PAN1080 BLE HCI 使用说明

PAN-CLT-VER-A0, Rev 1.1

PanchipMicroelectronics

www.panchip.com

修订历史

版本	修订日期	描述
V1.0	2023-09-28	初始版本创建

目录

目录

第1章 测试目的

1. 测试 PAN1080 BLE HCI 接口功能。

第2章 测试内容

2.1 测试内容

1. 此项目演示了通过 HCI 接口发送广播扫描命令：将接收到的广播数据通过串口打印出来。

2.2 环境配置

a) 环境要求

- board: pan108xxb5_evb
- uart: 显示串口输出 log
- PC 串口工具：Panchip Serial Assistant V0.0.009.exe

b) 编译和烧录

项目位置：“03_MCU\mcu_samples\BLE_HCI”。
用 keil 编译程序，用 j-link 烧录编译后的 hex 文件到 pan108xxb5_evb 板子中。

第3章 测试说明

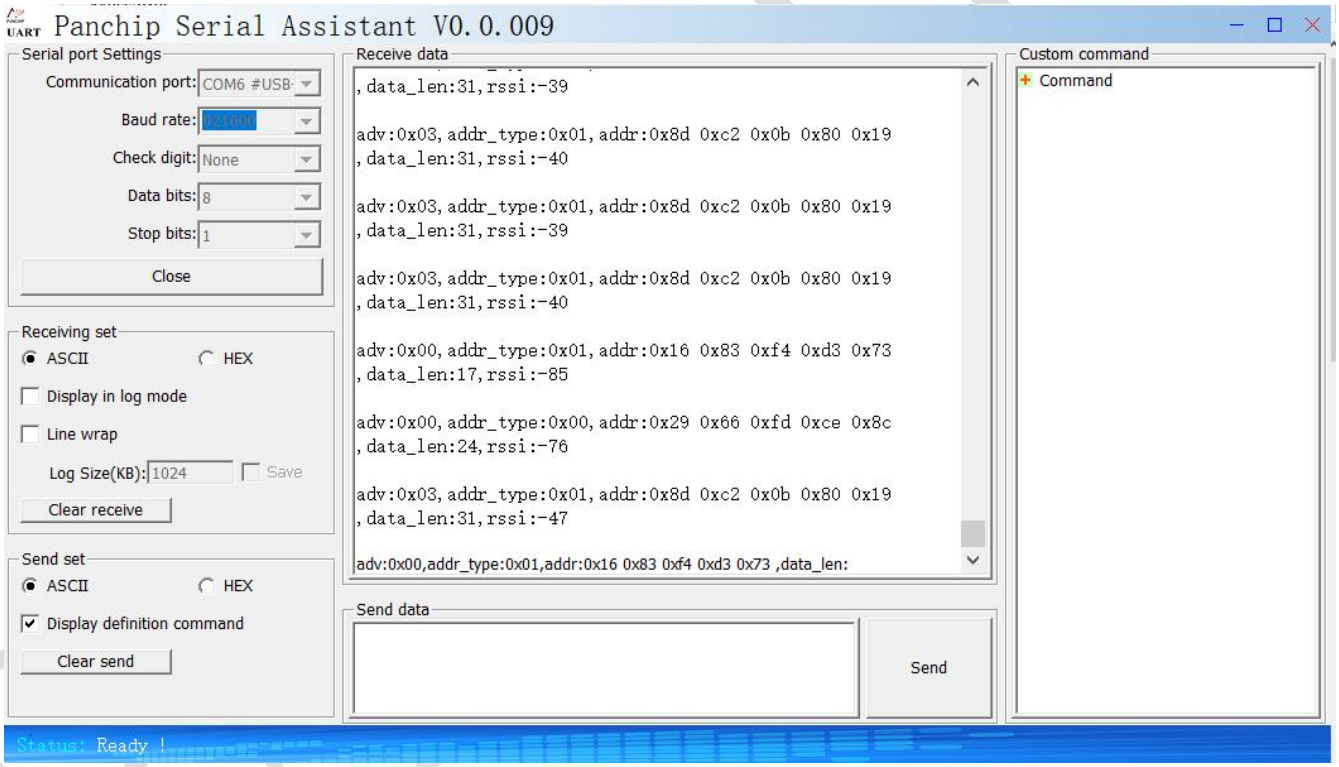
3.1 环境说明

a) 将 EVB 串口接到 PC 的 USB 端口上。

b) 观察 PC 串口工具的输出结果。

3.2 测试结果

1. 发射端输出结果：



3.3 开发说明

3.3.1 BLE HCI 初始化

设置接收频点 2450Mhz

```
1.      hci_eventq_init();    /* hci 上报事件队列初始化*/
2.      nimble_port_init();   /* ble 协议栈初始化*/
3.      ble_hs_startup_reset_tx();   /*hci 发送 reset 命令*/
4.      hci_set_event_mask();   /*hci 设置 controller 处理的事件*/
5.      hci_set_scan_params    /*hci 命令设置扫描参数*/
6.      hci_set_scan_enable    /*hci 扫描使能*/
```

3.3.2 BLE 广播事件处理

1. HCI 处理事件的线程

   因为 HCI 事件是异步上报的，我们单独使用了一个 HCI 线程进行处理异步事件。

```
7.  static void
8.  ble_hs_event_rx_hci_ev(struct ble_hci_ev *hci_ev)
9.  {
10.     ble_hs_hci_evt_process(hci_ev);
11. }
12.
13. void hci_thread_entry(void *parameter)
14. {
15.     BaseType_t ret;
16.     struct ble_hci_ev *ev;
17.
18.       while(1)
19.     {
20.         ret = xQueueReceive(hci_eventq, &ev, 0);
21.         assert(ret == pdPASS || ret == errQUEUE_EMPTY);
22.
23.         if( ret == pdPASS )
24.         {
25.             ble_hs_event_rx_hci_ev((struct ble_hci_ev *)ev);
26.         }
27.     }
28. }
```

2. 广播事件的 hci 处理函数 `ble_hs_hci_evt_le_adv_rpt`

```
1)  static int
2)  ble_hs_hci_evt_le_adv_rpt(uint8_t subevent, const void *data, unsigned int len)
3)  {
```

```
4)      const struct ble_hci_ev_le_subev_adv_rpt *ev = data;
5)      const struct adv_report *rpt;
6)      int rc;
7)      int i;
8)
9)      const uint8_t * iter = (const uint8_t *)data;
10)
11)     /* Validate the event is formatted correctly */
12)     rc = ble_hs_hci_evt_le_adv_rpt_first_pass(iter, len);
13)     if (rc != 0) {
14)         return rc;
15)     }
16)
17)     iter += sizeof(*ev);
18)
19)     for (i = 0; i < ev->num_reports; i++) {
20)         rpt = (const struct adv_report *)iter;
21)
22)         iter += sizeof(rpt) + rpt->data_len + 1;
23)
24)         printf("adv:0x%02x,addr_type:0x%02x,",rpt->type,rpt->addr_type);
25)         printf("addr:");
26)         for(uint8_t i; i<6; i++)
27)         {
28)             printf("0x%02x ",rpt->addr[i]);
29)         }
30)         printf(",");
31)         printf("data_len:%d,rssi:%d\n",rpt->data_len,(int8_t)rpt->data[rpt->data_len]);
32)         printf("\n");
33)     }
34)
35)     return 0;
36) }
```

对应的 HCI 处理命令：

[BLE_HCI_LE_SUBEV_ADV_RPT] = ble_hs_hci_evt_le_adv_rpt,

我们参考此函数，对自己感兴趣的 HCI 事件函数做自定义的实现。