everyday genius

BT问题集锦

Version: 1.0
Release date: 2016-02-16

Specifications are subject to change without notice.

Document Revision History

Revision	Date	Author	Description
0.1	2016-01-28	Jian Xu	BT 问题集锦

Table of Contents

1 BT(BR/EDR) BLE Feature

1.1 BT(BR/EDR) MakeFile Feature Option

Feature Option	Value	Comments
BT_A2DP_PROFILE	TRUE/FALSE	是否支持蓝牙立体声音乐功能
BT_AUDIO_VIA_SCO	TRUE/FALSE	是否支持音乐通过蓝牙 SCO 传输功能
BT_AVRCP_PROFILE	TRUE/FALSE	是否支持蓝牙音频/视频远程控制功能
BT_DIALER_SUPPORT	TRUE/FALSE	是否支持蓝牙拨号器功能
BT_DISABLE_SSP_SUPPORT	TRUE/FALSE	是否取消蓝牙简易安全配对功能
BT_FM_RADIO_VIA_SCO	TRUE/FALSE	是否支持 FM 通过 SCO 传输功能
BT_HFG_PROFILE	TRUE/FALSE	是否支持蓝牙打电话 AG 端(手机)功能
BT_HF_PROFILE	TRUE/FALSE	是否支持蓝牙打电话 HF 端(耳机)功能
BT_HIDD_PROFILE	TRUE/FALSE	是否支持蓝牙鼠标/键盘功能
BT_MAPC_PROFILE	TRUE/FALSE	是否支持蓝牙短消息访问功能
BT_OPP_PROFILE	TRUE/FALSE	是否支持蓝牙文件传输功能
BT_PBAPC_PROFILE	TRUE/FALSE	是否支持蓝牙联系人访问功能
BT_PBAP_PROFILE	TRUE/FALSE	是否支持蓝牙联系人被访问功能
BT_SPEAKER_SUPPORT	TRUE/FALSE	是否支持蓝牙扬声器功能
BT_SPP_PROFILE	TRUE/FALSE	是否支持蓝牙串口功能
BT_SPP_CLIENT	BT_SPP_CLI_NO_SCO/NONE	是否支持蓝牙串口连接从端功能
BT_SPP_SERVER	BT_SPP_SRV_NO_SCO/NONE	是否支持蓝牙串口连接主端功能
BLUETOOTH_VERSION	BT_VER_40/BT_VER_30	支持蓝牙的版本

注:

MT6261 及之前平台 BLUETOOTH_VERSION 只支持 BT_VER_30, 只支持 BT, 不支持 BLE.

MT2502 平台 BLUETOOTH_VERSION 支持 BT_VER_40, 支持 BT, 支持 BLE, 蓝牙双模.

以上 MakeFile Feature Option, 如果 MediaTek 给到贵司的代码包中其是关闭的, 贵司不可以自己打开. 需要 MediaTek 做 Full Source Code Flavor Build 打开, 即申请 patch.

1.2 BLE MakeFile Feature Option

Feature Option	Value	Comments
BT_GATTC_SUPPORT	TRUE/FALSE	是否支持蓝牙 BLE 从连接功能
BT_GATTS_PROFILE	TRUE/FALSE	是否支持蓝牙 BLE 主连接功能
BT_MAX_LE_LINK_NUM	1/4	BLE 最多连接设备数目

注:

MT2502 平台 BLUETOOTH_VERSION 支持 BT_VER_40 可以使能.

1.3 MT6261 Phone/BT Dialer

MT6261 有 Phone 版本和 BT Dialer 版本.
Phone 版本是使用 MT6261 做功能机手机使用.
BT Dialer 版本是使用 MT6262 做蓝牙拨号器使用.
即可以像蓝牙耳机/车载连接手机后, 打电话/听歌曲/访问联系人, 客制化蓝牙串口传输等.
MT6261 及之前平台 BLUETOOTH_VERSION 只支持 BT_VER_30(蓝牙 3.0), 只支持 BT, 不支持 BLE.

1.4 MT2502 Aster V1 与 Aster V2 区分

MT2502 平台 BLUETOOTH_VERSION 支持 BT_VER_40, 支持 BT, 支持 BLE, 蓝牙双模.
MT2502 的代码包是以 11CW1418SP4 开头.
其有分 Aster V1 与 Aster V2 两个阶段, 称为 Aster V1 版本和 Aster V2 版本.
Aster V1 版本和 Aster V2 版本在 BT (BR/EDR)部分没有差异, BLE 部分有差异.
MT2502 Aster V1 版本上, 只支持 BLE single link (max 1 link).
MT2502 Aster V2 版本上, 支持 BLE multi-link (max 4 link).

区别方法:
在代码包名称中, V15 以下(不含 V15)版本为 MT2502 Aster V1 版本.
在代码包名称中, V15 及 V15 以上(含 V15)版本为 MT2502 Aster V2 版本.

1.5 MT2502 BLE only 版本

MT2502 BLE only 版本是将所有 BT (BR/EDR) profile 关闭.
即 makefile 中关闭 BT (BR/EDR) profiles.
BT_DUN_PROFILE=FALSE
BT_FAX_PROFILE=FALSE
BT_FTC_PROFILE=FALSE
BT_FTS_PROFILE=FALSE
BT_HFG_PROFILE=FALSE
BT_HIDD_PROFILE=FALSE
BT_MAPC_PROFILE=FALSE
BT_OPP_PROFILE=FALSE
BT_PBAP_PROFILE=FALSE
BT_PBAPC_PROFILE=FALSE
BT_SIM_PROFILE=FALSE
BT_SPP_PROFILE=FALSE
BT_AVRCP_PROFILE=FALSE
BT_A2DP_PROFILE=FALSE
BT_GOEP_PROFILE=FALSE
BT_HF_PROFILE=FALSE

BT_SPP_CLIENT=NONE;
BT_SPP_SERVER=NONE;
BT_SPEAKER_SUPPORT=FALSE;
同时申请两个 patch, patch id: MAUI_03525551 / MAUI_03532616.
由 MediaTek 来做 full source code build, 即 flavor build.

2 BT 常见问题

2.1 蓝牙相关 NVRAM

NVRAM_EF_BT_INFO_LID => 存放蓝牙名称/属性等信息
NVRAM_EF_BT_SYS_INFO_LID => 存放蓝牙系统相关信息
NVRAM_EF_BT_DEV_LIST_INDEX_LID => 存放蓝牙设备列表索引信息
NVRAM_EF_BT_DEV_LIST_LID => 存放蓝牙设备列表
NVRAM_EF_BTRADIO_MTK_BT_CHIP_LID => 存放蓝牙 BT(BR/EDR)地址/BT Chip 信息

2.2 MT6260/MT6261 蓝牙 common 问题

请参照以下 FAQ.

FAQ07211 [setting]如何实现开机自动打开蓝牙？
FAQ08021 [Setting]如何修改蓝牙默认名称？
FAQ05001 如何修改 MTK 蓝牙芯片的功率?
FAQ05687 [common]如何关闭蓝牙 EDR 功能
FAQ13002 如何在 11BW1308MP 上关闭 A2DP Source 和 AVRCP 的 TG.
FAQ12908 打开蓝牙 HID 服务，发送键值给 iphone 时，对方无响应。
FAQ07704 Feature phone profile QDID 已更新，请客户务必使用最新的 QDID
FAQ13310 过 BQB 时 PBAP client 的注意事项

2.3 HFP 相关问题

请参照以下 FAQ.

FAQ14763 [BT]BT 问题调试—HFP
FAQ13514 60/61 平台过 BQB HFP 的 TP/OCM/BV-01-I 这个 case 时的注意事项

2.4 SPP 相关问题

请参照以下 FAQ.

FAQ14760 [BT]BT 问题调试—SPP
FAQ10530 [SPP]如何在手机端实现 2 个 SPP server
FAQ10531 [SPP]2 个 SPP client 无法和手机中的 2 个 SPP SERVER 同时连接
FAQ05262 [SPP]如何修改 SPP uart owner
FAQ05263 [SPP]SPP 发送数据一分钟左右，无法发送数据

2.5 MT6261 Phone 版本蓝牙配对时需输入 pin 码, 要改为无需输入 pin 码, 点击确定即可方式

MT6261 Phone 版本默认在 BT(BR/EDR)配对时是以 pin 码方式.
如果需要"简易安全配对"功能, 需要申请 patch.

将主 makefile 里关于"BT_SSP_SUPPORT"和"BT_DISABLE_SSP_SUPPORT"两个宏做解释说明:
==========
BT_SSP_SUPPORT 这个宏控制开关是"简易安全配对"的开关.
关于这个开关, MediaTek 之前的主 makefile 设定是反的.
即 MediaTek 之前的主 makefile 设定"BT_SSP_SUPPORT"宏控制:
BT_SSP_SUPPORT = TRUE, Option.mak 里则打开"__BT_DISABLE_SSP_SUPPORT__", 关闭"简易安全配对"功能.
BT_SSP_SUPPORT = FALSE, Option.mak 里则关闭"__BT_DISABLE_SSP_SUPPORT__", 打开"简易安全配对"功能.

目前主 MediaTek 已经纠正, BT_SSP_SUPPORT 这个宏已经移除掉. 由新的宏控制.
即: BT_DISABLE_SSP_SUPPORT.
==========

--Patch >> MAUI_03510814
--Flavor >> BT_DISABLE_SSP_SUPPORT = FALSE

需要注意:
Patch Id: MAUI_03510814
Flavor: 替换 BT_SSP_SUPPORT 为 BT_DISABLE_SSP_SUPPORT,
并且设定 BT_DISABLE_SSP_SUPPORT = FALSE
以 Flavor 方式 Build.

2.6 MT2502 开机自动打开蓝牙

请检查贵司的 MT2502 版本, 是“TRACKER”版本? 还是“WATCH”版本? 还是“WT”版本(WATCH+TRACKER)?

“TRACKER”版本 => 代码包名称中包含“TRACKER”
“WATCH”版本 => 代码包名称中包含“WATCH”
“WT”版本(WATCH+TRACKER) => 代码包名称中包含“WT”

“TRACKER”版本:
\plutommi\Service\BtcmSrv\res\BtcmSrv.res
...省略
/* For smart tracker flight mode */

#if defined(__IOT__) && defined(__MMI_TELEPHONY_SUPPORT__)
<CACHEDATA type="byte" id="SRV_BT_IOT_POWER_STATUS" restore_flag="TRUE">
<DEFAULT_VALUE> [0X01] </DEFAULT_VALUE>
<DESCRIPTION> Bt power record </DESCRIPTION>
</CACHEDATA>
...省略
编译以: make new 方式.
烧录以: 烧录 bin 档之后, 记得 Format FAT.

“WATCH”版本:
“WT”版本(WATCH+TRACKER):
\plutommi\MtkApp\Connectivity\ConnectivityRes\Bluetooth\Bluetooth.res
...省略
<CACHEDATA type="byte" id="NVRAM_BT_POWER_STATUS" restore_flag="TRUE">
#if defined(__MMI_BTBOX_NOLCD__) || defined(__FORCE_DISABLE_NOTIFICATION_POPUP__)
<DEFAULT_VALUE> [0x01] </DEFAULT_VALUE>
#else
<DEFAULT_VALUE> [0x01] </DEFAULT_VALUE>
...省略
编译以: make new 方式.
烧录以: 烧录 bin 档之后, 记得 Format FAT.

2.7 MT2502 修改蓝牙名称

MT2502 蓝牙名称默认为"MTKBTDEVICE". 最多可写入 56 个字符.
MediaTek MT2502 蓝牙协议栈中 BT(BR/EDR)和 BLE 只有一个蓝牙名称, 没有办法做区分.
修改方法:
\custom\common\PLUTO_MMI\nvram_common_config.c
...省略
static kal_uint8 const NVRAM_EF_SRV_BT_CM_SYS_DEFAULT[] = {
0xf0, 0x00, 0x00, 0x00,
#ifdef __MMI_BTBOX_NOLCD__
'B', 'T', 'B', 'O',
'X', 'N', 'o', 'L',
'C', 'D', 0x00, 0x00,
#else /* __MMI_BTBOX_NOLCD__ */
'M', 'T', 'K', 'B',
'T', 'D', 'E', 'V',
'I', 'C', 'E', 0x00,
#endif /* __MMI_BTBOX_NOLCD__ */
0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00
};
...省略

2.8 MT2502 蓝牙可见性

MT2502 是蓝牙双模, BT(BR/EDR)和 BLE, BT(BR/EDR)有可见性. BLE 没有可见性, BLE 是广播.

NVRAM_EF_SRV_BT_CM_SYS_DEFAULT => 这个 NV 项中的 0xf0 代表开启可见性.

#define SRV_BT_CM_MASK_VIS 0x00000020

算法依据: 0xf0, 0x00, 0x00, 0x00 其实是: 0x000000f0, 其包含 0x00000020, 所以其可见性是开启的.

\custom\common\PLUTO_MMI\nvram_common_config.c

...省略
static kal_uint8 const NVRAM_EF_SRV_BT_CM_SYS_DEFAULT[] = {
0xf0, 0x00, 0x00, 0x00,
...省略

NVRAM_EF_SRV_BT_CM_SYS_DEFAULT => 中的 0xf0 代表开启可见性.
NVRAM_EF_SRV_BT_CM_SYS_DEFAULT => 中的 0xd0 代表关闭可见性.

2.9 MT2502 关闭 BLE 广播

MT2502 有 BT(BR/EDR)和 BLE 双模式.
可见性只是 for BT(BR/EDR)的模式.
BLE 模式下没有关闭可见性这种说法.
在 MT2502 Aster V1 版本上, 只支持 BLE single link(max 1 link), 在 MT2502 Aster V2 版本上支持 multilink(max 4 link).
MT2502 Aster V1 版本上, 在 BLE 连接后, 会停止发送 Advertising, 不再被搜索到.
MT2502 Aster V2 版本上, 在 BLE 连接后, 会继续发送 Advertising, 可以再被搜索到, 与其他 BLE 设备连接, 最多连接 4 个 BLE 设备. 当 4 个 BLE 设备都连上之后, 会停止发送 Advertising, 不再被搜索到.

MediaTek BT Stack 在处理 BLE 的广播时, 在 BT Stack 是只有在 ACL 断连...等场景会先将 BLE 广播停掉, 在 BT Stack 操作完毕之后, 再恢复到之前的 BLE 广播状态.
BT Stack 打开广播和关闭广播, 是交由 MMI 来控制.

BLE 打开 advertising:
如果有 BLE 的 profile 已经调用 srv_gatts_listen() 接口(入参为 TRUE)打开 advertising, 则保持打开 advertising 状态. 如果没有 BLE 的 profile 调用 srv_gatts_listen(), 则也是打开 advertising 状态.

BLE 关闭 advertising:
所有注册的 BLE profile 都调用 srv_gatts_listen() 接口(入参为 FALSE)后, 才能真正关闭 advertising.

2.10 MT2502 如何通过修改 NVRAM 开机自动开蓝牙

“TRACKER”版本 => 代码包名称中包含“TRACKER”
“WATCH”版本 => 代码包名称中包含“WATCH”
“WT”版本(WATCH+TRACKER) => 代码包名称中包含“WT”

“TRACKER”版本:
连接 META 工具, 选择 "NVRAM Editor" -> load database 文件后,
点击 "Other LID" -> "NVRAM_EF_CACHE_BYTE_LID" -> "Cachebyte[13]"
SRV_BT_IOT_POWER_STATUS_byte0:8 从 "0x00" 改为 "0x01"

“WATCH”版本/“WT”版本(WATCH+TRACKER):
连接 META 工具, 选择 "NVRAM Editor" -> load database 文件后,
点击 "Other LID" -> "NVRAM_EF_CACHE_BYTE_LID" -> "Cachebyte[8]"
NVRAM_BT_POWER_STATUS_byte0:8 从 "0x00" 改为 "0x01"

2.11 MT2502 在代码中自行写蓝牙名称

1, 发送 MSG_ID BT_BM_WRITE_LOCAL_NAME_REQ 到 BT stack 来更改蓝牙名称.
可以参照 BtcmSrvPrmt.c, srv_bt_cm_set_local_name_req_hdler()函数.

2, 通过 NVRAM 方式写入蓝牙名称.

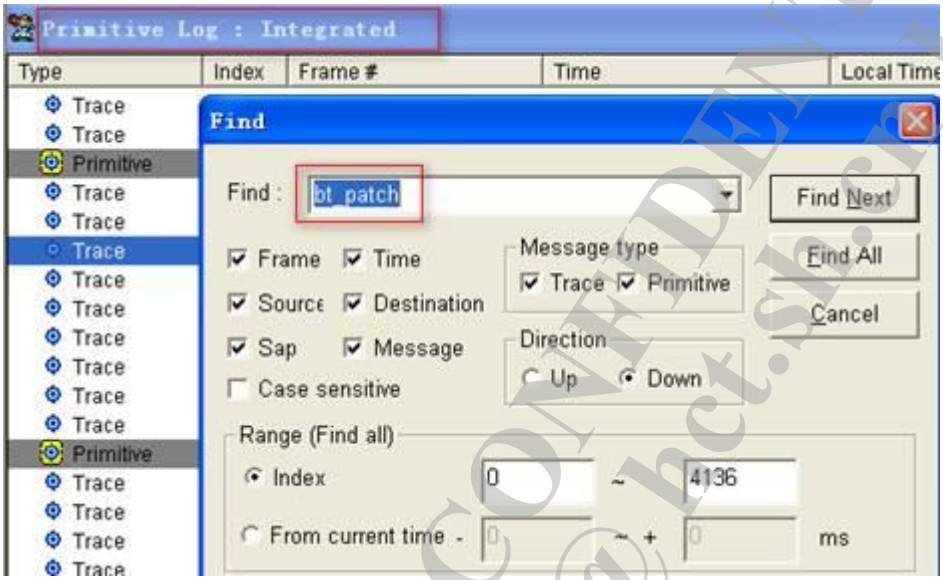
```
#define BT_NAME_LEN 56

srv_bt_cm_sys_nvram_struct data_struct;
char bt_name[BT_NAME_LEN] = {0};

strcpy(bt_name, "BT_NAME_TBD");
memcpy(data_struct.host_dev_name, bt_name, BT_NAME_LEN);
```

WriteRecordSlim(NVRAM_EF_BT_SYS_INFO_LID,1,&data_struct,NVRAM_EF_BT_SYS_INFO_SIZE);

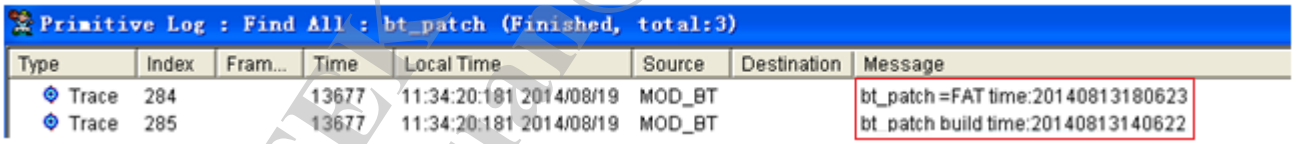
2.12 MT2502 检查蓝牙固件信息

1, MT2502 开机, 然后连接 Catcher 工具, 设置 Catcher filter “MOD_BT” all class on.

2, 操作打开蓝牙, 在 Catcher 中搜索关键字 “bt_patch” 在 “Primitive Log: Integrated” 中.



3, 在搜索到的信息中看到蓝牙固件的 FAT time 和 build time.

