泰凌 BT/BLE Controller SDK 开发手册

功能描述

AN-21122301-C1

Ver.1.0.0

2021/12/23

关键词

蓝牙，蓝牙低功耗，Controller，SDK

摘要

本文为泰凌 BT/BLE 双模 Controller SDK 的功能描述。

Published by

Telink Semiconductor

Bldg 3, 1500 Zuchongzhi Rd,

Zhangjiang Hi-Tech Park, Shanghai, China

Legal Disclaimer

This document is provided as-is. Telink Semiconductor reserves the right to make improvements without further notice to this document or any products herein. This document may contain technical inaccuracies or typographical errors. Telink Semiconductor disclaims any and all liability for any errors, inaccuracies or incompleteness contained herein.

Information

For further information on the technology, product and business term, please contact Telink Semiconductor Company (www.telink-semi.com).

For sales or technical support, please send email to the address of:

telinksales@telink-semi.com

telinksupport@telink-semi.com

更改历史

版本	更改内容	日期	作者
V1.0.0	初版	2021/12	Peng.QI, C.LIU

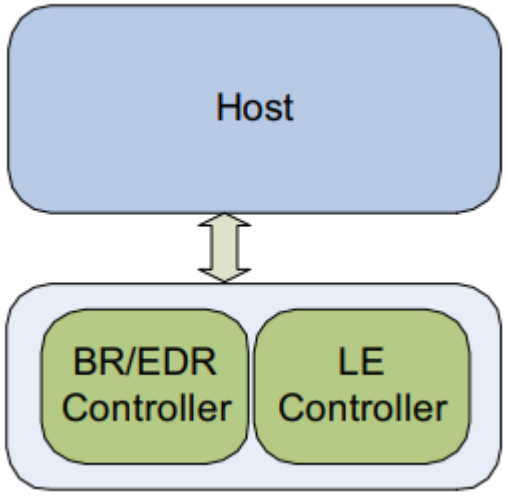
目录

1. 前言

1.1 文档描述

本文主要是对泰凌 BT/BLE 双模 controller SDK 功能的描述，通过 HCI 接口来适配不同的 host,支持标准的 HCI 协议和 HCI 流控，硬件上支持 UART 和 USB 模块作为传输接口。

