

AT32F423 USB HOST CCID DEMO

示例目的

演示使用USB OTG作为主机接 CCID 设备的demo。

注：本示例代码是基于雅特力提供的V2.x.x板级支持包（BSP）而开发，对于其他版本的BSP，需要注意使用上的区别。

支持型号列表：

支持型号	具备 USB Host 型号
------	----------------

1 快速使用方法

1.1 硬件资源

- 1) AT-START开发板 (AT32F423)
- 2) 外部USB ccid装置 (智能卡片阅读机)

1.2 软件资源

- 1) SourceCode
 - ccid_demo源程序
 - AT32驱动库
- 2) Doc
 - SC0126_AT32F423_USB_HOST_CCID_DEMO

注意: 所有project都是基于keil 5而建立, 若用户需要在其他编译环境上使用, 请参考BSP中templates工程中各种编译环境 (例如IAR6/7, keil 4/5) 进行简单修改即可。

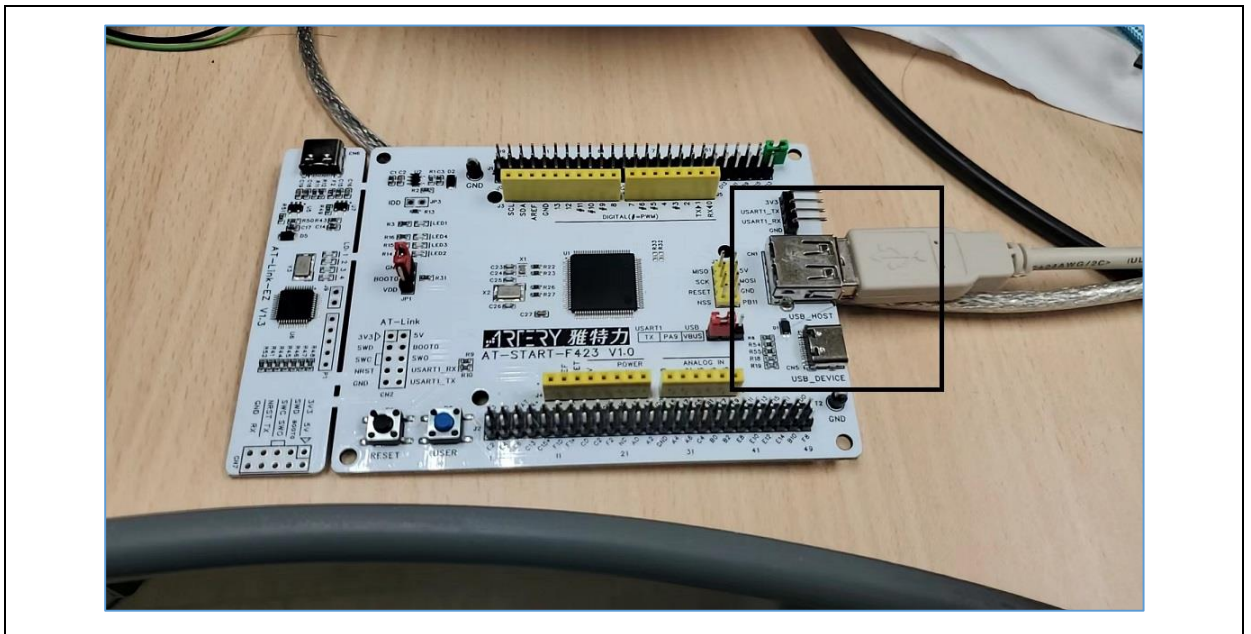
1.3 示例使用

软件使用:

打开对应的ccid_demo工程, project\at_start_f423\examples\otg_host\ccid_demo

- 编译通过之后下载到开发板
- 将USB ccid装置接到开发板的USB host接口。

图 1. 接到开发板的 USB host 接口



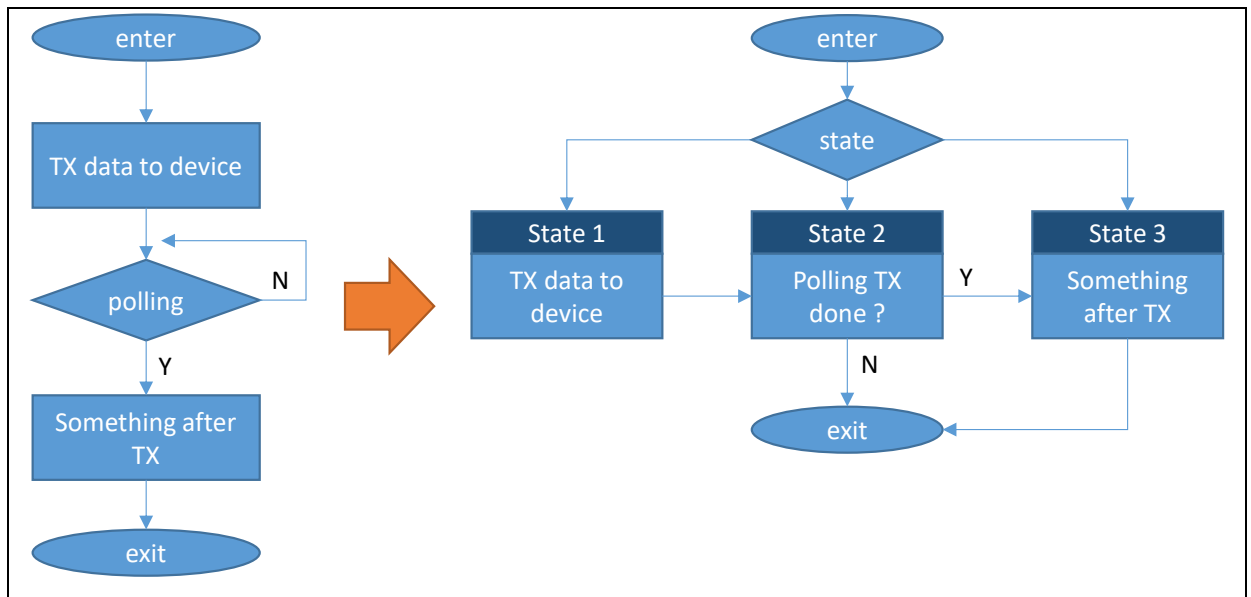
1.4 示例说明

1.4.1 注意事项

USB示例中所有的代码流程都必须是non-block的方式, 以状态机和轮询的方式取代在代码中的循环

