msg_id_t const msgid,
171                                          struct rdtss_val_write_ind const *ind_value,
172                                          ke_task_id_t const dest_id,
173                                          ke_task_id_t const src_id)
174 {
175     HP_LOG_DEBUG("%s,write handle = %x,length = %x\r\n",__func__ ,ind_value->handle, ind_value->length);
176
177     for(uint16_t i=0; i<ind_value->length; i++)
178     {
179         HP_LOG_DEBUG("%x ",ind_value->value[i]);
180     }
181     HP_LOG_DEBUG("\r\n");
182
183     uint16_t handle = ind_value->handle;
184     uint16_t length = ind_value->length;
185
186     switch (handle)
187     {
188         case RDTSS_IDX_NTF_CFG:
189
190             if(length == 2)
191             {
192                 uint16_t cfg_value = ind_value->value[0] + ind_value->value[1];
193
194                 if(cfg_value == PRF_CLI_START_NTF)
195                 {
196                     //enabled notify
197                   HP_LOG_INFO("Start notification, RDTSS_IDX_NTF_CFG\r\n");
198                 }
199                 else if(cfg_value == PRF_CLI_STOP_NTFIND)
200                 {
201                   HP_LOG_INFO("Stop notification/indication, RDTSS_IDX_NTF_CFG\r\n");
202                 }
203             }
204
205             break;
```



5.2数据传输主设备代码流程分析

数据传输主设备ble_rdts_central的代码流程如下：



主要功能模块与上一节中从设备的描述大致相同，我们仅对有差异的部分作进一步的阐述。

app_ble_init()

初始化蓝牙协议栈。

预设置蓝牙GAP相关的参数，包括设备蓝牙地址、角色等信息，连接间隔、超时时间等。

初始化蓝牙安全相关的功能，包括PIN code，IO capabilities等。

初始化蓝牙扫描相关参数，包括设备过滤条件，discover mode，扫描窗口和扫描间隔等。

初始化相关的蓝牙profile，例如，ble_rdts_central中会初始化raw data transfer client profile。

开始扫描。

主设备在扫描到指定名称的设备（例如 HP_RDTS_P）后，将自动使用预先设定的参数建立连接。连接建立之后，主从设备即可以通过蓝牙进行数据的传递。

版本历史

日期	版本	修改
2023.05.26	V1.0	初始版本