图 1-1 Host 和 Controller 间的传输



	仅 LE	仅 BT	BT/BLE
模式	√	√	√

1.2 SDK 架构总览

图 1-2 SDK 架构总览

main.c

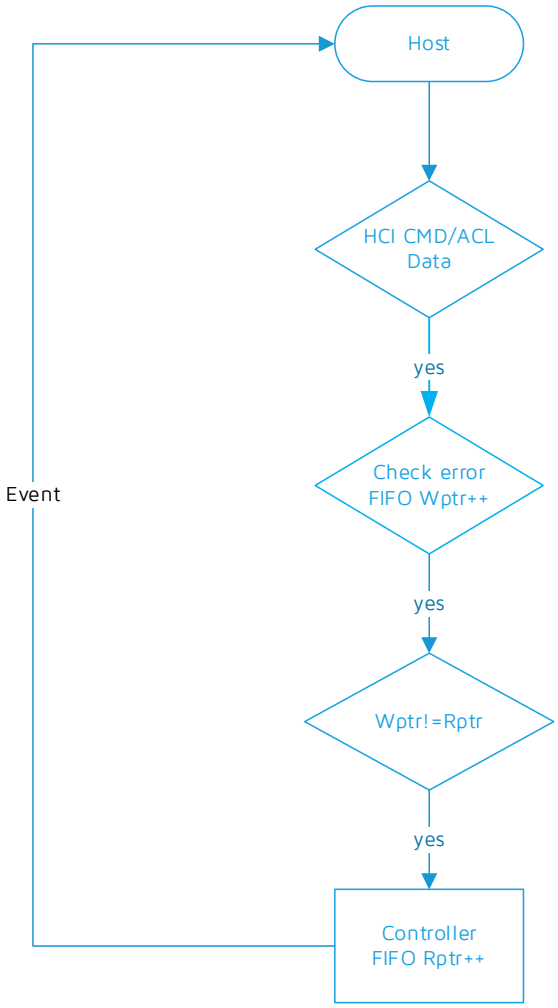
```
{
    sys_init(DCDC_1P4_LDO_1P8,VBAT_MAX_VALUE_GREATER_THAN_3V6);

    clock_init(PLL_CLK_192M, PAD_PLL_DIV, PLL_DIV4_TO_CCLK, CCLK_DIV2_TO_HCLK, HCLK_DIV1_TO_PCLK, CCLK_TO_MSPI_CLK);
    /* random number generator must be initiated here( in the beginning of user_init_nromal).
     * When deepSleep retention wakeUp, no need initialize again */
    random_generator_init();  //this is must

    intcntl_init();

    io_debug_init();

#if (ENABLE_PRINT||ENABLE_VCD)
    debug_init();
#endif

#if(ENABLE_BT_CLASSIC||ENABLE_LE)
    hci_transport_init(HCI_TRANSPORT_MODE);
#endif

#if(ENABLE_BT_CLASSIC||ENABLE_LE)
    btble_schedule_init();
#endif

#if ENABLE_LE
    ble_init();
#endif

#if ENABLE_BT_CLASSIC
    bt_init();
#endif

    user_init();

    core_enable_interrupt();

    while (1) {
        main_loop();
```

- sys_init：系统初始化，配置系统供电模式，电压等级
- clock_init：时钟初始化
- random_generator_init：随机数生成模块初始化
- intcntl_init：中断表初始化
- io_debug_init：gpio 调试初始化，通用 gpio 功能，用户可以自己配置
- debug_init：系统调试初始化，开发者可以用来打印 log

- hci_transport_init：HCI 模块初始化，用来配置传输的方式
- btble_schedule_init：系统调度初始化，系统用来调度各种任务的运行
- ble_init：ble 模块初始化
- bt_init：bt 模块初始化
- user_init：预留给开发者使用的模块
- core_enable_interrupt：经过前面全部的初始化后，使能全局中断，系统便可以进入运行状态
- main_loop：系统循环，在这里处理各类的任务。

1.3 HCI 流程总览

图 1-3 HCI 流程



2. Controller 配置

2.1 LE Link 配置

这部分是关于 LE 部分的配置，是否使能 LE，以及 LE 作为 master 和 slave 时连接个数。

```
/* LE */
#define ENABLE_LE                       1
#define MASTER_MAX_NUM                  2
#define SLAVE_MAX_NUM                   1
#define BLE_MAX_CONN_NUM                (MASTER_MAX_NUM + SLAVE_MAX_NUM)
```

若完全禁用 LE 的功能，将 ENABLE_LE 置为 0 即可。

2.2 BT Link 配置

这部分是关于 BT 部分的配置，注意如果只需一路 BT 连接，BT_ACL_LINK_NUM 设置为 1 即可。

```
/* CLASSIC BLUETOOTH*/
#define ENABLE_BT_CLASSIC               1


#define BT_ACL_LINK_NUM                 1
```

BT 模块包含的功能模块采用注册模式，客户可根据自己的需求，是否选择注册该功能。

图 2-1 BT Link 配置

*app_bt.c stack_config.h

```
102
103 void btc_module_init(void)
104 {
105     // AFH
106     btc_afh_register_module(&env_afh[0]);
107     // AFH CLS
108     btc_afh_cls_register_module();
109     // ROLE SWITCH
110     btc_rsw_register_module(&env_rsw[0]);
111
112     // LEGACY PAIR
113     btc_legacy_pair_register_module();
114     // SECURE SIMPLE PAIR
115     btc_ssp_register_module(&env_ssp[0]);
116     // LEGACY AUTHEN
117     btc_legacy_authen_register_module();
118     // SECURE AUTHEN
119
120     // ENCRYPTION
121     btc_enc_register_module(&env_enc[0]);
122     // ESCO
123 //    btc_sco_register_module(&env_sco[0]);
124     // SNIFF
125 //    btc_sniff_register_module(&env_sniff[0]);
126     // LOW POWER
127
128     //TEST MODE
129 //    btc_test_mode_register_module(&env_test_mode[0]);
130 }
131
132
```