等待。

图 2. 循环修改成状态机轮询示意图

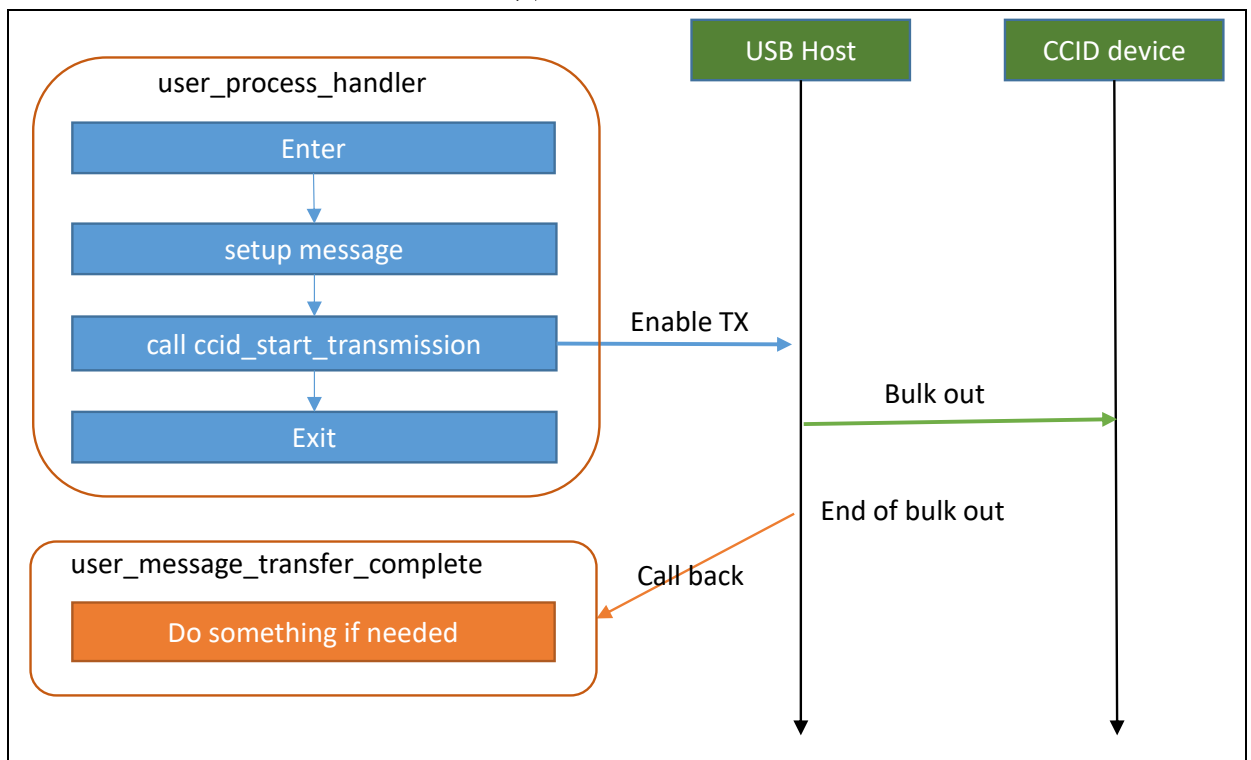


1.4.2 USB bulk-out flow

USB host需要对CCID装置透过bulk端点传输命令等数据，本章节将说明这部分的代码流程。

如下图，用户在准备好资料后，呼叫ccid_start_transmission。当host端完成数据的传送后，在示例中会呼叫user_message_transfer_complete。此时使用者可以在此添加想要的代码处理。