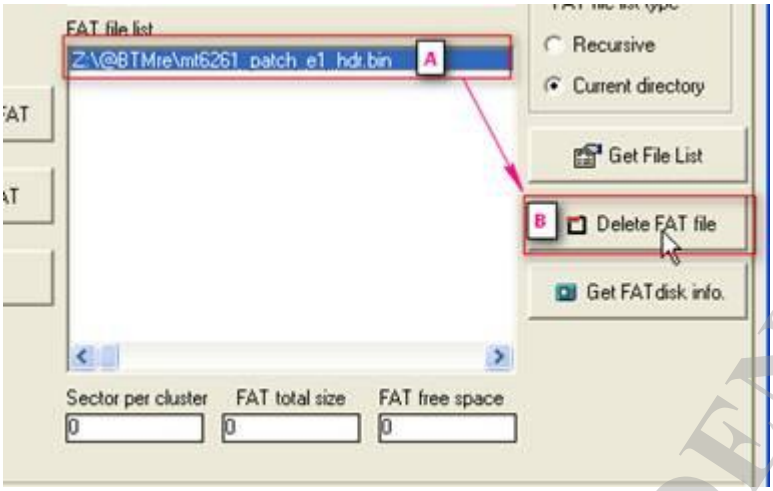


2.13 MT2502 更新蓝牙固件方法

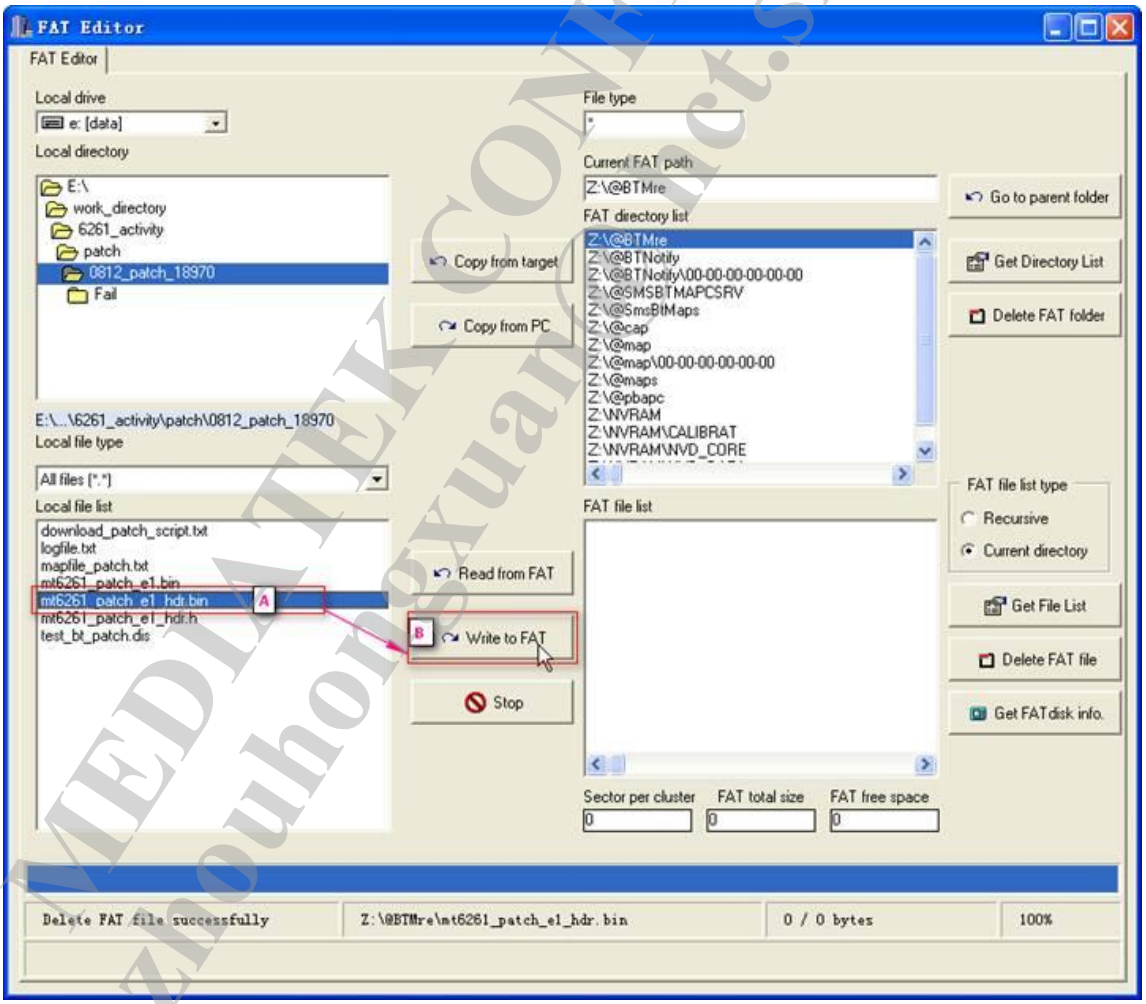
[Update BT Firmware]

通过 META 工具连接 MT2502, 在 “FAT Editor” 中操作.

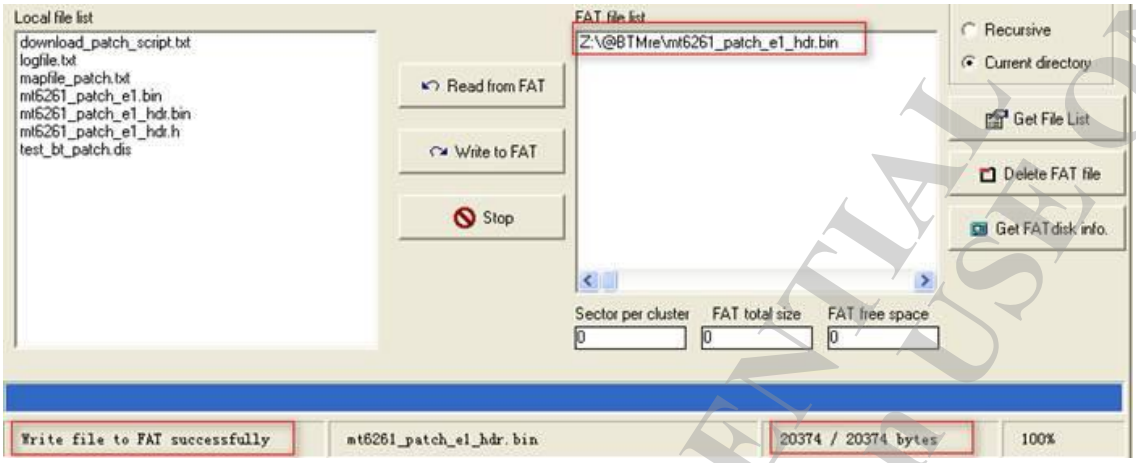
1, 删除原有的蓝牙固件文件 (must be)



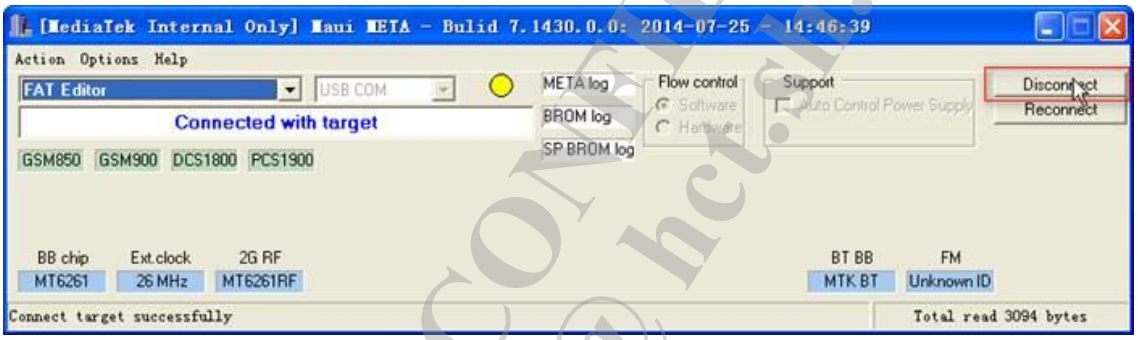
2, 选择要更新的蓝牙固件文件 (命名为: mt6261_patch_e1_hdr.bin), 在左侧选择要更新蓝牙固件路径, 选择文件, 然后点击 “Write to FAT”.



在点击 “Write to FAT” 之后, 可以在右侧 (即 target MT2502 端) 看到写入的蓝牙固件文件, 然后在右下角检查写入的蓝牙固件文件大小.



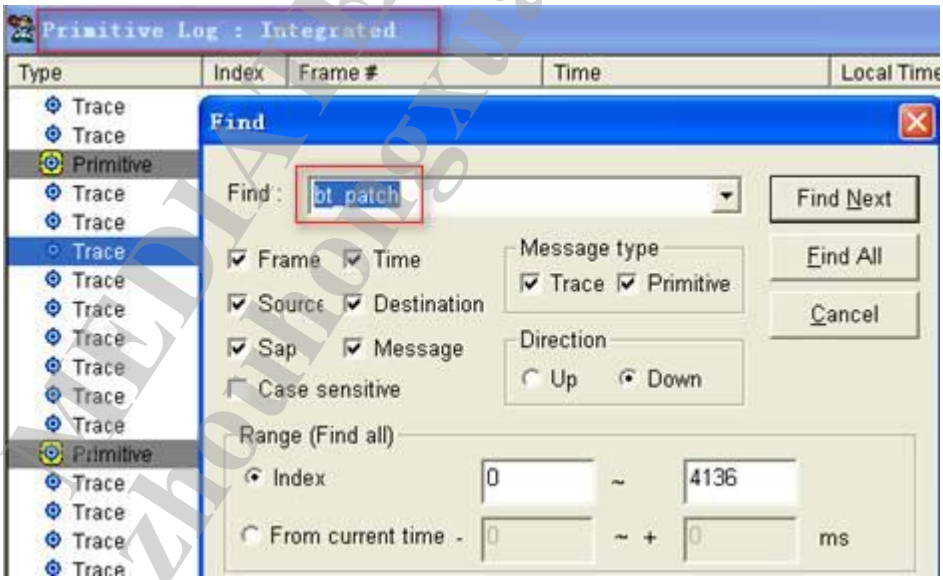
在更新蓝牙固件成功之后, 请关闭 “FAT Editor” 窗口, 并且点击 “Disconnect”, 离开 META 工具.



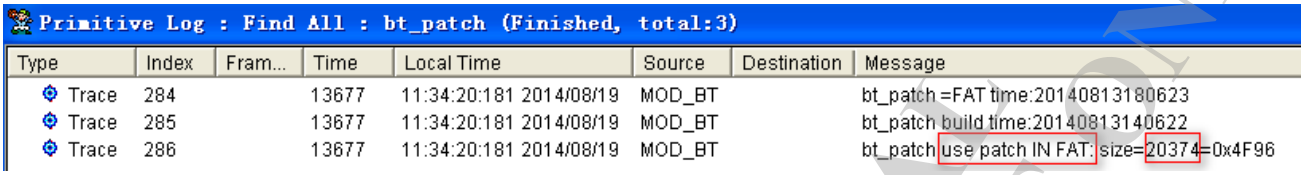
[更新蓝牙固件之后, 检查蓝牙固件是否被正确使用]

1, MT2502 开机, 然后连接 Catcher 工具, 设置 Catcher filter “**MOD_BT**” **all class on**.

2, 操作打开蓝牙, 在 Catcher 中搜索关键字 “bt_patch” 在 “Primitive Log: Integrated” 中.



3, 在搜索到的信息中看到 “use patch IN FAT”, 说明蓝牙固件有被成功是能, 另外还需要再检查下 size, 是否和写入的蓝牙固件 size 相同.



2.14 MT2502 抓 BT Log 说明

1, MT2502 蓝牙问题在厘清时, 通常需要抓 BT 相关 Catcher log. 下面说下 MT2502 在抓 BT 相关 log 时需要注意的事项.

==========

MOD_BT (all class on) => BT stack 相关 log

MOD_MMI_CONN_APP (trace_group_7) => MMI 层 BT 相关 log

MOD_BRT (all class on) => BT stack 和 BT controller 之间相关 log

==========

2, 由于 MT2502 平台有分 MT2502D/MT2502A/MT2502C, 可以使用的 ROM size 有差, 所以在有的版本工程中会将打印 log 的模块关闭 (即主 makefile 中 KAL_TRACE_OUTPUT=NONE), 来节省出 ROM size.

或者 MediaTek 给到贵司的版本工程中只有主 makefile, 并且 KAL_TRACE_OUTPUT=NONE, 没有备选 makefile(KAL_TRACE_OUTPUT=FULL).

这样会导致在抓 log 时, 只能抓到 msg 消息, 而没有其他 log 送出的现象. 请贵司参照如下:

==========

如果贵司的工程中主 makefile 有备选 makefile(KAL_TRACE_OUTPUT=FULL), 请将 "KAL_TRACE_OUTPUT" / "KAL_DEBUG_LEVELL" / "FLAVOR" 从备选 makefile(KAL_TRACE_OUTPUT=FULL)合入设置(其他多余的 feature option 不要合哦!!!), 然后 build(make new)出 load 烧入到 MT2502 内抓 Catcher log.

如果贵司的工程中主 makefile 没有备选 makefile(KAL_TRACE_OUTPUT=FULL), 并且主 makefile 中 KAL_TRACE_OUTPUT=NONE. 再或者贵司如果将主 makefile 中 KAL_TRACE_OUTPUT 定义为 FULL 后, 编译时出现超出 ROM size 情况, 请修改为 KAL_TRACE_OUTPUT=CUST_PARTIAL, 同时修改 PARTIAL_TRACE_LIB=custom drv peripheral(贵司需要 debug 打印信息的模块...). 目前我们需要 debug "MOD_BT", 请定义 PARTIAL_TRACE_LIB = btadp btdrv btprofiles btstack, 然后 build(make new)出 load 烧入到 MT2502 内抓 Catcher log.

==========

这样之后, 就可以在抓到的 Catcher log 中看到不仅有 "MOD_BT" msg 还有其他相关 log.

3, MT2502 在操作蓝牙场景下有各种电流问题, 同时需要抓 sleep log 厘清问题, 请贵司参照如下:

==========

抓取 sleep log 时, 一定要用 UART1 抓取. 在工模中设定 TST-PS Config & TST-L1 Config -> UART1 -> 921600 (460800) (设置为最高波特率).

(1), 抓取整个过程的 Catcher log.

(2), 同时记录关键时间点, 电流正常/电流异常/操作行为…时间点.
(3), 复现电流大问题后, 量测的时间久些. 不少于 15 分钟. 同时注意电流数据变化.
(4), 抓取 log 时, Catcher filter: MOD_BT(all class on), 同时选择 Default Filter -> Others -> Modem_Sleep_Mode(勾选). (如果开机后, 没有内容, 则再次设置一次这个 filter).
查看 "Catcher 菜单 -> Advanced -> SleepMode(勾选)", Sleep Mode Lock 及 Catcher log 里的内容 (此界面 3 个窗口都有内容, 如果没有内容, 再重新设置一次 filter).
记得 Target 端(MT2502)的 Trace filter 有打开, 如果第一次设定完 filter 之后, 不要马上抓取复现问题的 Catcher log, 请重启 MT2502, 再开始录制复现整个问题 log.
记得按照第二步骤中的操作, 设置好主 makefile, 以确保抓到 MOD_BT(all class on)和 sleep 的 log.
==========

注意: 不论抓什么 log 一定一定要避免丢 trace!!!

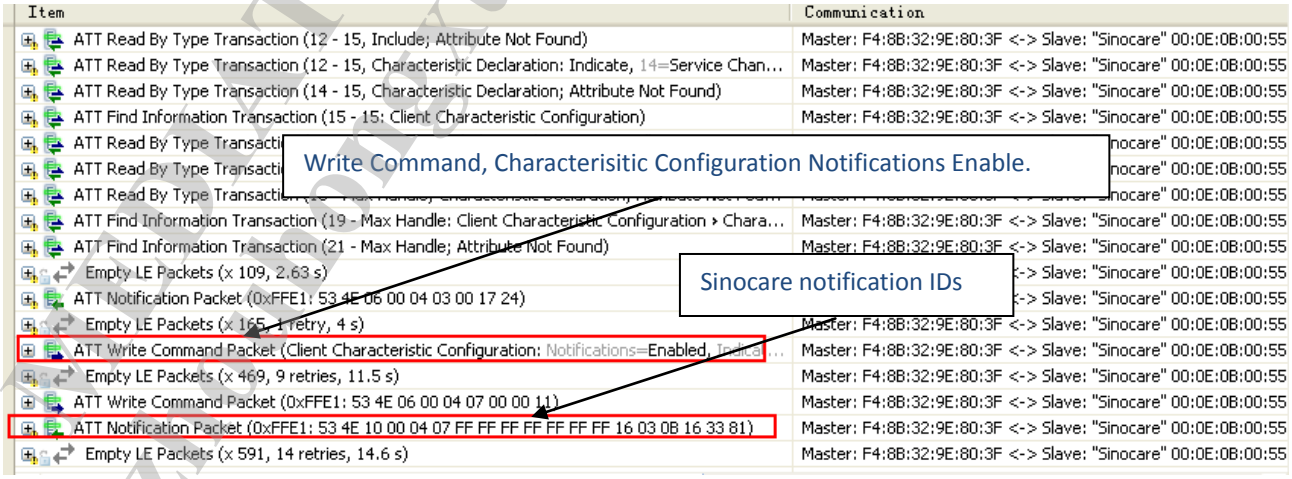
2.15 MT2502 BLE 主动连接其他 BLE 设备后 MT2502 操作这个 BLE 设备时失败

描述问题:

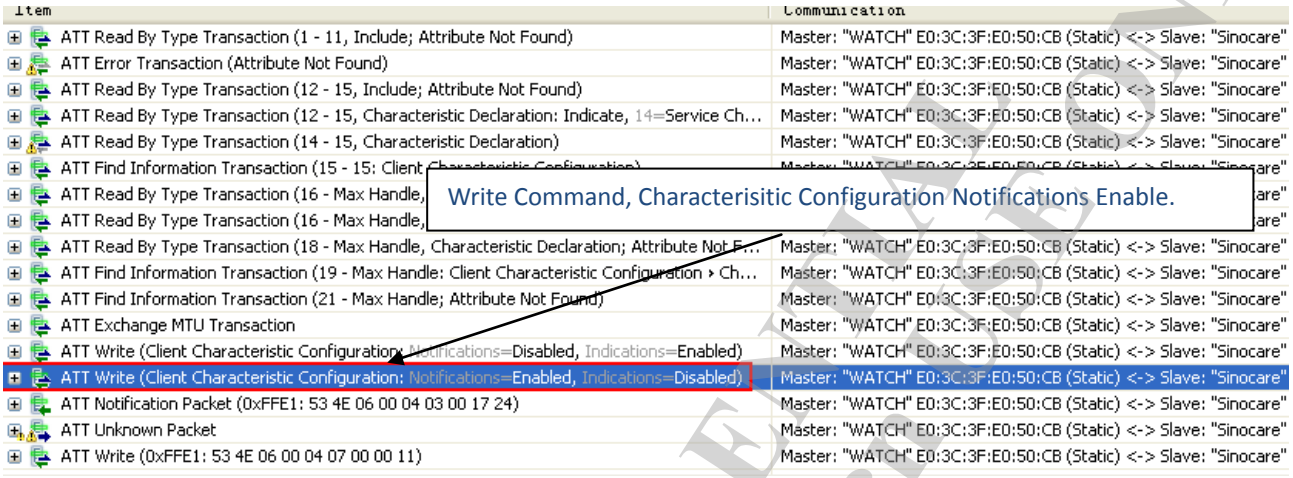
MT2502 BLE 主动连接其他 BLE 设备后 MT2502 操作这个 BLE 设备时失败.
由于客户的不同产品需求, 客户会客制化使用 MT2502 BLE 主动连接其他 BLE 设备, 进行操控, 例如读出测试数据, 控制开关…
本案例中 BLE 设备为一款 Sinocare 血糖仪, MT2502 和这款 Sinocare 血糖仪 BLE 连接成功后, Sinocare 血糖仪 ID 读不到.
使用 Smart Phone 安装 APK, 通过 BLE 连接这款 Sinocare 血糖仪后, Sinocare 血糖仪 ID 可以读到.