2.3 HCI 传输配置

这部分是关于 HCI 部分的配置，用户选择使用哪种方式作为传输。

```
/* HCI TRANSPORT*/

#define ENABLE_HCI_TRANSPORT            1

#define HCI_TRANSPORT_MODE              2//1-usb bulk,2-uart
```

2.3.1 UART 传输配置

关于 UART 作为 HCI 传输的配置，要求 host 端的对接设备必须支持 UART 的流控。

包含波特率配置，端口配置，pin 脚配置。

图 2-2 UART 传输配置

hci_transport.h

```
21
22 ////////////////////  FLOW CONTROL //////////////////////////////////
23 #define UART_FLOW_CTR                0
24 #define CTS_STOP_VOLT                1 //0 :Low level stops TX.  1 :High level stops TX.
25 #define UART_MANUAL_FLOW_CTR_RTS_STOP               gpio_set_high_level(UART_RTS_PIN)
26 #define UART_MANUAL_FLOW_CTR_RTS_START              gpio_set_low_level(UART_RTS_PIN)
27
28 //baudrate of UART
29 #define UART_BAUDRATE   115200
30 /*! HCI Transport configuration  */
31 #if 0//UART0
32 #define UART_ID          UART0
33 #define UART_TX_PIN      UART0_TX_PD2
34 #define UART_RX_PIN      UART0_RX_PD3
35 #define UART_CTS_PIN     UART0_CTS_PA1
36 #define UART_RTS_PIN     UART0_RTS_PA2
37 #define UART_IRQ         IRQ19_UART0
38 #define UART_IRQHandler uart0_irq_handler
39
40 #else
41 #define UART_ID          UART1
42 #define UART_TX_PIN      UART1_TX_PE0
43 #define UART_RX_PIN      UART1_RX_PE2
44 #define UART_CTS_PIN     UART1_CTS_PE1
45 #define UART_RTS_PIN     UART1_RTS_PE3
46 #define UART_IRQ         IRQ18_UART1
47 #define UART_IRQHandler uart1_irq_handler
48
```

2.3.2 USB 传输配置

USB 的配置如下。注意该模式下的打印方式只能使用 UART debug 方式。

```
#define ENABLE_HCI_TRANSPORT          1

#define HCI_TRANSPORT_MODE            1//1-usb bulk,2-uart
```

3. 调试和打印

3.1 USB 打印 log

该模式需要配合 Telink 专有软件使用，在 USB HCI 模式下有冲突，不可以使用该模式，只能在 UART HCI 模式下使用，演示如下：

```
#define PRINT_MODE                  0 //usb 打印模式
```