分析问题:

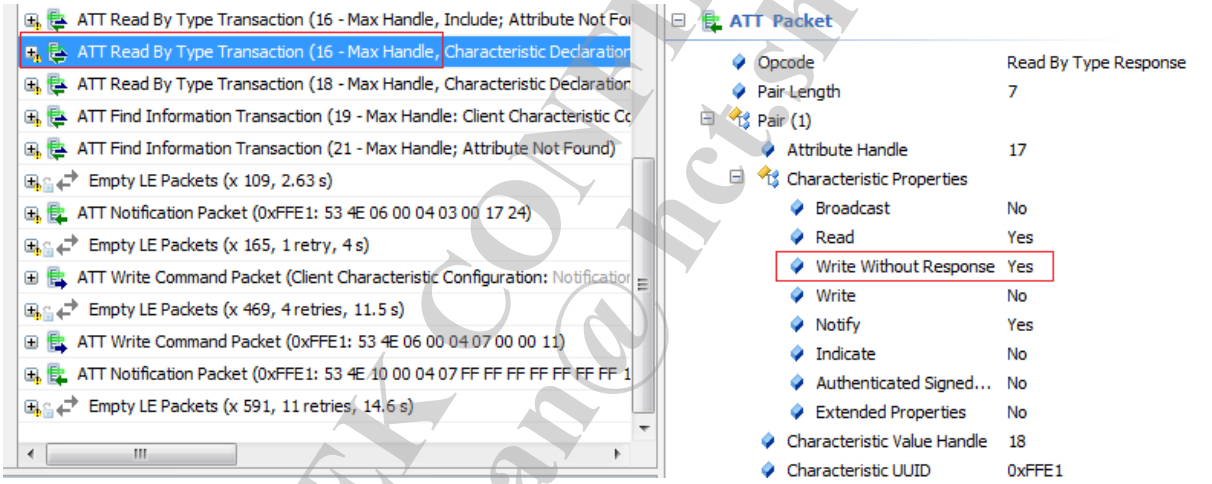
Smart Phone 的 Air Sniffer Log:



MT2502 的 Air Sniffer Log:



通过空中包看到, 这个 BLE 设备是 “Write Without Response” 属性.



通过检查客户客制化代码, 看到在调用以下 API 时, 使用属性如下:

srv_gattc_write_descriptor(conn, descr, &value, GATTC_WRITE_TYPE_REQUEST, GATTC_AUTH_REQ_NONE);

srv_gattc_write_descriptor(&pBeltercCtx->conn, &desc_info, &value, GATTC_WRITE_TYPE_REQUEST, GATTC_AUTH_REQ_NONE);

srv_gattc_write_characteristic(&pBeltercCtx->conn, &char_info, &value, GATTC_WRITE_TYPE_REQUEST, GATTC_AUTH_REQ_NONE);

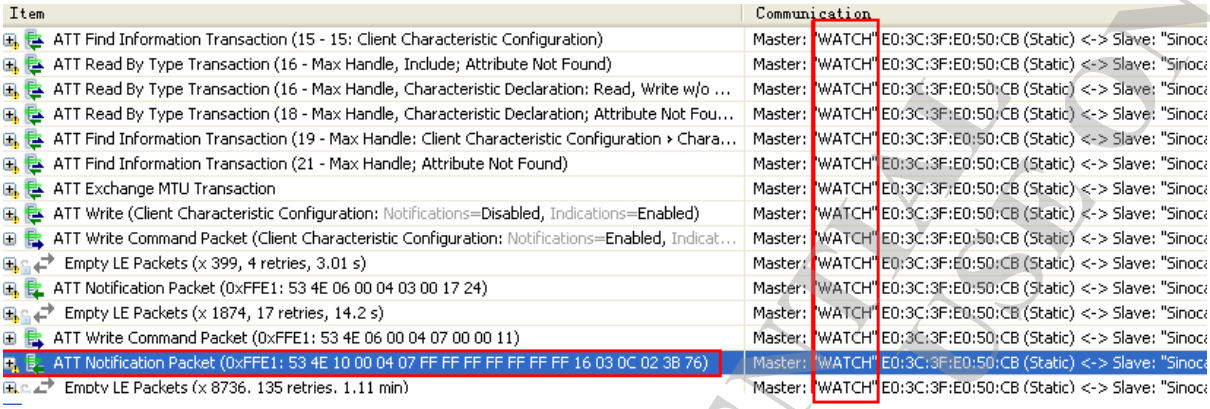
修改调用以下 API 的使用属性:

srv_gattc_write_descriptor(conn, descr, &value, GATTC_WRITE_TYPE_NO_RSP, GATTC_AUTH_REQ_NONE);

srv_gattc_write_descriptor(&pBeltercCtx->conn, &desc_info, &value, GATTC_WRITE_TYPE_NO_RSP, GATTC_AUTH_REQ_NONE);

srv_gattc_write_characteristic(&pBeltercCtx->conn, &char_info, &value, GATTC_WRITE_TYPE_NO_RSP, GATTC_AUTH_REQ_NONE);

可以成功读取出 Sinocare 血糖仪 ID.



解决方法:

API: srv_gattc_write_descriptor()/srv_gattc_write_characteristic(), 参数从"GATTC_WRITE_TYPE_REQUEST"改为"GATTC_WRITE_TYPE_NO_RSP".

可以先通过 read 的方式, 通过"MSG_ID_BT_GATTC_GET_CHAR_CNF"的属性 properties 值判断以什么样的方式来写数据.

GattSrv.h

#define GATT_CHAR_PROP_READ 0x02

#define GATT_CHAR_PROP_WRITE _WO_RESPONSE 0x04

#define GATT_CHAR_PROP_WRITE 0x08

2.16 MT2502 蓝牙地址

MT2502 设计是如果存放 BT(BR/EDR)/BLE 地址的区域为空, 则随机生成 BT(BR/EDR)/BLE 地址.

SN Write Tool 写入的蓝牙地址是 BT(BR/EDR)的地址, 其在 MT2502 内, 有单独的 NVRAM 项存放.

BLE 地址存放在 FAT 区域的 "Z:\@bt\le" 文件中, 没有 NVRAM 项存放, BLE 的地址有在代码中开出接口供客户自己写入. (默认代码中是不包含写入 BLE 地址的接口, 需要申请 patch).

2.17 MT2502 写 BT(BR/EDR)地址

MT2502 没有现有的 function 写入 BT(BR/EDR)地址. 在代码中写 BT(BR/EDR)地址请参照如下.

Example:

```
==========================================================
    btbm_bd_addr_t bd_addr;//写入的 bt(br/edr) address
    kal_bool result = KAL_FALSE;
    nvram_ef_btradio_mtk_bt_chip_struct read_struct;
    srv_bt_cm_nvram_struct read_struct2;

    if(KAL_TRUE == nvram_external_read_data(NVRAM_EF_BTRADIO_MTK_BT_CHIP_LID, 1, (kal_uint8*)&read_struct, NVRAM_EF_BTRADIO_MTK_BT_CHIP_SIZE))
```

```
    {
        read_struct.BDAddr[0] = (bd_addr.lap & 0xFF);
        read_struct.BDAddr[1] = ((bd_addr.lap >> 8) & 0xFF);
        read_struct.BDAddr[2] = ((bd_addr.lap >> 16) & 0xFF);
        read_struct.BDAddr[3] = bd_addr.uap;
        read_struct.BDAddr[4] = ((bd_addr.nap & 0xFF));
        read_struct.BDAddr[5] = ((bd_addr.nap >> 8) & 0xFF);

        result =
nvram_external_write_data(NVRAM_EF_BTRADIO_MTK_BT_CHIP_LID,1,(kal_uint8*)&read_struct,
NVRAM_EF_BTRADIO_MTK_BT_CHIP_SIZE);

        if(result && KAL_TRUE == nvram_external_read_data(NVRAM_EF_BT_INFO_LID, 1,
(kal_uint8*)&read_struct2, NVRAM_EF_BT_INFO_SIZE))
        {
            read_struct2.host_dev.bd_addr.lap = bd_addr.lap;
            read_struct2.host_dev.bd_addr.uap = bd_addr.uap;
            read_struct2.host_dev.bd_addr.nap = bd_addr.nap;

            result =
nvram_external_write_data(NVRAM_EF_BT_INFO_LID,1,(kal_uint8*)&read_struct2,NVRAM_EF_BT
_INFO_SIZE);
        }
    }
============================================================
```

2.18 MT2502 写 BLE 地址

MT2502 BLE 地址存放在 FAT 区域的 "Z:\@bt\le" 文件中, 没有 NVRAM 项存放, BLE 的地址有在代码中开出接口供客户自己写入.
默认代码中是不包含写入 BLE 地址的接口, 需要申请 patch. 请申请 patch. Patch ID: MAUI_03539779.
申请 patch 之后, 参照下面的方法进行 ble 地址的写入.

```
============================================================
#include "bluetooth_gap_struct.h"

bt_bm_change_le_address_req_sturct *res_msg;

res_msg = (bt_bm_change_le_address_req_struct*)
OslConstructDataPtr(sizeof(bt_bm_change_le_address_req_struct));

res_msg->addr[0] = 0x12;
res_msg->addr[1] = 0x34;
res_msg->addr[2] = 0x56;
res_msg->addr[3] = 0x78;
res_msg->addr[4] = 0x9A;
```

```
res_msg->addr[5] = 0xBC;

mmi_frm_send_ilm(MOD_BT,                    MSG_ID_BT_BM_CHANGE_LE_ADDRESS_REQ,
(oslParaType*)res_msg, NULL);
```

==========

调用时间点:

没有限制, User 可在任意时间改写 BLE address, 建议在 mmi init 或是开启蓝牙之前调用, 最好不要在蓝牙通信过程中调用 (行为不预期, 有可能会断线).

2.19 MT2502 AT Command 实现 BT TX Power Test

通过 AT Command 实现配置 TX Pattern/Channel(signal frequency)/Packet type,
进行 BT TX Power Test.
Example:

==========

1. ata_external.h

```
extern custom_rsp_type_enum ata_led_hdlr_v2(custom_cmdLine * commandBuffer_p);
extern custom_rsp_type_enum TxRxTest(custom_cmdLine * commandBuffer_p);//修改
extern custom_rsp_type_enum TxRxTestExit(custom_cmdLine * commandBuffer_p);//修改
```

2. customer_at_command.c

```
                                        {"AT+ECUSLCD",ata_customer_lcd_hdlr},
        //LCD
                                        {"AT+ELEDV2",ata_led_hdlr_v2},
    //LED
                                        {"AT+TxRxTest",TxRxTest},//修改
                                        {"AT+TxRxTestExit",TxRxTestExit},//修改
```

3. ata_at_command_customer_hdlr.c

```
#include "bluetooth_struct.h"//修改
#include "mmi_frm_queue_gprot.h"//修改

custom_rsp_type_enum ata_led_hdlr_v2(custom_cmdLine * commandBuffer_p)
{

    char buffer[MAX_UART_LEN+1];

    kal_uint32 para;
```

```
        para = ata_get_at_para(commandBuffer_p);


        sprintf (buffer, "\r\n+ELEDV2:OK\r\n");
        rmmi_write_to_uart((kal_uint8*)buffer, strlen(buffer), KAL_TRUE);
        return CUSTOM_RSP_OK;

}

//修改开始
extern void BtRadio_EMEnabledTxRxTest(U8 pattern,U8 channel_hopping,U8 tx_freq,U8 rx_freq,U8
poll_period, U8 packet_type,U16 packet_length);
extern void BtRadio_EMEnabledTxRxTestExit(void);

custom_rsp_type_enum TxRxTest(custom_cmdLine * commandBuffer_p)
{
        char buffer[MAX_UART_LEN+1];

        kal_uint32 para;

        //bt_engineer_mode_txrx_test_req_struct *tx_power_struct;

        para = ata_get_at_para(commandBuffer_p);

        kal_prompt_trace(MOD_BT, "[TxRxTest]+");

        //tx_power_struct = OslConstructDataPtr(sizeof(bt_engineer_mode_txrx_test_req_struct));
        //tx_power_struct->ref_count = 1;
        //tx_power_struct->pattern = 0x03;
        //tx_power_struct->packet_type = 0x04;
        //tx_power_struct->packet_length = 27;
        //tx_power_struct->tx_freq = 39;
        //tx_power_struct->poll_period = 0;
        //tx_power_struct->channel_hopping = 0;
        //tx_power_struct->rx_freq = 0;

        //mmi_frm_send_ilm((oslModuleType)MOD_BT,
(oslMsgType)MSG_ID_BT_ENGINEER_MODE_TXRX_TEST_REQ, (oslParaType *)tx_power_struct,
NULL);

//TX Pattern/Channel(signal frequency)/Packet type 等信息根据 BT Spec 可以在此进行设定调整

        BtRadio_EMEnabledTxRxTest(
                0x03, //((bt_engineer_mode_txrx_test_req_struct*)(bt_p->bt_rftest_buffer))->pattern,
                0,              //((bt_engineer_mode_txrx_test_req_struct*)(bt_p->bt_rftest_buffer))-
>channel_hopping,
```

```
                39, ///((bt_engineer_mode_txrx_test_req_struct*)(bt_p->bt_rftest_buffer))->tx_freq,
                0, ///((bt_engineer_mode_txrx_test_req_struct*)(bt_p->bt_rftest_buffer))->rx_freq,
                0, ///((bt_engineer_mode_txrx_test_req_struct*)(bt_p->bt_rftest_buffer))->poll_period,
                0x04,           ///((bt_engineer_mode_txrx_test_req_struct*)(bt_p->bt_rftest_buffer))-
>packet_type,
                27);            ///((bt_engineer_mode_txrx_test_req_struct*)(bt_p->bt_rftest_buffer))-
>packet_length);


        kal_prompt_trace(MOD_BT, "[TxRxTest]-");

        sprintf (buffer, "\r\n+TxRxTest:OK\r\n");
        rmmi_write_to_uart((kal_uint8*)buffer, strlen(buffer), KAL_TRUE);
        return CUSTOM_RSP_OK;

}



custom_rsp_type_enum TxRxTestExit(custom_cmdLine * commandBuffer_p)
{
        char buffer[MAX_UART_LEN+1];

        kal_uint32 para;

        //bt_engineer_mode_txrx_test_req_struct *tx_power_struct;

        para = ata_get_at_para(commandBuffer_p);

        kal_prompt_trace(MOD_BT, "[TxRxTestExit]+");

        BtRadio_EMEnabledTxRxTestExit();

        kal_prompt_trace(MOD_BT, "[TxRxTestExit]-");

        sprintf (buffer, "\r\n+TxRxTestExit:OK\r\n");
        rmmi_write_to_uart((kal_uint8*)buffer, strlen(buffer), KAL_TRUE);
        return CUSTOM_RSP_OK;

}
```

//修改结束

修改后, 编译出 load 下载到 MT2502 上后, 先发送 AT+EBTAT=bt_power_on, 在发送 AT+TxRxTest, TX Power 可以量测到, 发送 AT+TxRxTestExit 退出, AT+EBTAT=bt_power_off 关闭蓝牙.

(和贵司说明: AT+EBTAT=bt_power_on 后, 发送 AT+TxRxTest, 即调用 BtRadio_EMEnabledTxRxTest() 函数不要太快, 否则导致死机, 建议起 timer 延后 10S 做.)

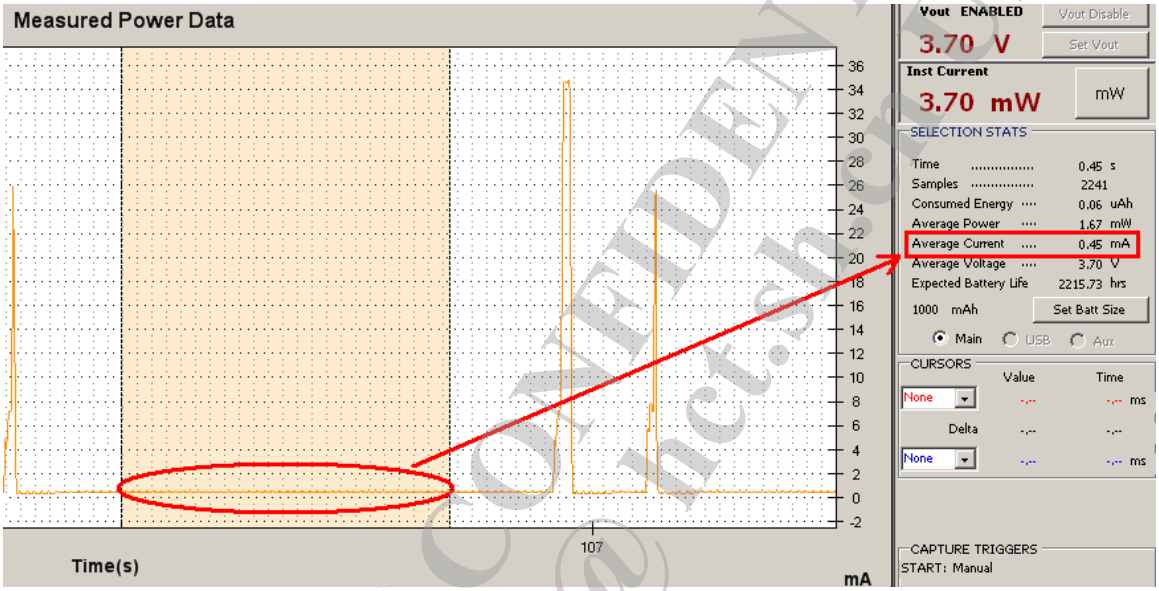
==========

3 BT 电流数据及问题

3.1 MT2502 蓝牙双模版本电流数据

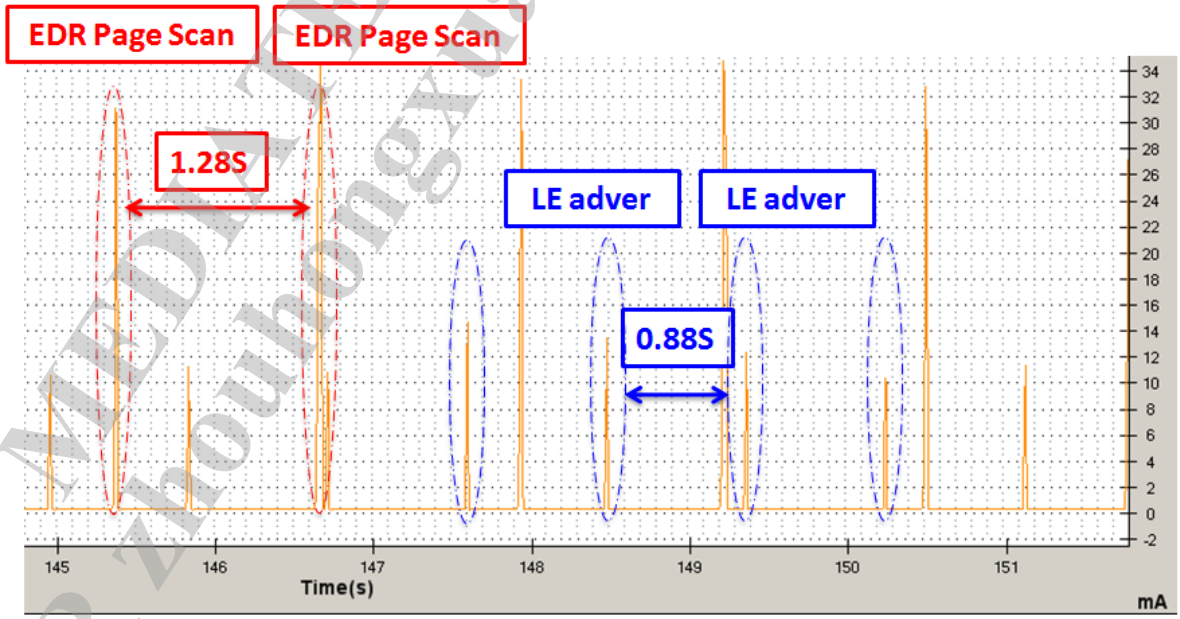
MT2502 在 BT 不工作情况下 MT2502 耗电

MT2502 在 BT 不工作情况下, MT2502 耗电为: 0.45mA.



BT 打开, 不连接

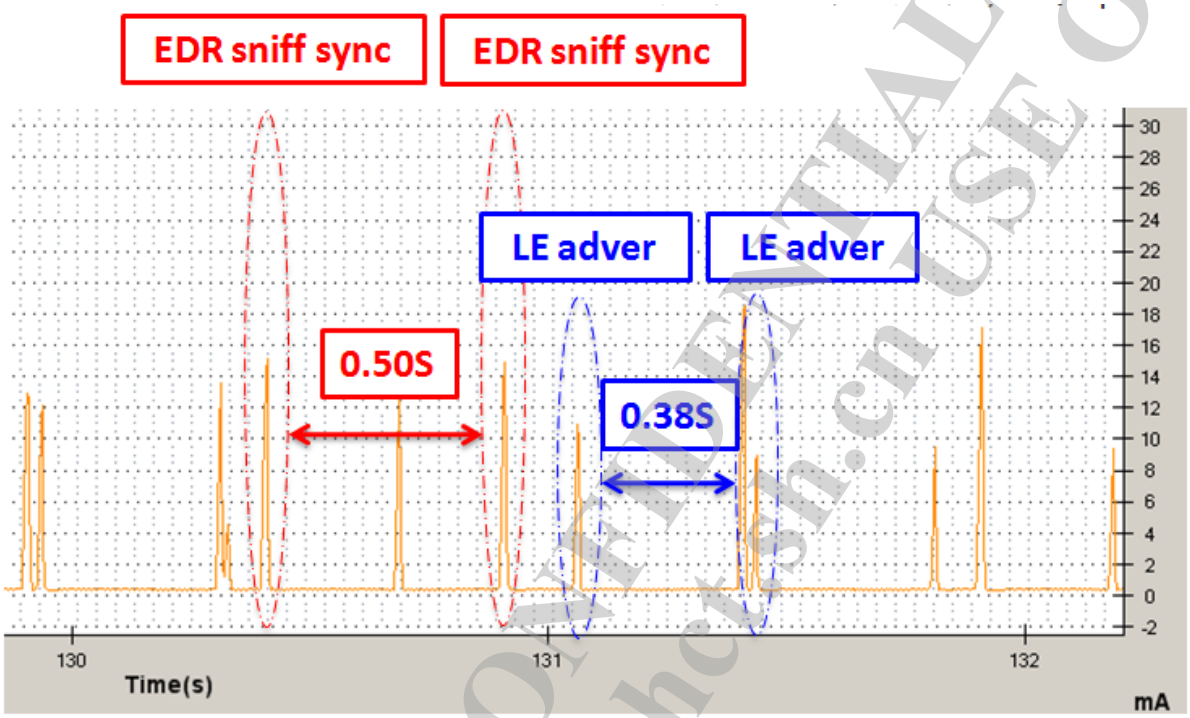
BT on, EDR/LE 都不连接, 平均电流: 1.17mA.



BT 打开, EDR 连接, LE 不连接

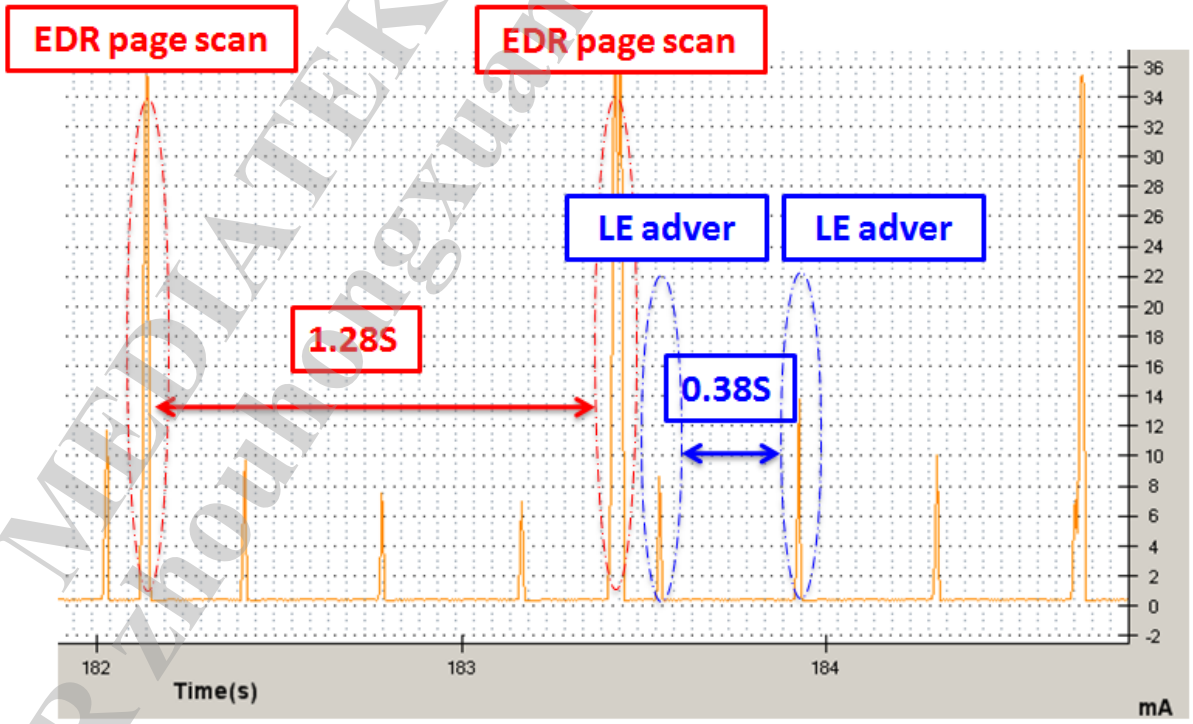
BT on, EDR 连接, LE 不连接, 平均电流: 0.88mA.

EDR sniff 是根据双方协商设定, 连接 iPhone sniffer timer 是 0.5S



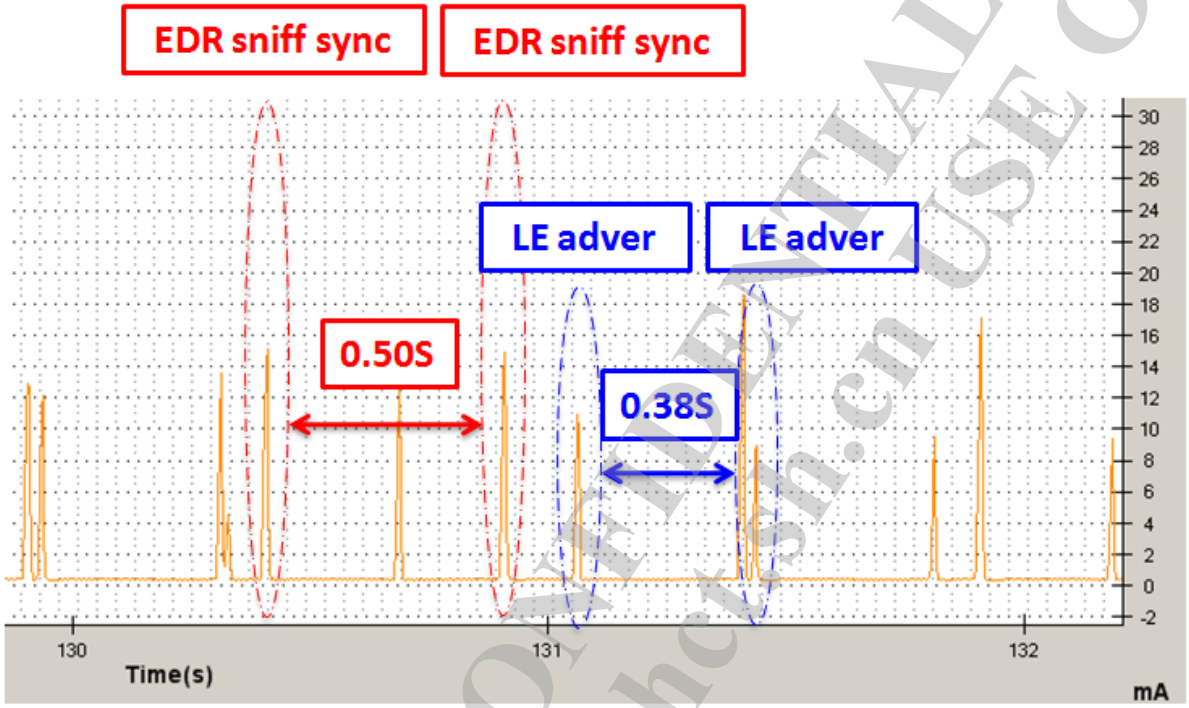
Aster V1 BT 打开, EDR 不连接, LE 连接

Aster V1 BT on, EDR 不连接, LE 连接, 平均电流: 1.20mA.



Aster V1 BT 打开, EDR 连接, LE 连接

Aster V1 BT on, EDR 连接, LE 连接, 平均电流: 0.96mA.
EDR sniff 是根据双方协商设定, 连接 iPhone sniffer timer 是 0.5S



MT2502 Aster V1 数据:

	MT2502 整机耗电	MT2502 BT 部分耗电
MT2502 在 BT 不工作情况下, MT2502 耗电	0.45mA	0.00mA
BT on, EDR/LE 都不连接	1.17mA	0.72mA
BT on, EDR 连接, LE 不连接	0.88mA	0.43mA
BT on, EDR 不连接, LE 连接	1.20mA	0.75mA
BT on, EDR 连接, LE 连接	0.96mA	0.51mA

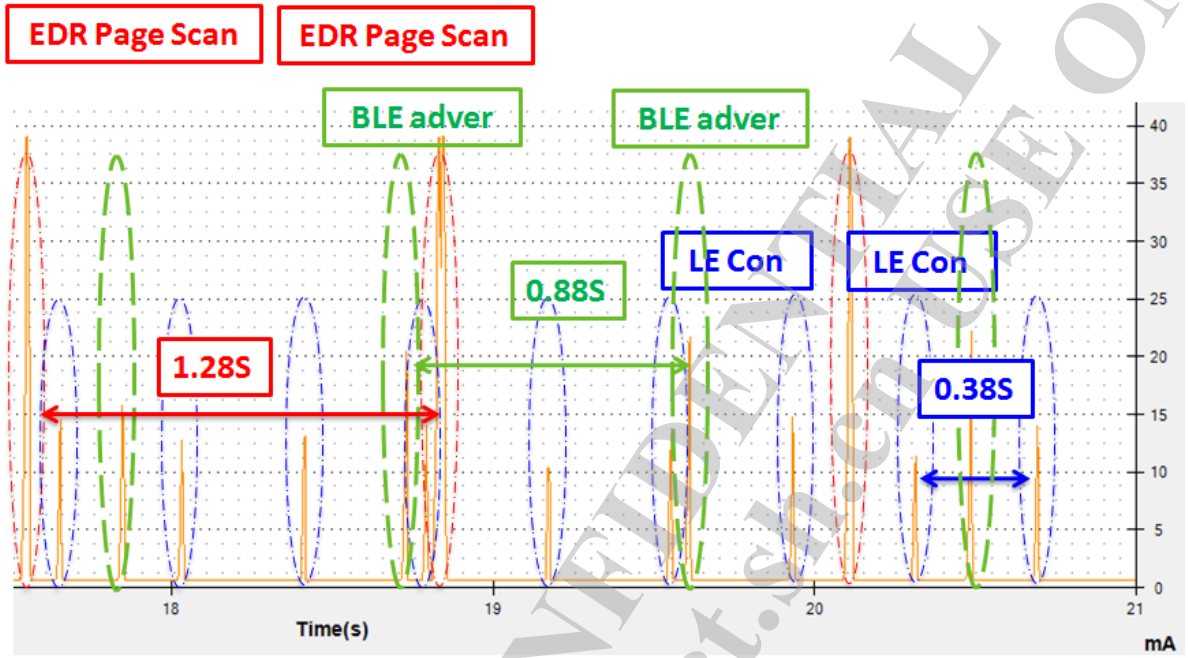
MT2502 Aster V2 数据:

	MT2502 整机耗电	MT2502 BT 部分耗电
MT2502 在 BT 不工作情况下, MT2502 耗电	0.67mA	0.00mA
BT on, EDR/LE 都不连接	1.27mA	0.60mA
BT on, EDR 连接, LE 不连接	1.18mA	0.51mA
BT on, EDR 不连接, LE 连接	1.66mA	0.99mA
BT on, EDR 连接, LE 连接	1.54mA	0.87mA

由于 Aster V2 增加 BLE multi-link, 即在 BLE 连接上一路的场景下, 还会再打出 advertising 包, 所以功耗会稍大点, 即在 BLE 连接上的场景下的功耗会比 Aster V1 稍大点.
如果 BLE 连接的路数越多, 功耗也会越稍微大点, 最多是在 4 条 BLE 连接都存在场景下.

Aster V2 BT 打开, EDR 不连接, LE 连接

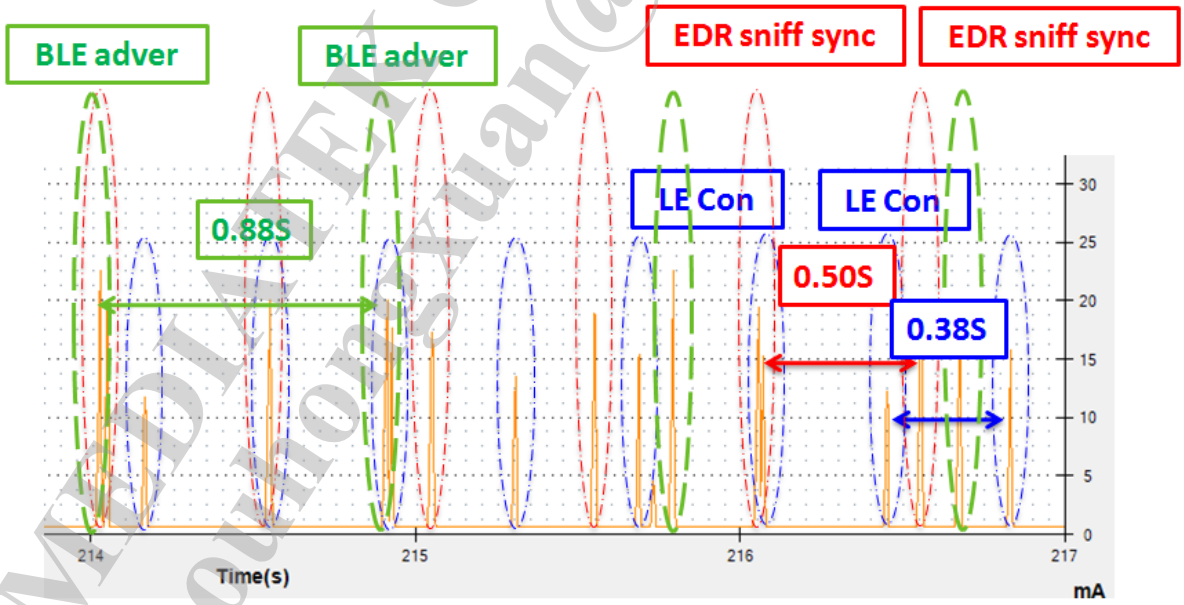
Aster V2 BT on, EDR 不连接, LE 连接, 平均电流: 1.66mA.



Aster V2 BT 打开, EDR 连接, LE 连接

Aster V2 BT on, EDR 连接, LE 连接, 平均电流: 1.54mA.

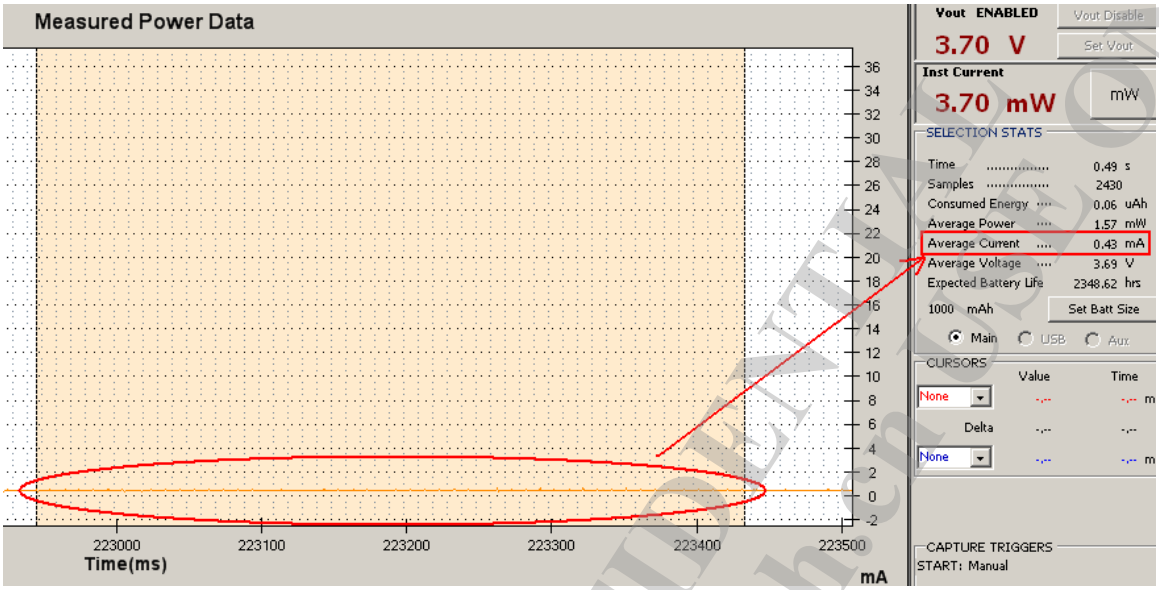
EDR sniff 是根据双方协商设定, 连接 iPhone sniffer timer 是 0.5S



3.2 MT2502 蓝牙 BLE only 单模版本电流数据

MT2502 在 BT 不工作情况下 MT2502 耗电

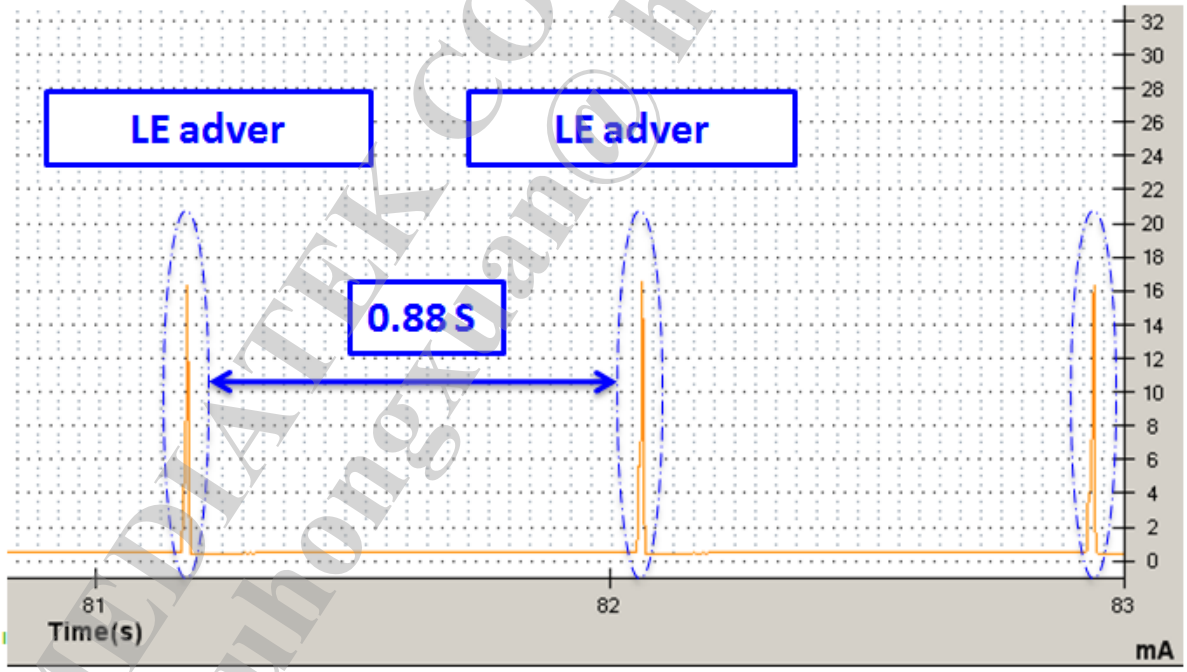
MT2502 在 BT 不工作情况下, MT2502 耗电为: 0.43mA.