图 3-1 USB 打印 log

```
17:54:27.807  <53> HCI-> Event:         04 0e 09 01 27 0c 00 11  00 00 00 00
17:54:28.241  <54> <<<_CMD_CMPT>        04 070e         01 05 14 00 11 00 00
17:54:28.242  <55> dma_offset&readmasize:       04 00 00 00 ff 03 00 00  02 00 00 00 00 00 00 00
17:54:28.242  <56> fifo_hci_data :
        06 00 00 00 01 05 14 02  11 00 00 00 00 00 00 00
        00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
        00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
        00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
17:54:28.242  <57> packet_len :         06 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
17:54:28.243  <58> fifo_hci_data_p :
        00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
        00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
        00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
        00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
17:54:28.243  <59> complete_hci_data from_host:
17:54:28.243  <60> <<<_CMD_CMPT>        04 090e         01 27 0c 00 11 00 00 00  00
17:54:28.243  <61> @ bt-->hci_exec data(8)_id(4)_fucntion:      01 05 14 02 11 00 00 00  f5 00 00 00 68
17:54:28.244  <62> @ HCI_RD_RSSI_CMD_OPCODE_1405_f5:    01 05 14 02 11 00
17:54:28.244  <63> HCI-> Event:         04 0e 07 01 05 14 00 11  00 00
17:54:28.244  <64> dma_offset&readmasize:       04 00 00 00 ff 03 00 00  02 00 00 00 00 00 00 00
17:54:28.244  <65> fifo_hci_data :
        06 00 00 00 01 27 0c 02  11 00 00 00 00 00 00 00
        00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
        00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
        00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
17:54:28.244  <66> packet_len :         06 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
17:54:28.245  <67> fifo_hci_data_p :
        00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
        00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
        00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
        00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
17:54:28.245  <68> complete_hci_data from_host:
17:54:28.245  <69> @ bt-->hci_exec data(8)_id(4)_fucntion:      01 27 0c 02 11 00 00 00  83 00 00 00 64
17:54:28.245  <70> @ HCI_RD_AUTO_FLUSH_TO_CMD_OPCODE_0c27_83:   01 27 0c 02 11 00
17:54:28.245  <71> HCI-> Event:         04 0e 09 01 27 0c 00 11  00 00 00 00
```

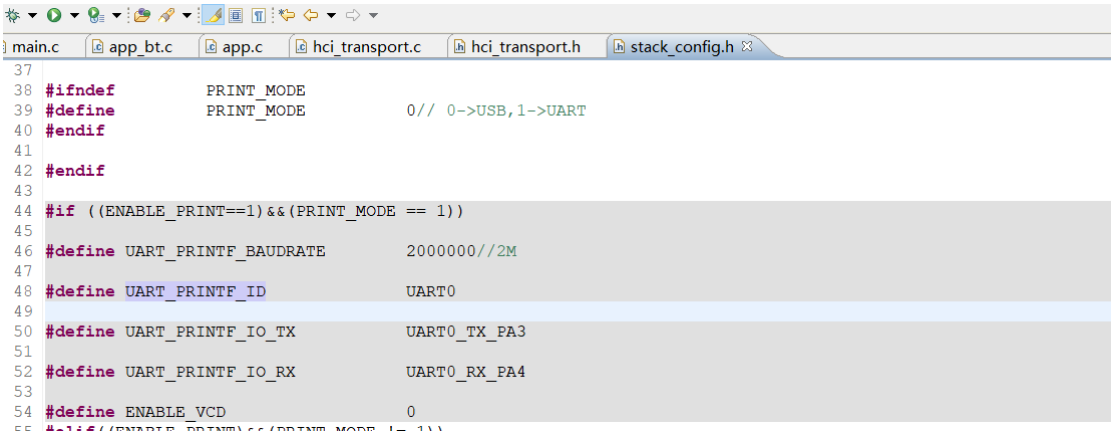
3.2 UART 打印 log

该打印 log 模式，可以在 UART/USB 的 HCI 下均可以使用

```
#define PRINT_MODE                  1//uart 打印模式
```