BT 打开, 不连接

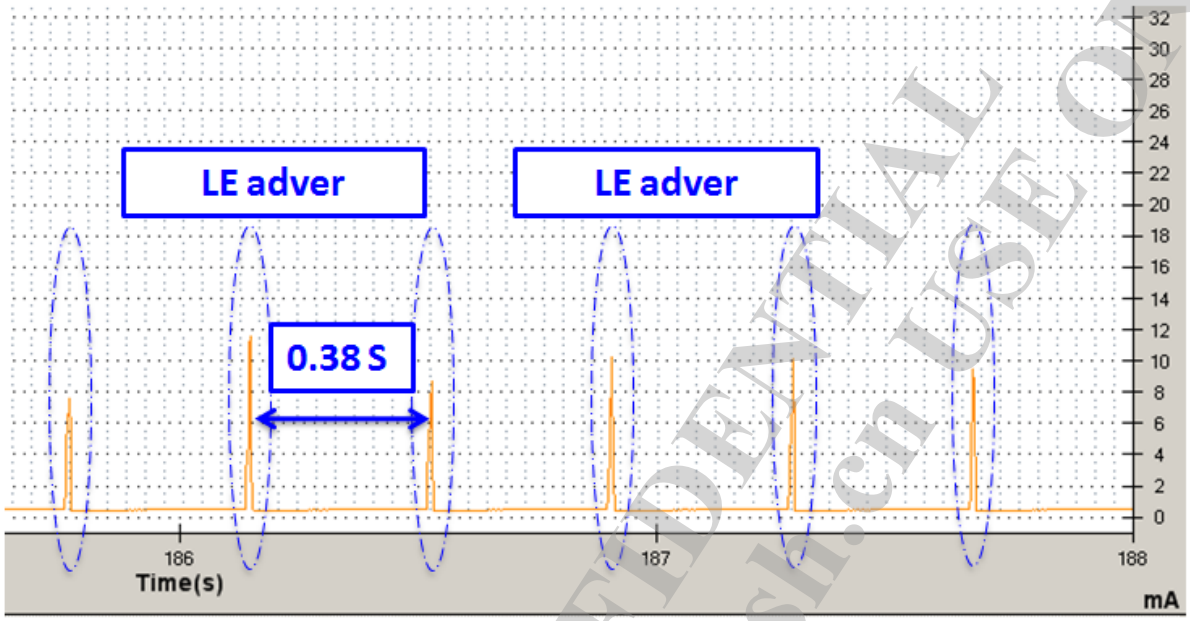
BT on, LE 都不连接, 平均电流: 0.58mA.

Only LE 存在, 可被 Smart Phone 搜索到.



BT 打开, LE 连接

BT on, LE 连接, 平均电流: 0.65mA.



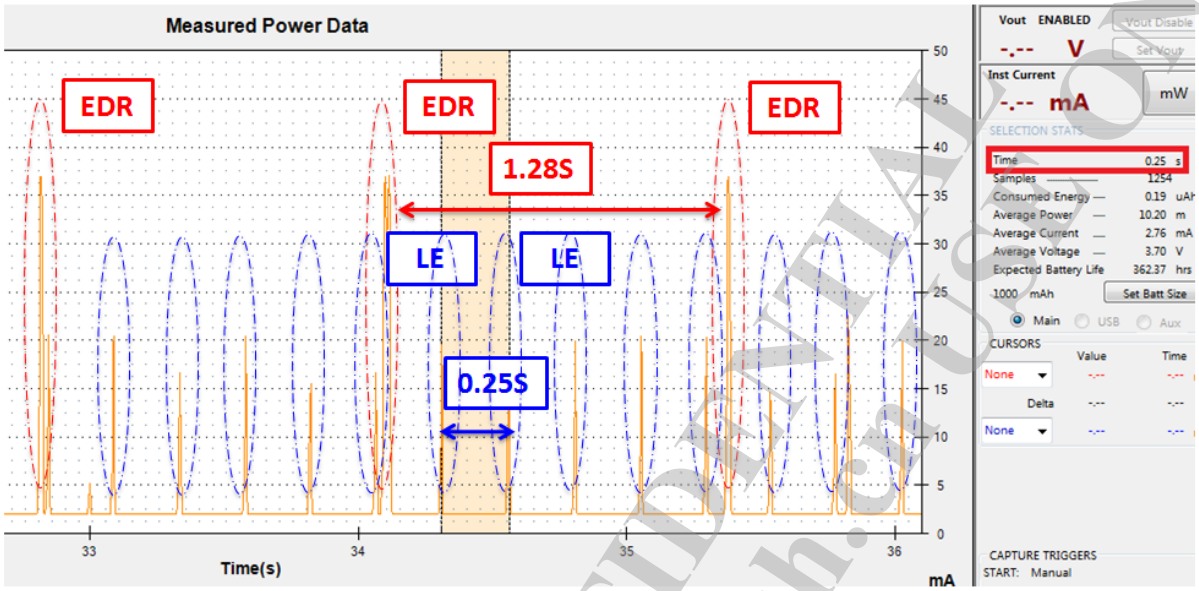
	MT2502 整机耗电	MT2502 BT 部分耗电
MT2502 在 BT 不工作情况下, MT2502 耗电	0.43mA	0.00mA
BT on, LE 不连接	0.58mA	0.15mA
BT on, LE 连接	0.65mA	0.22mA

3.3 MT2502 Aster V1 Tracker 版本在打开蓝牙之后, LE 的 adv interval 不是 880mS (是 240mS), 引起打开蓝牙不连接场景电流偏大.

MT2502 的代码包是以 11CW1418SP4 开头, 在代码包名称中, V15 以下(不含 V15)版本为 MT2502 Aster V1 版本.

“TRACKER”版本 => 代码包名称中包含“TRACKER”
“WATCH”版本 => 代码包名称中包含“WATCH”
“WT”版本(WATCH+TRACKER) => 代码包名称中包含“WT”

- ⇨ TRACKER 版本主 makefile 里 “MMI_VERSION = IOT_MMI”, 所以 “mmi_app” core 不会 build.
- ⇨ LE adv interval 是在 PxpAppMain.c 中启动 timer, 来设定 880mS adv interval 到蓝牙协议栈. 而 PxpAppMain.c 是在 “mmi_app” 这部分 core.
- ⇨ 通过 Catcher log 可以检查到 MMI 没有将 “MSG_ID_BT_GATTC_SET_ADV_PARAM_REQ” (min interval: 0x0320; max interval: 0x07d0) 发送到 BT.



修改方法:

将 PXP 那部分设置 adv interval 的逻辑, 直接挪到 gattcsrv.c.

(1), 首先是 event: SRV_BT_CM_EVENT_ACTIVE 以及 SRV_BT_CM_EVENT_PANIC_IND 的 handler 注册时机.

pxp 是在 bootup 阶段去注册的, 所以当挪到 gattcsrv 时, 可以放在 srv_gattc_init() API 里面去注册这两个 event, 仿照 mmi_pxp_server_init() 以及 mmi_bt_event_notify_pxp_app() 去写.

注意: 要 include “BtcmSrvGprot.h”.

(2), 直接将 mmi_pxpapp_set_adv_pattern() 以及 mmi_pxpapp_set_adv_param_req() 两个 API, 搬到 gattcsrv, 修改一下函数名, 及变量名即可.

(3), 设置 adv interval (即 call mmi_pxpapp_set_adv_pattern) 的时机:

一是: 在类似 mmi_bt_event_notify_pxp_app() 的仿造函数里, 收到 SRV_BT_CM_EVENT_ACTIVE 时去触发.

二是: 在 disconnect 时去触发, 即在 gattc_handle_disconnected_ind() API 中, 在 callback->connection_cb(&conn, false, ind->bd_addr) 语句之后触发.

(4), 由于代码执行中, 会用到一个 30s 的 timer, 在 PXP 的设计时, Timer ID 是直接定义在 pxpapp.res 的, 但是要利用相同方法, 在 GattcSRv 中定义这样 一 个 timer, 需要贵司自己添加一个 GattSrv.res file 并将 Timer ID 定义在其中.

3.4 MT2502 Aster V1 版本 BT(BR/EDR)&BLE 连接 MX4 (MT6595) 等个别 Smart Phone 之后, 底电流 10mA, 功耗大.

MT2502 的代码包是以 11CW1418SP4 开头, 在代码包名称中, V15 以下(不含 V15)版本为 MT2502 Aster V1 版本.

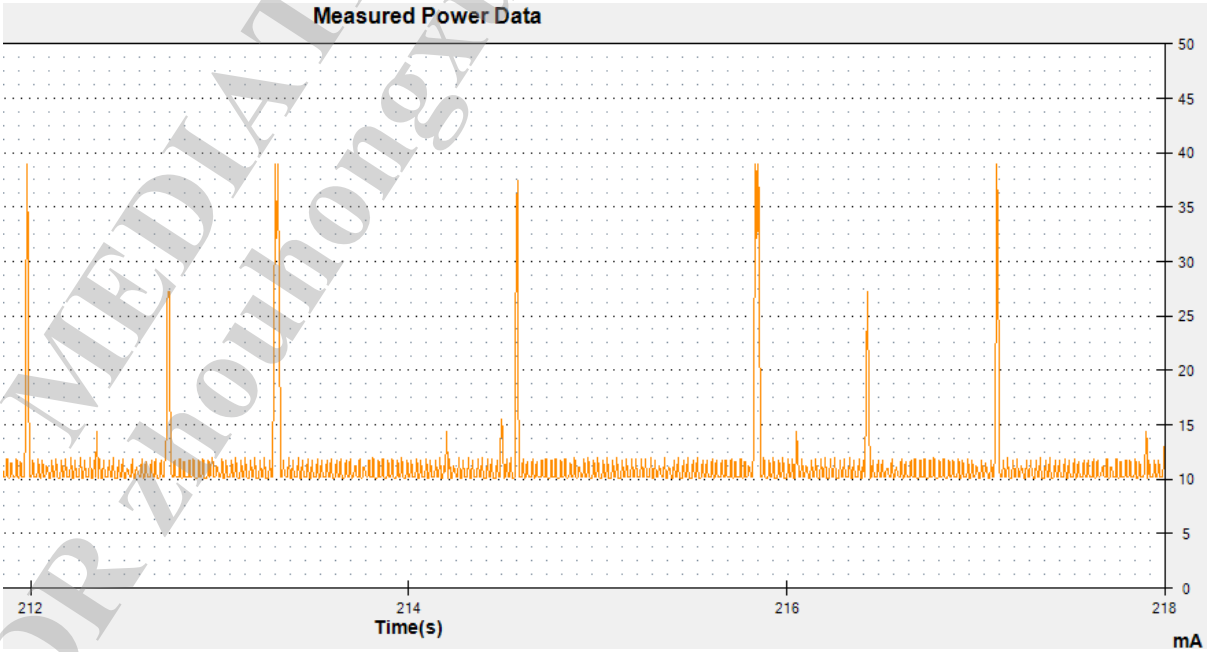
产生该问题的 MT2502 Aster V1 版本中, BR/EDR 的 profile 只有开 SPP(BT_SPP_PROFILE / BT_SPP_CLIENT / BT_SPP_SERVER)才会发生.

MediaTek MT2502 Aster V1 触发 BR/EDR 进入 sniff mode 条件是通过 HFP profile (BT_HF_PROFILE) / A2DP profile (BT_A2DP_PROFILE)触发. MT2502 Aster V2 版本有增加 SPP 触发.

如果贵司 MT2502 Aster V1 工程中将 HFP profile (BT_HF_PROFILE) / A2DP profile (BT_A2DP_PROFILE) 这两个 profile 关闭, 会出现 BR/EDR 连接之后, 在连接不会主动发起 sniff mode 的 Smart Phone, 会出现无法进入 sniff mode 的问题, 从而引起 MT2502 功耗大.

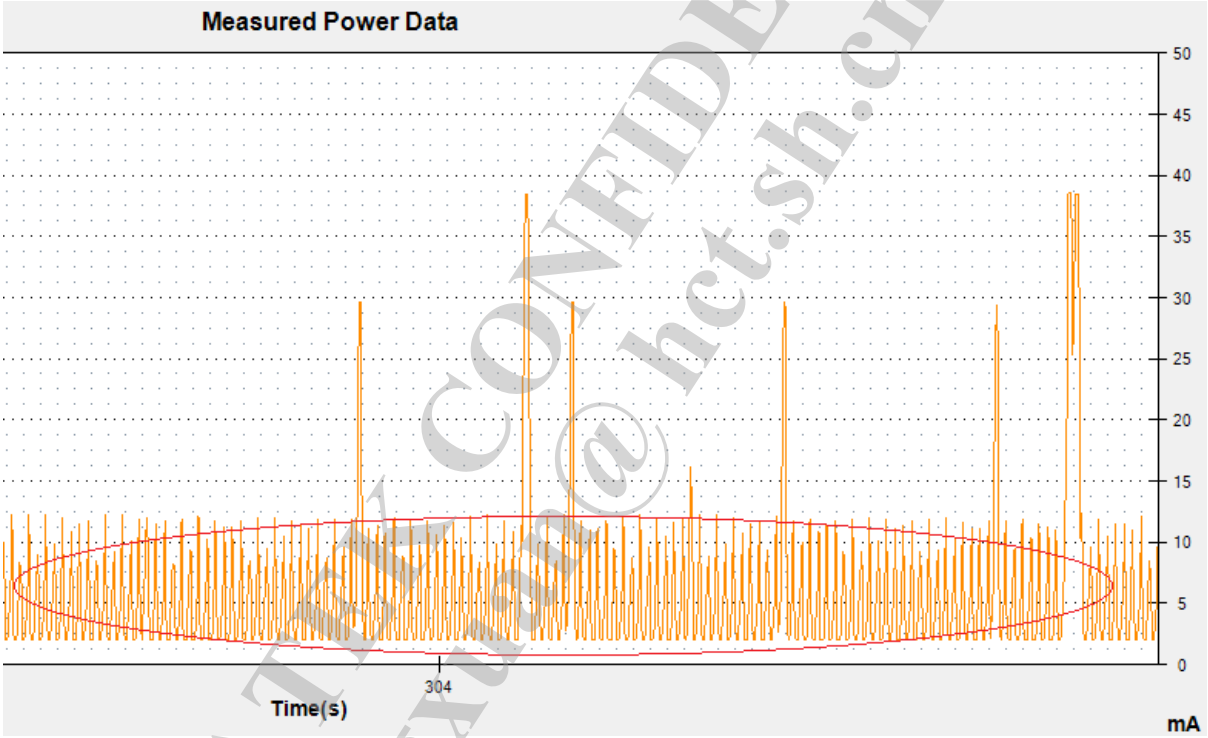
请检查贵司的 MT2502 Aster V1 工程中, HFP profile (BT_HF_PROFILE) / A2DP profile (BT_A2DP_PROFILE) 是否有关闭, 如果这两个 profile 是关闭状态, SPP(BT_SPP_PROFILE / BT_SPP_CLIENT / BT_SPP_SERVER)是打开状态, MT2502 在 BR/EDR SPP 连接 MediaTek 平台的 Smart Phone 或者其他平台的 Smart Phone(个别平台, iPhone / Samsung 是会主动发起 sniff mode), 由于双方都不发送 sniff request, 从而无法进入 sniff mode, 导致 MT2502 底电流大. 需要申请增加 SPP 触发 sniff mode patch.

解决方法:
Patch id: MAUI_03539385



3.5 MT2502 BT(BR/EDR)&BLE 连接 HUAWEI P7 (Broadcom BT) 之后, 底电流正常, conn interval 不是 380mS (是 20mS), 引起功耗偏大.

MT2502 MMI 有发 MSG_ID_BT_GATTC_CONN_PARAM_UPDATE_REQ (min interval: 0x0120; max interval: 0x0130) 给到 BT, 同时 BT 有回 MSG_ID_BT_GATTC_CONN_PARAM_UPDATE_CNF(result 为 0), 但是 conn interval 还是 20mS.



MediaTek 尝试多组 conn interval 参数, 但是连接 huawei P7 的 conn interval 还是 20mS. 这个问题是 huawei 的问题, 因为 MediaTek 2502 有和对方 sync conn interval. Huawei 回复 ok, 但是没有 update. 通过抓取的 Air Sniff Log 可以确定.

1, 第一次 update 为 20ms 成功.

Smart Phone huawei P7 端收到 MT2502 发送的 update request 并回复 accepted.

L2CAP Parameter Update (Min=20.00 ms, Max=20.00 ms, Lat=0, T/o=6'000 ms)	14:39:44.462 446 875
L2CAP Connection Parameter Update Request (Min=20.00 ms, Max=20.00 ms, Lat=0, T/o=6'000 ms)	14:39:44.462 446 875
L2CAP Connection Parameter Update Response (Accepted)	14:39:44.510 861 250

大概 2S 后, Smart Phone huawei P7 端真正 update connection interval 为 20ms.

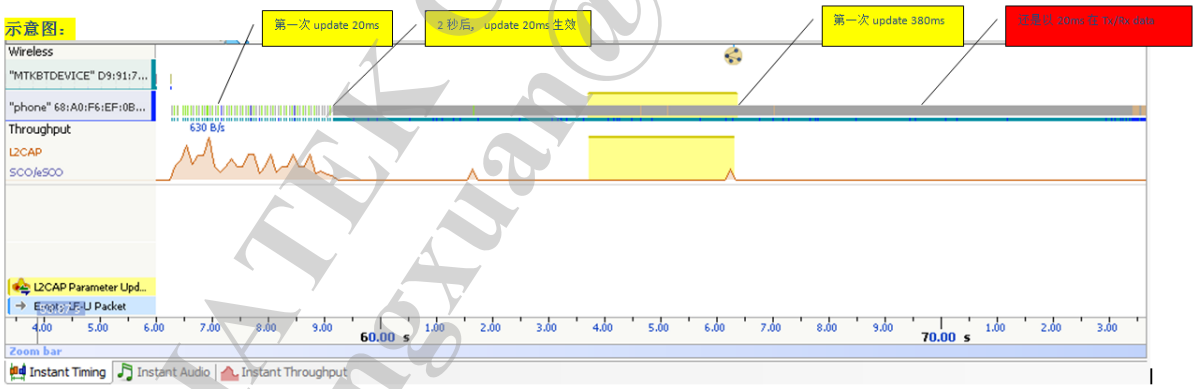
LLCP Connection Update Request (WinSz=3.75 ms, WinOfs=7.50 ms, Int=20.00 ms, Lat=0, T/o=6'000 ms, Inst=59)	14:39:46.850 877 375
LE-C Transfer (1 retry) (Connection Update Request)	14:39:46.850 877 375

2, 第二次 update 为 380ms 失败.

Smart Phone huawei P7 端收到 MT2502 发送的 update request 并回复 accepted.

没有看到 Smart Phone huawei P7 端真正 update connection interval 为 380ms.

L2CAP Parameter Update (Min=360.00 ms, Max=380.00 ms, Lat=4, T/o=6'000 ms)	14:39:54.291 277 000
L2CAP Connection Parameter Update Request (Min=360.00 ms, Max=380.00 ms, Lat=4, T/o=6'000 ms)	14:39:54.291 277 000
L2CAP Connection Parameter Update Response (Accepted)	14:39:54.311 047 250
LLCP Channel Map Request (Used: 0-10,21-36 / Unused: 11-20, Inst=493)	14:39:55.711 129 500
LLCP Channel Map Request (Used: 10,21-36 / Unused: 0-9,11-20, Inst=693)	14:39:59.711 358 125
LLCP Channel Map Request (Used: 21-36 / Unused: 0-20, Inst=1'110)	14:40:03.711 588 000
LLCP Channel Map Request (Used: 24-36 / Unused: 0-23, Inst=1'293)	14:40:11.711 734 500
LLCP Channel Map Request (Used: 25-36 / Unused: 0-24, Inst=2'893)	14:40:43.712 389 125
LLCP Channel Map Request (Used: 25-29,31-36 / Unused: 0-24,30, Inst=3'093)	14:40:47.712 583 750
LLCP Channel Map Request (Used: 25,31-36 / Unused: 0-24,26-30, Inst=3'360)	14:40:51.753 271 000
L2CAP Signaling Reserved (0xDD) (15 bytes (85 D7 C9 05 98 30 2A 5F 5B D5 C6 A1 8B D7 89))	14:41:35.294 625 125
LLCP Channel Map Request (Used: 0-1,29,31-36 / Unused: 2-28,30, Inst=6'693)	14:41:59.715 002 875
LLCP Channel Map Request (Used: 0,29-36 / Unused: 1-28, Inst=7'493)	14:42:15.715 549 125
LLCP Channel Map Request (Used: 4,9,29,31-36 / Unused: 0-3,5-8,10-28,30, Inst=8'693)	14:42:39.716 105 750
LLCP Channel Map Request (Used: 4,8,29,31-36 / Unused: 0-3,5-7,9-28,30, Inst=9'494)	14:42:55.716 350 000
LLCP Reserved (0x79)	14:42:56.836 360 250
LLCP Channel Map Request (Used: 2,4,29,31-36 / Unused: 0-1,3,5-28,30, Inst=10'493)	14:43:15.717 064 250
LLCP Channel Map Request (Used: 25-26,28-29,31-36 / Unused: 0-24,27,30, Inst=10'694)	14:43:19.717 752 375
LLCP Channel Map Request (Used: 2-3,5-10,25-26,29,31-36 / Unused: 0-1,4,11-24,27-28,30, Inst=10'894)	14:43:23.717 518 750
Empty LE-U (x 262)	14:43:37.898 251 375



该问题是 Smart Phone huawei P7 端问题.

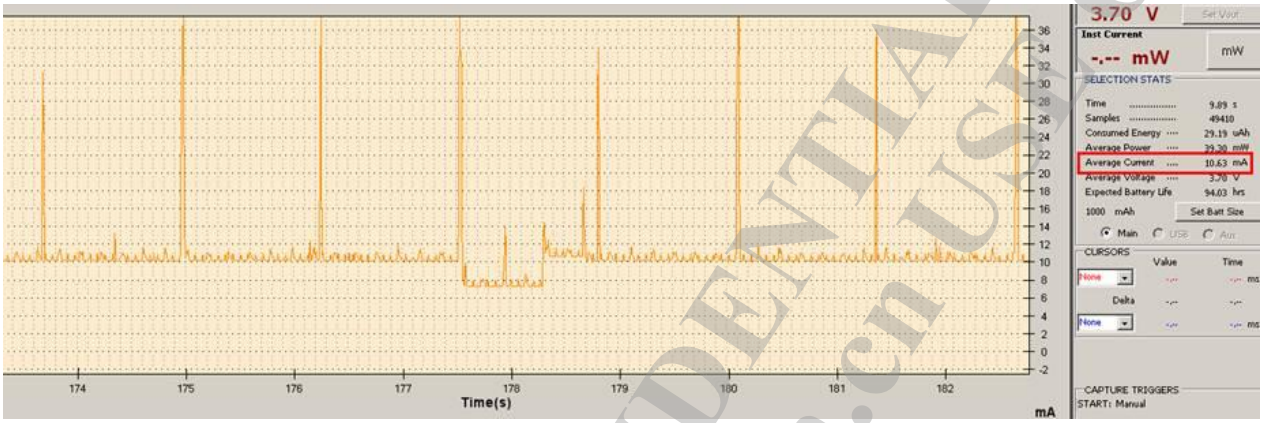
3.6 苹果手机下载 IPA 联发设备宝连接 MT2502 手表, 在 IPA 中打开 “警报” “区域报警”, MT2502 待机平均电流 12.3mA, 最小电流 10.1mA, 电流值偏大.

描述问题:

苹果手机下载 IPA 联发设备宝连接 MT2502 手表, 在 IPA 中打开 “警报” “区域报警”, MT2502 待机平均电流 12.3mA, 最小电流 10.1mA, 电流值偏大.

定位问题:

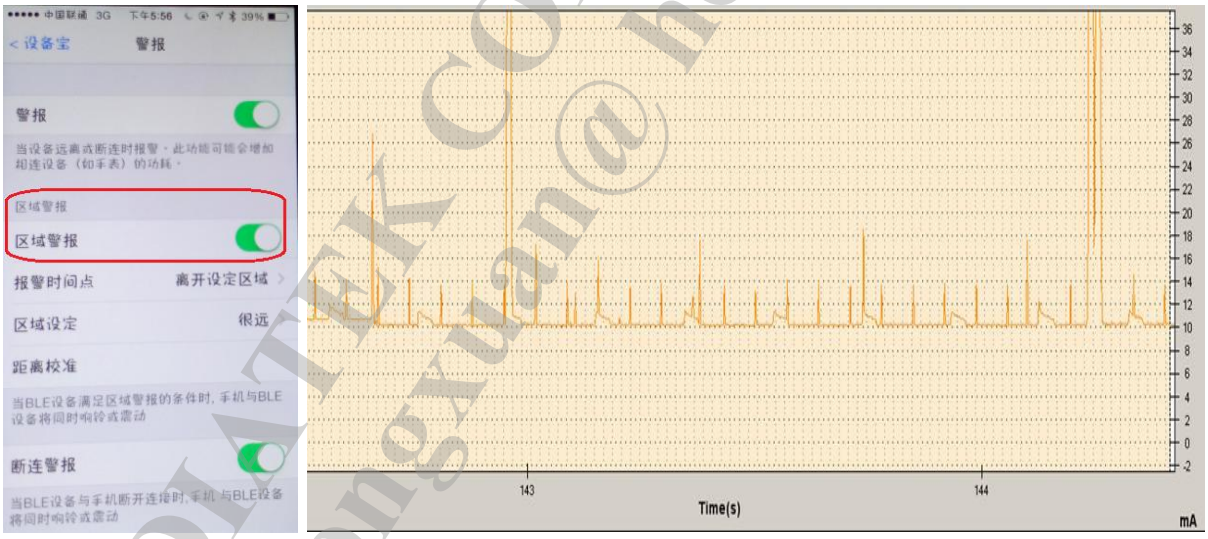
使用 iPhone5C 测试, 操作 IPA (联发设备宝) 连接 MT2502 手表, 在 IPA 中打开 “警报” “区域报警”, MT2502 手表待机平均电流 10.63mA.



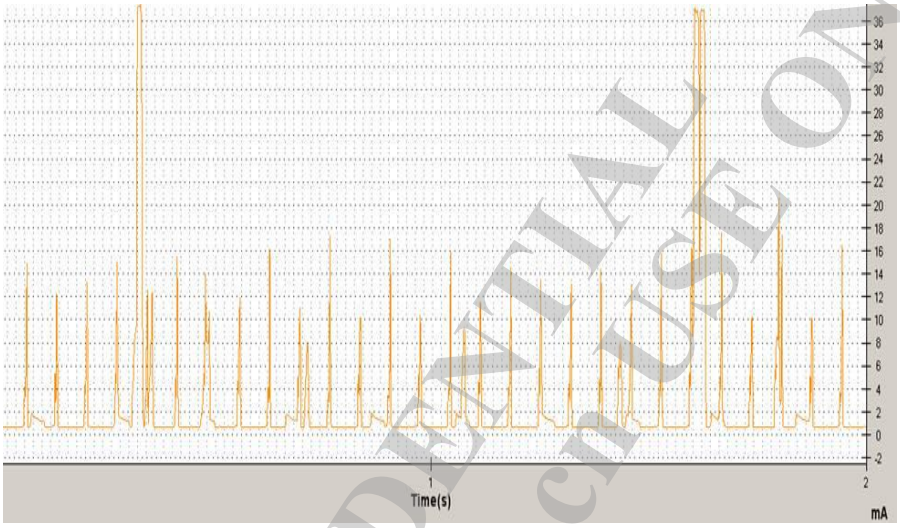
厘清问题:

连接 iPhone5C 后, 操作 IPA (联发设备宝) 打开 “警报” “区域报警”, 导致底电流不正常. 从 0.5mA 变为 10.12mA.

打开 “警报” “区域报警” 平均电流 10.72mA:



关闭 “警报” “区域报警” 平均电流 2.37mA:



抓取 打开 “警报” “区域报警” 平均电流 10.72mA 场景下, MT2502 的 catcher log, 发现是只要 IPA 联发设备宝中打开“区域报警”.

iPhone 端会不停向 MT2502 端通过 GATT 获取 TX power. 从而导致 M2502 无法 sleep, 继而底电流从 0.5mA 变为 10.12mA.

Local Time	Source	Destination	SAP	Message
17:45:52:542 2015/04/28	MOD_BT	MOD_MMI	INVALID_SAP	MSG_ID_BT_GATTS_READ_REQUEST_IND
17:45:52:542 2015/04/28	MOD_MMI	MOD_BT	BT_APP_SAP	MSG_ID_BT_GATTS_READ_TX_POWER_REQ
17:45:52:542 2015/04/28	MOD_BT	MOD_MMI	INVALID_SAP	MSG_ID_BT_GATTS_READ_TX_POWER_CNF
17:45:52:542 2015/04/28	MOD_MMI	MOD_BT	BT_APP_SAP	MSG_ID_BT_GATTS_SEND_RESPONSE_REQ
17:45:53:042 2015/04/28	MOD_BT	MOD_MMI	INVALID_SAP	MSG_ID_BT_GATTS_SEND_RESPONSE_CNF
17:45:57:846 2015/04/28	MOD_BT	MOD_MMI	INVALID_SAP	MSG_ID_BT_GATTS_READ_REQUEST_IND
17:45:57:846 2015/04/28	MOD_MMI	MOD_BT	BT_APP_SAP	MSG_ID_BT_GATTS_READ_TX_POWER_REQ
17:45:57:846 2015/04/28	MOD_BT	MOD_MMI	INVALID_SAP	MSG_ID_BT_GATTS_READ_TX_POWER_CNF
17:45:57:846 2015/04/28	MOD_MMI	MOD_BT	BT_APP_SAP	MSG_ID_BT_GATTS_SEND_RESPONSE_REQ
17:45:57:846 2015/04/28	MOD_BT	MOD_MMI	INVALID_SAP	MSG_ID_BT_GATTS_SEND_RESPONSE_CNF
17:46:04:778 2015/04/28	MOD_BT	MOD_MMI	INVALID_SAP	MSG_ID_BT_GATTS_READ_REQUEST_IND
17:46:04:778 2015/04/28	MOD_MMI	MOD_BT	BT_APP_SAP	MSG_ID_BT_GATTS_READ_TX_POWER_REQ
17:46:04:793 2015/04/28	MOD_BT	MOD_MMI	INVALID_SAP	MSG_ID_BT_GATTS_READ_TX_POWER_CNF
17:46:04:793 2015/04/28	MOD_MMI	MOD_BT	BT_APP_SAP	MSG_ID_BT_GATTS_SEND_RESPONSE_REQ
17:46:04:793 2015/04/28	MOD_BT	MOD_MMI	INVALID_SAP	MSG_ID_BT_GATTS_SEND_RESPONSE_CNF
17:46:10:441 2015/04/28	MOD_BT	MOD_MMI	INVALID_SAP	MSG_ID_BT_GATTS_READ_REQUEST_IND
17:46:10:441 2015/04/28	MOD_MMI	MOD_BT	BT_APP_SAP	MSG_ID_BT_GATTS_READ_TX_POWER_REQ
17:46:10:441 2015/04/28	MOD_BT	MOD_MMI	INVALID_SAP	MSG_ID_BT_GATTS_READ_TX_POWER_CNF
17:46:10:441 2015/04/28	MOD_MMI	MOD_BT	BT_APP_SAP	MSG_ID_BT_GATTS_SEND_RESPONSE_REQ
17:46:10:721 2015/04/28	MOD_BT	MOD_MMI	INVALID_SAP	MSG_ID_BT_GATTS_SEND_RESPONSE_CNF
17:46:16:556 2015/04/28	MOD_BT	MOD_MMI	INVALID_SAP	MSG_ID_BT_GATTS_READ_REQUEST_IND
17:46:16:556 2015/04/28	MOD_MMI	MOD_BT	BT_APP_SAP	MSG_ID_BT_GATTS_READ_TX_POWER_REQ
17:46:16:556 2015/04/28	MOD_BT	MOD_MMI	INVALID_SAP	MSG_ID_BT_GATTS_READ_TX_POWER_CNF
17:46:16:556 2015/04/28	MOD_MMI	MOD_BT	BT_APP_SAP	MSG_ID_BT_GATTS_SEND_RESPONSE_REQ
17:46:16:556 2015/04/28	MOD_BT	MOD_MMI	INVALID_SAP	MSG_ID_BT_GATTS_SEND_RESPONSE_CNF

因为操作 IPA 联发设备包, 打开 Alert 开关之后(即打开 “警报” “区域报警”), iPhone 会每隔 5S 从 watch read TX power 一次, 导致 MT2502 watch 无法进入 sleep.

区域报警的作用是: 当 watch 离开 iPhone 超过预设距离, 或者进入预设范围, 就会 Alert 提示 User, 这就要求 iPhone 端不断监控两者的距离.

由于 IOS 有一个限制: IPA 切到后台之后, 大概 10S 左右, 就会被 IOS 系统 Kill 掉, 不允许 IPA 一直处于 Running 的状态.

在 IOS 系统看来, 如果 IPA always 需要处理外部事件, 则 IOS 系统允许 IPA always 运行.

所以 MediaTek IOS IPA 的写法是每隔 5S 去 read TX power 外部事件，保持 IPA 切到后台之后，IPA 一直处于运行状态.

MediaTek 最新上架 IPA 和 release 出去的 code，已经在这个开关后面增加一行提示，告诉 User，如果打开这个开关(打开 “警报” “区域报警”), 会导致 MT2502 watch target 端更加耗电.

目前是这样的.

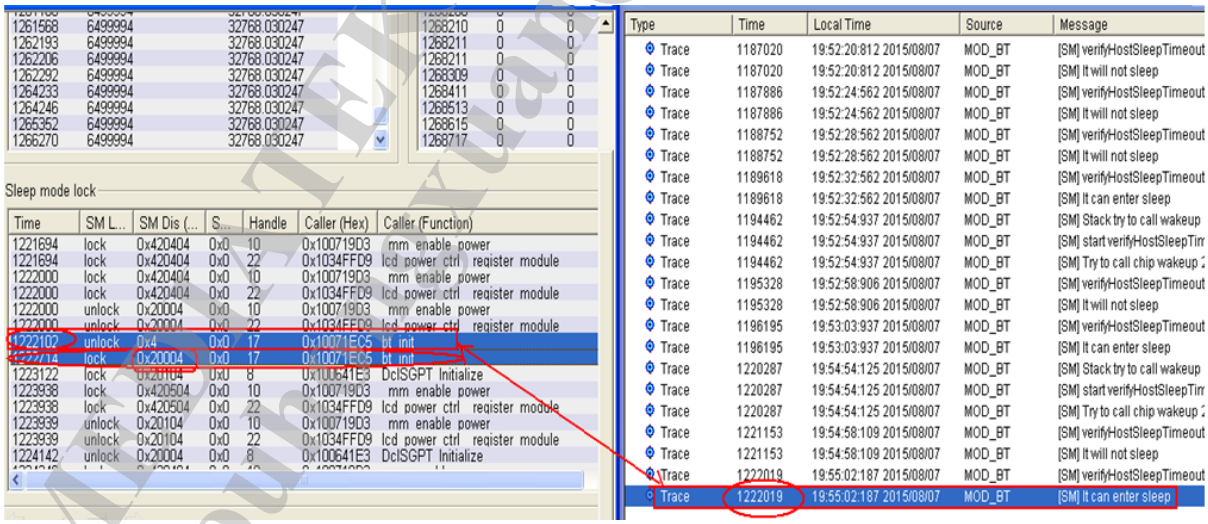
3.7 MT2502 通过 BLE 进行蓝牙操作后，概率发生待机电流 8mA.

描述问题:

MT2502 进行蓝牙 BLE 操作后，低概率出现待机电流增加 8mA.
操作过程: 摇摇 -> MCU 上报摇摇事件 -> 蓝牙搜素 -> 蓝牙连接 -> 交换联系人数据 -> 蓝牙断开，如此反复.
复现问题后，如果开关一次蓝牙，问题会消失.

厘清问题:

UART catcher log，结合 catcher log 中的 sleep 信息，看到是 bt_init 注册的 sleep handler 没有释放而引起.
之后的 sleep 信息中一直可以看到 0x20000 bt_init() 获得的 sleep handler，这个 handler 一直 lock 着，导致系统无法 sleep.