该 debug 方式使用了 UART 的 DMA 方式，用户可配置该模块引脚以及波特率

图 3-2 UART 打印配置



需要注意的是当使用 UART HCI 时，切勿将 UART debug 的引脚配置成和 UART HCI 一样。

图 3-3 UART 打印 log

```
@ bt—>hci_exec data(8)_id(4)_fucntion:01 03 0c 00 00 00 00 00 5f 00 00 00 e0 7e 02 20
HCI_RESET_CMD_OPCODE_0c03_5f:01 03 0c 00|
pscan_win:40 00
pscan_intv:ac 00
HCI-> Eent:04 0e 04 01 03 0c 00
dma_offset&readmasize:04 00 00 00 ff 03 00 00 00 00 00 00 00 00 00 00
fifo_hci_data :04 00 00 00 01 05 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 
00 00 00 00 00 00 00 00 00 00 00
packet_len :04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
fifo_hci_data_p :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0
00 00 00 00 00 00 00 00 00 00 00 00
complete_hci_data from_host:
@ bt—>hci_exec data(8)_id(4)_fucntion:01 05 10 00 00 00 00 00 e5 00 00 00 ec 69 02 20
@ HCI_RD_BUFF_SIZE_CMD_OPCODE_1005_e5:01 05 10 00
HCI-> Event:04 0e 0b 01 05 10 00 bc 02 40 04 00 08 00
dma_offset&readmasize:08 00 00 00 fe 03 00 00 03 00 00 00 00 00 00 00
fifo_hci_data :0b 00 00 00 01 33 0c 07 9b 06 ff 14 00 0a 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 
00 00 00 00 00 00 00 00 00 00 00
packet_len :0b 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
fifo_hci_data_p :00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0
00 00 00 00 00 00 00 00 00 00 00 00
complete_hci_data from_host:
@ bt—>hci_exec data(8)_id(4)_fucntion:01 33 0c 07 9b 06 ff 14 8f 00 00 00 30 76 02 20
@ HCI_HOST_BUF_SIZE_CMD_OPCODE_0c33_8f:01 33 0c 07 9b 06 ff 14 00 0a 00
HCI-> Event:04 0e 04 01 33 0c 00
dma_offset&readmasize:04 00 00 00 ff 03 00 00 00 00 00 00 00 00 00 00
fifo_hci_data :04 00 00 00 01 01 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 
00 00 00 00 00 00 00 00 00 00 00
packet_len :04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
fifo_hci_data_p :00 00 00 00 00 0a 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0
00 00 00 00 00 00 00 00 00 00 00 00
complete_hci_data from_host:
@ bt—>hci_exec data(8)_id(4)_fucntion:01 01 10 00 00 00 00 00 e1 00 00 00 78 6b 02 20
@ HCI_RD_LOCAL_VER_INFO_CMD_OPCODE_1001_e1:01 01 10 00
HCI-> Event:04 0e 0c 01 01 10 00 0b 0c 00 0b 11 02 0b 00
dma_offset&readmasize:04 00 00 00 ff 03 00 00 00 00 00 00 00 00 00 00
fifo_hci_data :04 00 00 00 01 09 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 
00 00 00 00 00 00 00 00 00 00 00
packet_len :04 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
fifo_hci_data_p :00 00 00 00 00 0a 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0
00 00 00 00 00 00 00 00 00 00 00 00
complete_hci_data from_host:
@ bt—>hci_exec data(8)_id(4)_fucntion:01 09 10 00 00 00 00 00 e9 00 00 00 90 69 02 20
@ HCI_RD_BD_ADDR_CMD_OPCODE_1009_e9:01 09 10 00
HCI-> Event:04 0e 0a 01 09 10 00 2e c5 c9 60 b8 03
dma_offset&readmasize:04 00 00 00 ff 03 00 00 00 00 00 00 00 00 00 00
```

4. 用户自定义 HCI CMD 功能

该 SDK 预留了 callback 接口，来供实现用户自己的 HCI 自定义功能，需要按照 0CF 值注册对应自定义功能。如下表：

表格 4-1 0CF 对应表

0GF	0CF
0x3F	0x08
	0x09
	0x0a
	0x0b

演示：

当返回值为 0，注册正确，否则注册失败。

```
int hci_vendor_spec_register_cb_port(hci_cmd_vendor_spec_callback_t cb,u8 vs_opcode_num);
```

图 4-1 注册演示

app.c

```
47 }
48 void my_vs_test_func(u8 *para)
49 {
50     gpio_toggle(GPIO_D);
51 }
52
53
54 void user_init(void)
55 {
56     hci_vendor_spec_register_cb_port(my_vs_test_func,0x08);
57 }
```