解决方法:

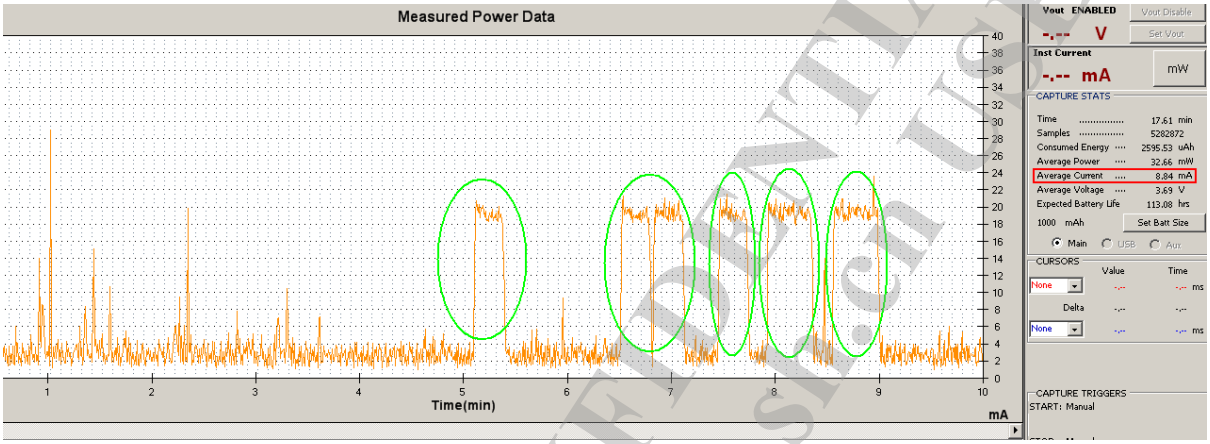
Patch id: MAUI_03543022

3.8 MT6261 Phone 版本连接蓝牙耳机功耗偏高

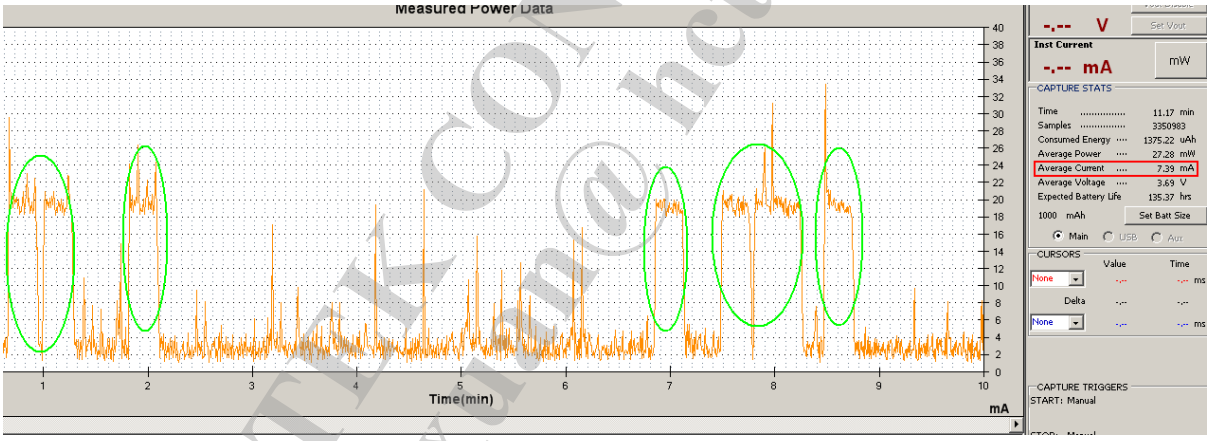
描述问题:

MT6261 MAUI.11C.W13.52.MP 版本代码, 只有 Phone 功能.
MT6261 连接任意一款蓝牙耳机, 进行插入实网 SIM 卡, 同时连接蓝牙耳机后, 进入 IDLE 模式, 测试时间为 10 分钟的测试结果, 平均电流在 8 毫安左右.

测试结果 1: 平均电流 8.84mA

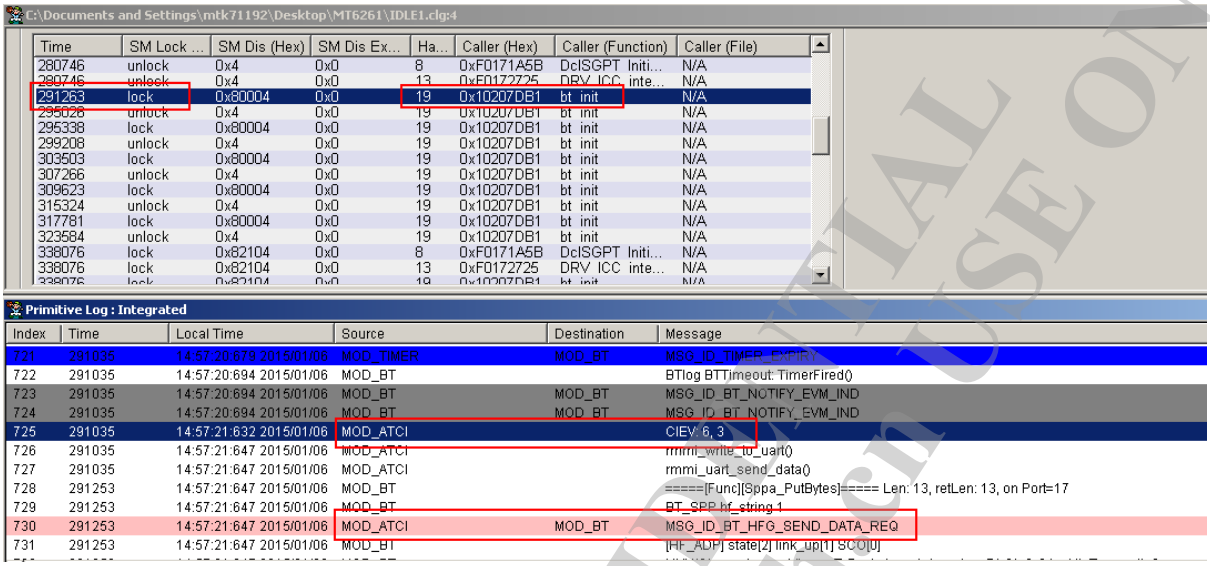


测试结果 2: 平均电流 7.39mA



厘清问题:

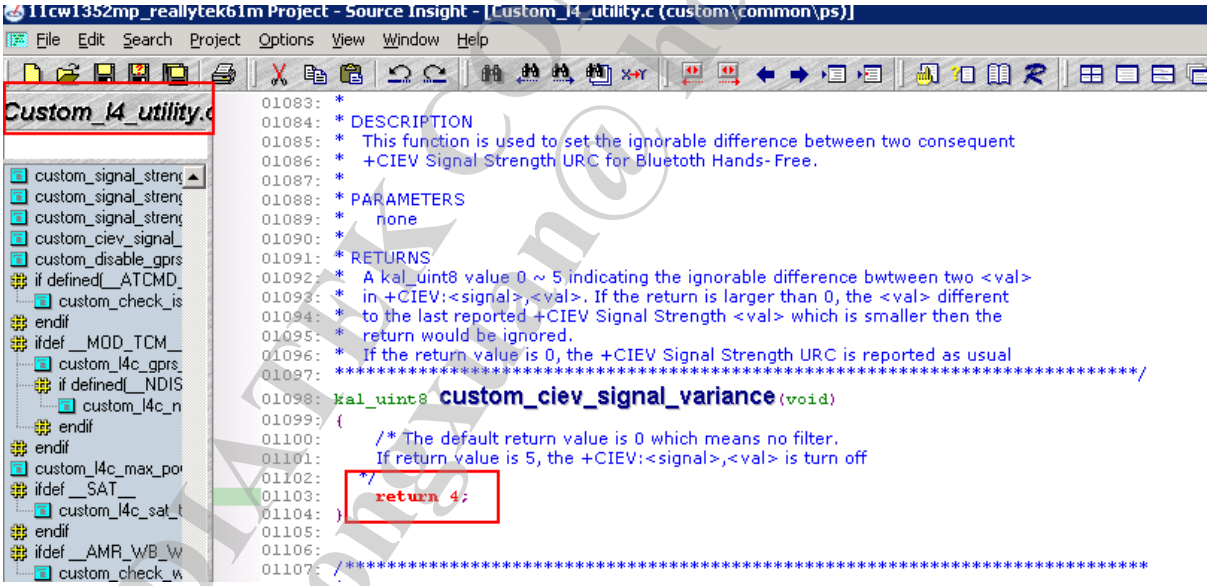
由于出现上两幅图中 绿色线圈中的 20mA 左右高耗电时段, 从而将整体的平均电流抬高.
于是在记录 IDLE 下电流出现绿色线圈中的 20mA 左右高耗电时段, 同时录制 MT6261 的 UART1 Catcher log. Filter: MOD_BT/MOD_ATCI/ Sleep.
看到是 bt_init() 中获得的 sleep handler 在 lock, 从而使电流抬升.
所以说明该问题是和 BT 相关. 确实是 BT 在 active, 使电流变大.
BT 为什么会 active 是有原因的, 是因为 ATCI 发送 MSG_ID_BT_HFG_SEND_DATA_REQ 给 BT, 要 BT 从 signal cmd (CIEV: 6, 3) 给到蓝牙耳机.
查至此, 涉及 ATCI 相关代码, ATCI 代码都是 lib 的形式给到贵司, 所以贵司无法看到. 我大致讲下这段代码的写法.
判断 连接蓝牙耳机的 signal ind, 是否大于 ciev signal 变化的预设值, 如果大于, 马上发送连接该蓝牙耳机的 signal ind 确认.
所以就产生了绿色线圈中的 20mA 左右高耗电时段.



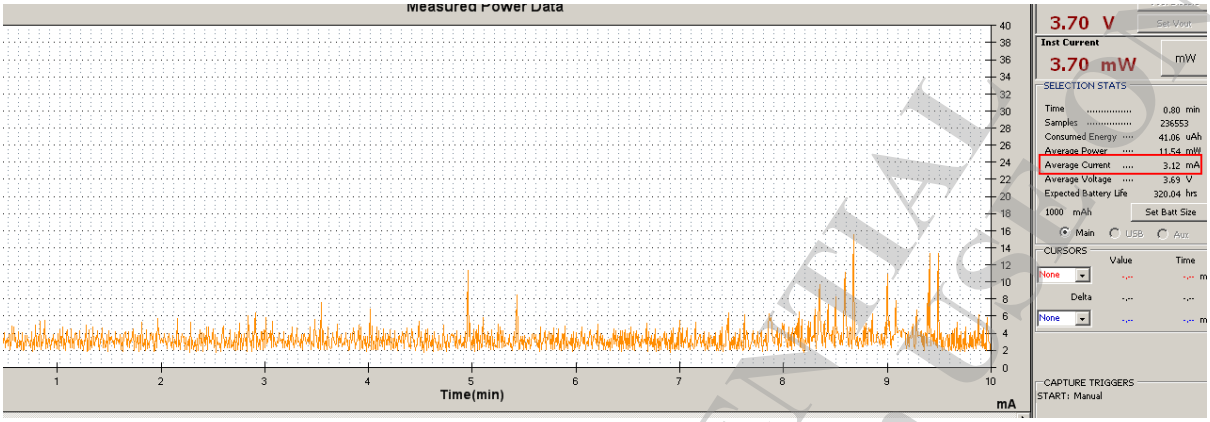
ciev signal 变化的预设值在 custom_l4_utility.c 中的 custom_ciev_signal_variance() 函数中定义.

解决方法:

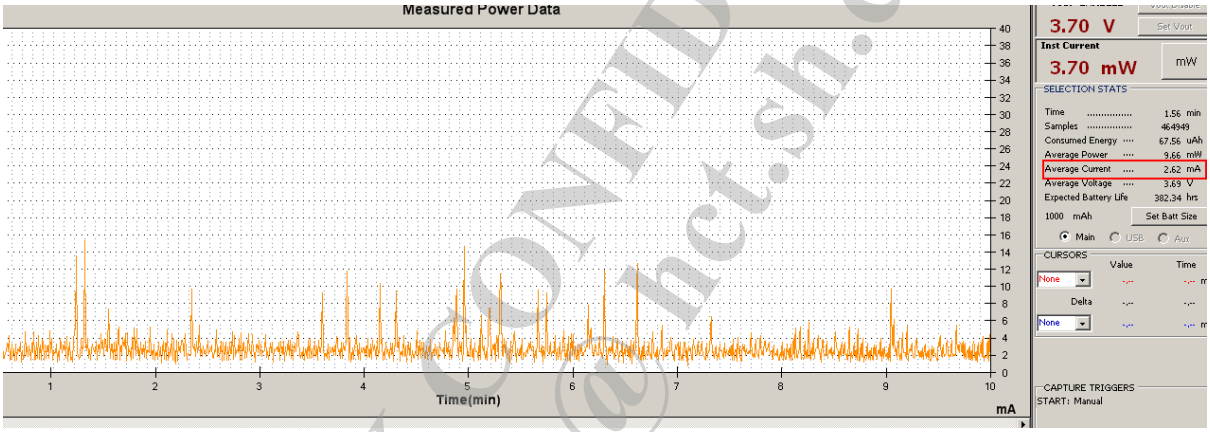
将该函数的 return 数值从 0 变为 4. 测试结果正常. 如下.



修改后, 测试结果 1: 平均电流 3.12mA



修改后, 测试结果 2: 平均电流 2.62mA



3.9 MT6261 Phone BT Dialer 版本连接蓝牙耳机后待机电流大

描述问题:

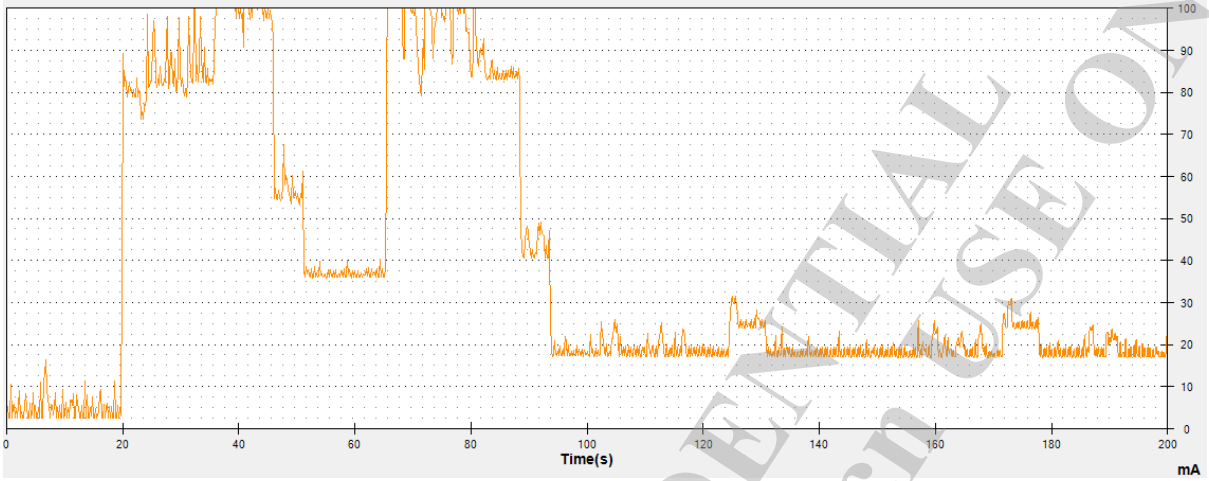
MT6261 MAUI.11C.W13.52.MP.V26 版本代码, 既有 Phone 功能, 也有 BT Dialer 功能.
在连接任意一款蓝牙耳机后, 平均电流 20-30mA.

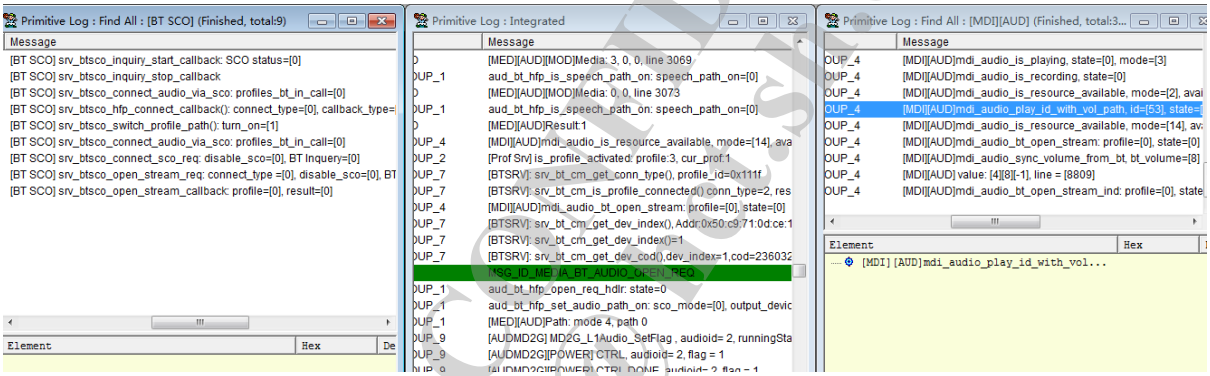
厘清问题:

该问题从 UART1 sleep log 一直没有看到有效信息.
sleep handler 在发生问题时, 有全部释放. 但是电流一直维持在较高准位.
进一步深入分析, 从 UART1 catcher log 及电流数据看到是 BT SCO 一直保持建立导致这种现象.
BT SCO 在建立时, 不会 lock 住 sleep handler, 所以从 sleep log 无法得到有效分析信息.
而是从大量的 log 中不断检查, 最后发现是 BT SCO 一直保持建立导致这种现象.

根本原因是 srv_btsco_connect_audio_via_sco() BthScoPatchSrv.c 调用 srv_btsco_connect_sco_req() 函数里会调用 srv_btsco_open_stream_req()建 SCO, 但是没有启动 timer 来断 SCO.

前 20S 是蓝牙打开待机的平均电流在 2-3mA.
后面是连接 Jabra 蓝牙耳机, 经过 100S 后灭屏, 电流稳定后, 平均电流维持在 20-30mA 之间.





解决方法:

修改如下:

BthScoPathSrv.c

```
//修改
static void srv_btsco_audio_via_sco_callback (void)
{
        srv_btsco_set_always_on(MMI_FALSE);
}
//修改


void srv_btsco_connect_audio_via_sco(void)
{
   /*----------------------------------------------------------*/
   /* Local Variables                                          */
   /*----------------------------------------------------------*/

   /*----------------------------------------------------------*/
   /* Code Body                                                */
```

```
    /*----------------------------------------------------------------*/
    MMI_TRACE(MMI_MEDIA_TRC_G2_APP,                SRV_BTSCO_CONNECT_AUDIO_VIA_SCO,
g_mmi_bt_sco_is_in_call);

    /* Press key or play audio in call will not connect SCO,
    *   call setup may leave the voice in phone */
#ifdef __BT_DIALER_SUPPORT__
    if(!g_mmi_bt_sco_is_in_call                                                    &&
srv_bt_cm_is_profile_connected(SRV_BT_CM_HFP_CONNECTION)
&&                  !srv_bt_cm_is_profile_connected(SRV_BT_CM_HSP_CONNECTION)    &&
srv_bt_cm_is_audio_path_to_headset())
#else
    if(!g_mmi_bt_sco_is_in_call                                                    &&
((srv_bt_cm_is_profile_connected(SRV_BT_CM_HFP_CONNECTION)                          ||
srv_bt_cm_is_profile_connected(SRV_BT_CM_HSP_CONNECTION))                          &&
srv_bt_cm_is_audio_path_to_headset()))
#endif
    {
        srv_btsco_connect_sco_req(srv_btsco_audio_via_sco_callback);//修改

    }
}
```

3.10 MT6261 BT Dialer 版本开启 CFG_MMI_BT_ANTI_LOST_BY_RSSI 防丢功能后平均电流 20mA

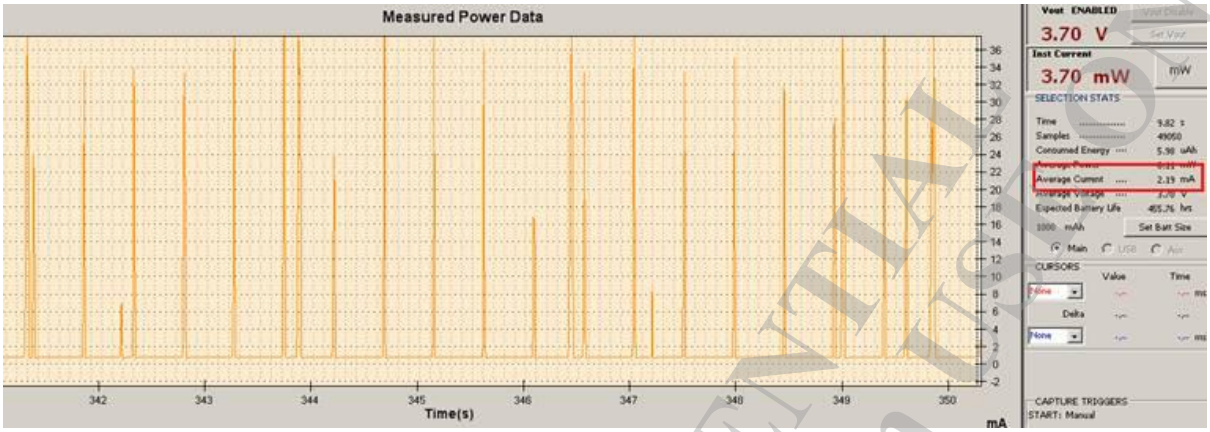
描述问题:

MT6261 BT Dialer 版本, 开启 CFG_MMI_BT_ANTI_LOST_BY_RSSI 功能, 当 BT Dialer 开启防丢功能后, 平均电流 20mA.

厘清问题:

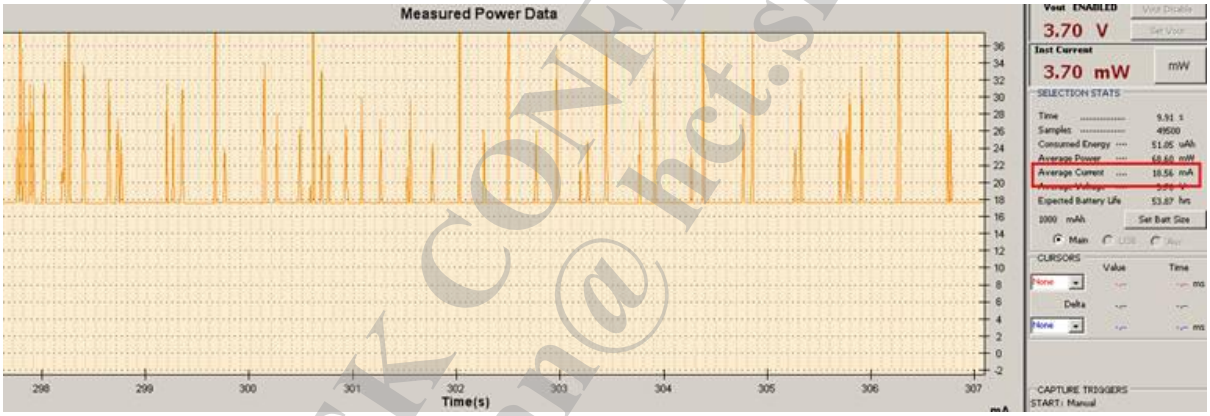
BT Dialer 版本代码工程中开启 CFG_MMI_BT_ANTI_LOST_BY_RSSI 功能, 当 BT Dialer 和 Android 智能机蓝牙连接, 不开启防丢功能, 电流数据.

平均电流: 2.19mA



代码工程中开启 CFG_MMI_BT_ANTI_LOST_BY_RSSI 功能, 当 BT Dialer 和 Android 智能机蓝牙连接, 开启防丢功能, 电流数据.

平均电流: 18.56mA



通过抓取 UART1 的 sleep log, 发现确实是 BT 部分 lock 住导致.

从 sleep log 看到 sleep handler 为 0xC0124(二进制: 1100 0000 0001 0010 0100), 其中包含 bit18(从 0 算起, 二进制: 0100 0000 0000 0000 0000 / 十六进制: 0x4000).

这证明 sleep handler 的 bit18 为 BT sleep 操作位.

130614	unlock	0x80124	0x0	10	0xF01E819B	mm enable power	N/A
130614	unlock	0x80124	0x0	12	0xF01F6D61	lcd power ctrl register module	N/A
130617	lock	0xC0124	0x0	18	0x1024F1B9	bt_init	N/A
130620	lock	0xC1524	0x0	10	0xF01E819B	mm enable power	N/A

从后面的 log 看到, 是 sleep handler 的 bit18 没有释放导致系统耗电.

148499	lock	0x41524	0x0	8	0xF01E3423	DclSGPT Initialize	N/A
148499	lock	0x41524	0x0	10	0xF01E819B	mm enable power	N/A
148499	lock	0x41524	0x0	12	0xF01F6D61	lcd power ctrl register module	N/A
148600	unlock	0x40024	0x0	8	0xF01E3423	DclSGPT Initialize	N/A
148600	unlock	0x40024	0x0	10	0xF01E819B	mm enable power	N/A
148600	unlock	0x40024	0x0	12	0xF01F6D61	lcd power ctrl register module	N/A
152272	lock	0x41524	0x0	8	0xF01E3423	DclSGPT Initialize	N/A
152272	lock	0x41524	0x0	10	0xF01E819B	mm enable power	N/A
152272	lock	0x41524	0x0	12	0xF01F6D61	lcd power ctrl register module	N/A
152374	unlock	0x40024	0x0	8	0xF01E3423	DclSGPT Initialize	N/A
152374	unlock	0x40024	0x0	10	0xF01E819B	mm enable power	N/A
152374	unlock	0x40024	0x0	12	0xF01F6D61	lcd power ctrl register module	N/A
152680	unlock	0x40004	0x0	5	0x1024D0D5	PWM_Init	N/A

代码工程中开启 CFG_MMI_BT_ANTI_LOST_BY_RSSI 功能, 当 BT Dialer 和 Android 智能机蓝牙连接, 开启防丢功能, 电流数据异常.
查看 CFG_MMI_BT_ANTI_LOST_BY_RSSI 控制的代码, 发现这个防丢功能是通过启动一个 5S 的定时器, 每隔 5S 从 BT 模块获取对方蓝牙的信号, 即 RSSI 数值, 判断这个 RSSI 的数值是否低于一个阈值, 如果低于这个阈值, 报警.
通过和 BT Stack / BT Controller 沟通, BT Controller 在获取 RSSI 数值后 12S 的时间内 BT Controller 是不会释放 unlock sleep.
只有过了 12S 后, BT Controller 会 unlock sleep. 所以每隔 5S 从 BT 模块获取对方蓝牙的信号是无法让 BT Controller unlock sleep.

解决方法:
增加从 BT 模块获取对方蓝牙信号的时间间隔, 调整为 60S.

BTMMIAntilostScr.c

#define MMI_BT_SLOW_DETECT_TIME (60)//修改

...省略
...省略
...省略

```
 void mmi_bt_cm_read_rssi_cnf_hdler(void* msg)
{

        /*----------------------------------------------------------------*/
        /* Local Variables                                                */
        /*----------------------------------------------------------------*/
        bt_bm_read_rssi_cnf_struct* read_rssi_ind = (bt_bm_read_rssi_cnf_struct*)msg;

        /*----------------------------------------------------------------*/
        /* Code Body                                                      */
        /*----------------------------------------------------------------*/
    MMI_TRACE(MMI_CONN_TRC_G7_BT, MMI_BT_ANTILOST_RSSI_CNF);

        if(read_rssi_ind->result == 0)
        {
                MMI_TRACE(MMI_CONN_TRC_G7_BT,    MMI_BT_ANTILOST_RSSI,    read_rssi_ind-
>rssi_value, BT_ANTI_LOST_CFG_RSSI_ALERT_THRESHOLD);

                if(read_rssi_ind->rssi_value < BT_ANTI_LOST_CFG_RSSI_ALERT_THRESHOLD)
                {
            mmi_frm_nmgr_notify_by_app(
                MMI_SCENARIO_ID_HIGH_SCRN,
                MMI_EVENT_WARNING,
```

```
            mmi_bt_antilost_nmgr_callback,
            NULL);

        mmi_bt_anti_lost_stop_timer();
             }
      else
      {
        if (mmi_frm_group_is_present(GRP_ID_BT_ANTI_LOST_IND))
              {
                    mmi_bt_anti_lost_ind_reset();
          return;
              }

        //mmi_bt_anti_lost_restart_timer();//修改
        if(mmi_bt_anti_lost_get_status() == MMI_BT_ANTI_LOST_ON)//修改
        {   //修改
          mmi_bt_anti_lost_restart_timer();//修改
        }//修改

      }

       }

       return ;
}
...省略
...省略
...省略
```

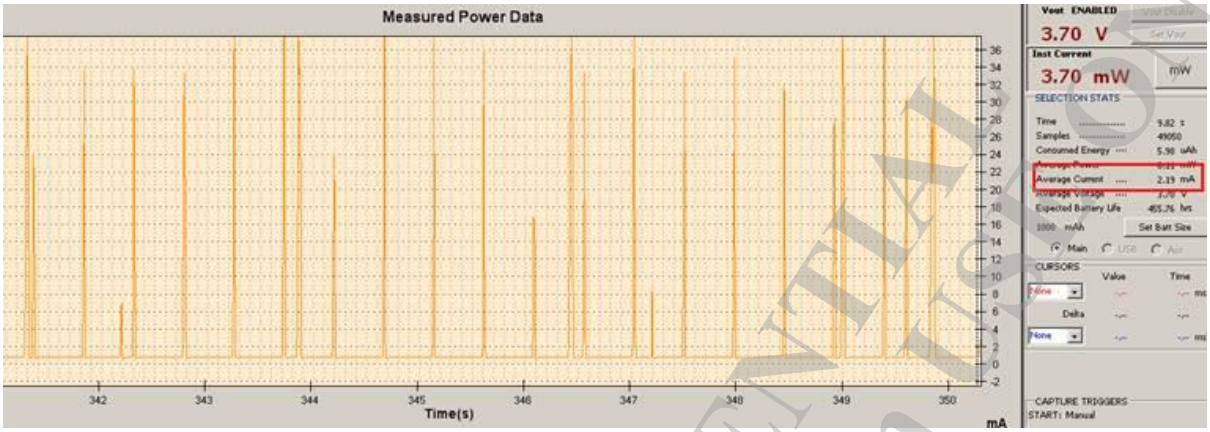
BTMMIAntilostScr.h

```
#define BT_ANTI_LOST_CFG_QUERY_INTERVAL 60//修改
```

修改后测试电流数据如下:

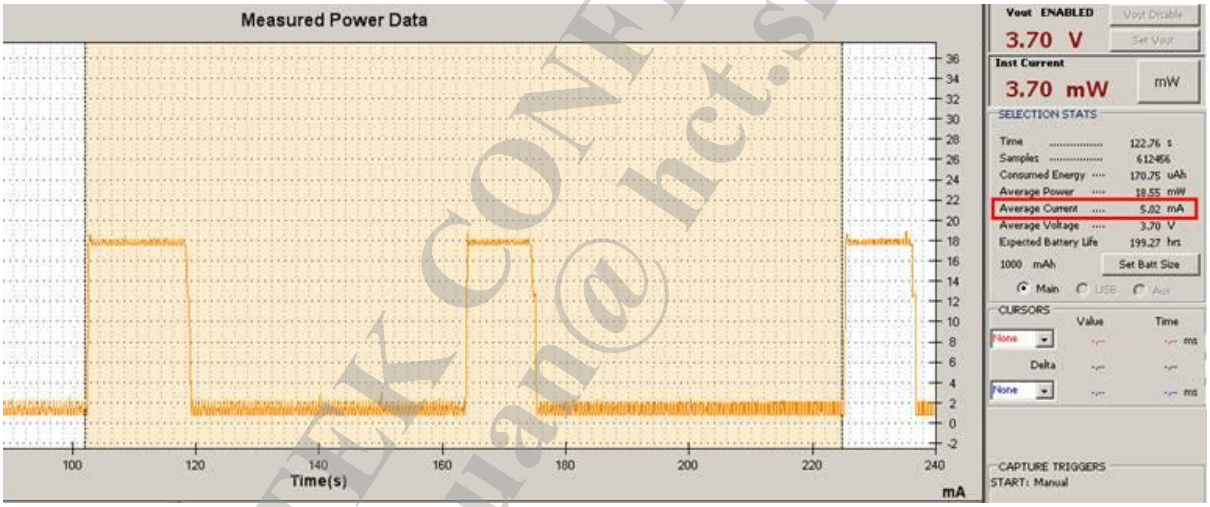
1, 代码工程中开启 CFG_MMI_BT_ANTI_LOST_BY_RSSI 功能, 当 BT Dialer 和 Android 智能机蓝牙连接, 不开启防丢功能, 电流数据.

平均电流: 2.19mA



2, 代码工程中开启 CFG_MMI_BT_ANTI_LOST_BY_RSSI 功能, 当 BT Dialer 和 Android 智能机蓝牙连接, 开启防丢功能, 电流数据.

平均电流: 5.02mA



4 BT 死机问题

4.1 MT2502 打开蓝牙后主板重启

较早参考设计:

双路 buck.

VDDK - > BUCK_EN 控制 -> External Buck => 绿色框图.

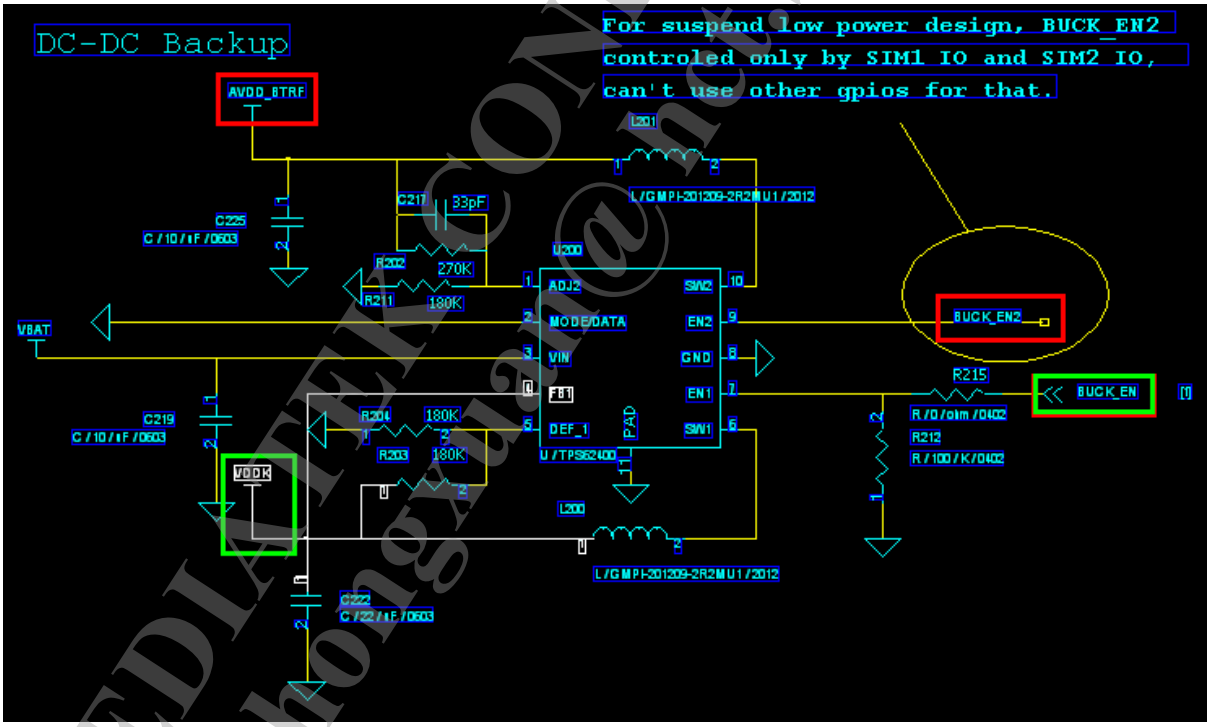
AVDD_BTRF -> BUCK_EIN2 控制 -> BT RF power in => 红色框图.

最新参考设计:

单路 buck.

VDDK - > BUCK_EN 控制 -> External Buck => 绿色框图.

AVDD_BTRF -> always VIO28 供电.



目前 MT2502 最新 reference design 上是单路 buck.

如有还使用双路 buck, 请改回单路 buck.

4.2 MT2502 Aster V1 版本在使用蓝牙过程中出现极低概率死机

MT2502 的代码包是以 11CW1418SP4 开头, 在代码包名称中, V15 以下(不含 V15)版本为 MT2502 Aster V1 版本.

在使用蓝牙过程中, 极低概率出现蓝牙死机问题. 通过死机之后抓取的 memory dump 看到如下信息.

调用栈: SP: 0xF00FB9F8, PC: 0x10069EDD

kal_assert_fail_internal(参数 1:0xF02243E8 = uaxprt + 0x60, 参数 2:0xF02243E0 = uaxprt + 0x58, 参数 3:0x1393A = med_aud_mem + 0x1662, 参数 4:0)

CheckTimers(参数 4:0x3EF3 = gFS_BufferData + 0x1EB3)

EVM_Process()

bt_main(参数 1:0xF00FBB70 = System_Mem_Pool + 0x35FAC)

bt_task_main(参数 1:0xA800A8, 参数 2:0x3EF3009F, 参数 3:0, 参数 4:0)

kal_prepare_stack_to_run(参数 1:0xF010EE30 = System_Mem_Pool + 0x4926C)

TCC_Task_Shell()

解决方法:

Patch ID: MAUI_03542150

4.3 MT2502 手表与手机蓝牙(BT/BLE)连接后播放音乐过程中死机

描述问题:

MT2502 手表与 iPhone/Android 手机蓝牙连接成功后, 一直播放蓝牙音乐, 20 分钟左右, MT2502 手表出现死机.

不安装 “联发设备宝” IPA, 测试两台 iPhone 手机, 正常, 没发现有死机.

安装 “联发设备宝” iPhone 手机 BR/EDR(A2DP) / BLE…应用场景连上, 播放蓝牙音乐会出现死机.

厘清问题:

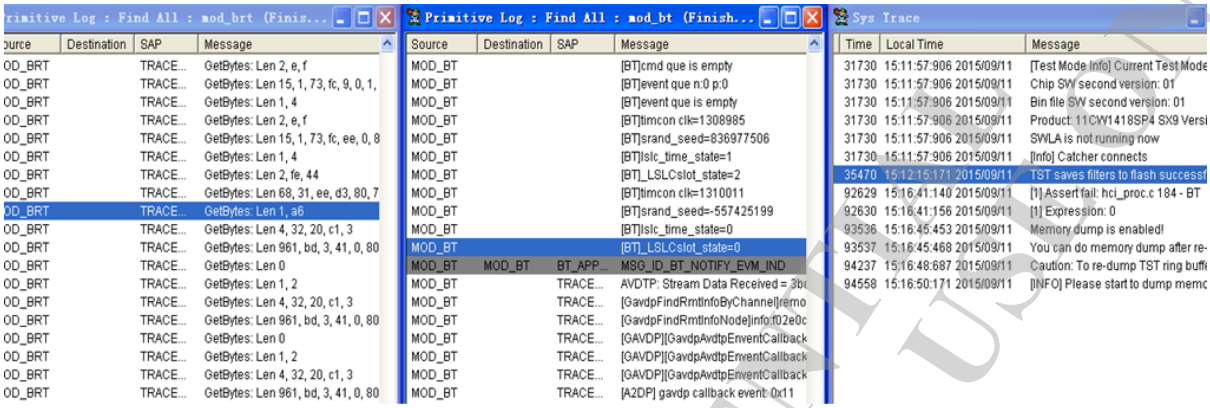
出现异常在 Assert fail: hci_proc.c 184 – BT

Call stack 看到是 BT stack 收到 BT controller 发的 UART data 之后, 在 HciTransportError1()函数内 ASSERT(0)掉.

Call stack 中显示 HciTransportError1()函数的入参为 11, 应该是在 uartReadData()中执行到 BT_PANIC_INVALID_HCI_BUF_TYPE 这里.

Air Sniffer Log 发现 BR/EDR 的数据和 BLE 的数据在收发时有碰撞.

```
== 进程检查 ==
TASK:   BT (执行中, 优先级:0, 外部队列共50条, 栈最大使用100%)
调用栈: SP: 0xF0119C60, LR: 0x10284E6D, PC: 0x101D3F7F
        kal_assert_fail_native(参数1:0x10284E84, 参数2:0x10284E78, 参数3:184)
        HciTransportError1(参数1:11)
        uartReadData()
        uartEventHandler()
        bt_main(参数1:0xF0119DE0 = System_Mem_Pool + 0x24C98)
        bt_task_main(参数1:0xAB005A, 参数2:0x58A10084, 参数3:0, 参数4:0)
        kal_prepare_stack_to_run(参数1:0xF0129624 = System_Mem_Pool + 0x344DC)
        TCC_Task_Shell()
        == 栈结束 ==
```



解决方法:

Patch ID: MAUI_03544865