图 3. Bulk-out flow

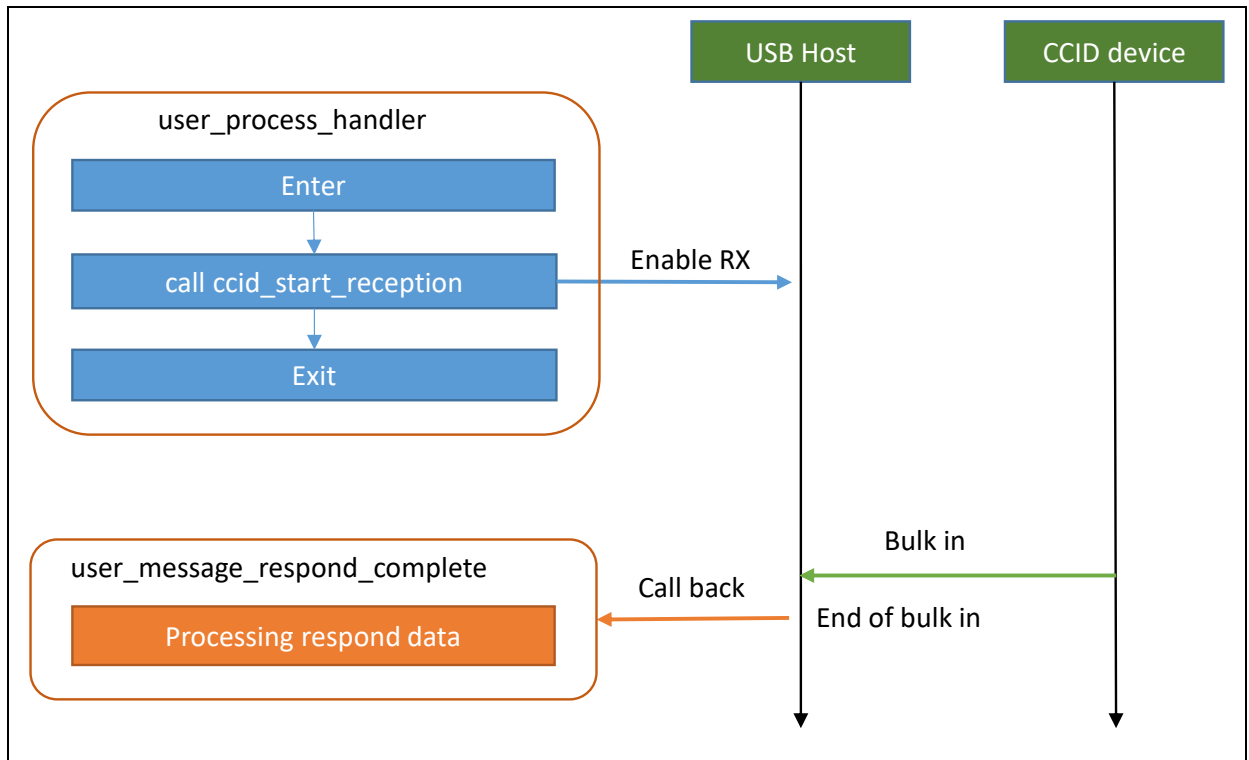


1.4.3 USB bulk-in flow

CCID装置会透过bulk端点回传数据给USB host，本章节将说明这部分的代码流程。

如下图，使用者预计要接收资料前，先呼叫ccid_start_reception。当host端收到装置端传送的数据后，示例中会呼叫user_message_respond_complete。此时使用者可以在此添加想要的代码处理。

图 4. Bulk-in flow



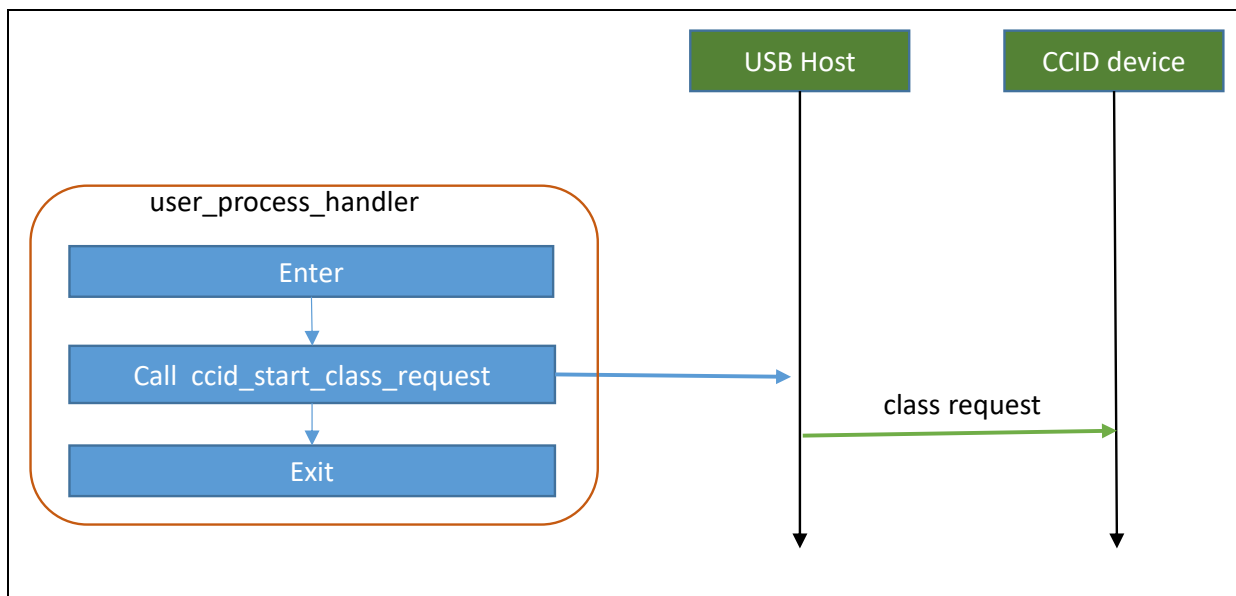
1.4.4 USB class request

本章节将说明如何修改代码，发送class request给装置端。

如下图，可以直接呼叫`ccid_start_class_request`，传入对应参数即可。

图 5. Class request flow

```
//class request(abort) testing
ccid_start_class_request(USB_CCID_STATE_ABORT);
```



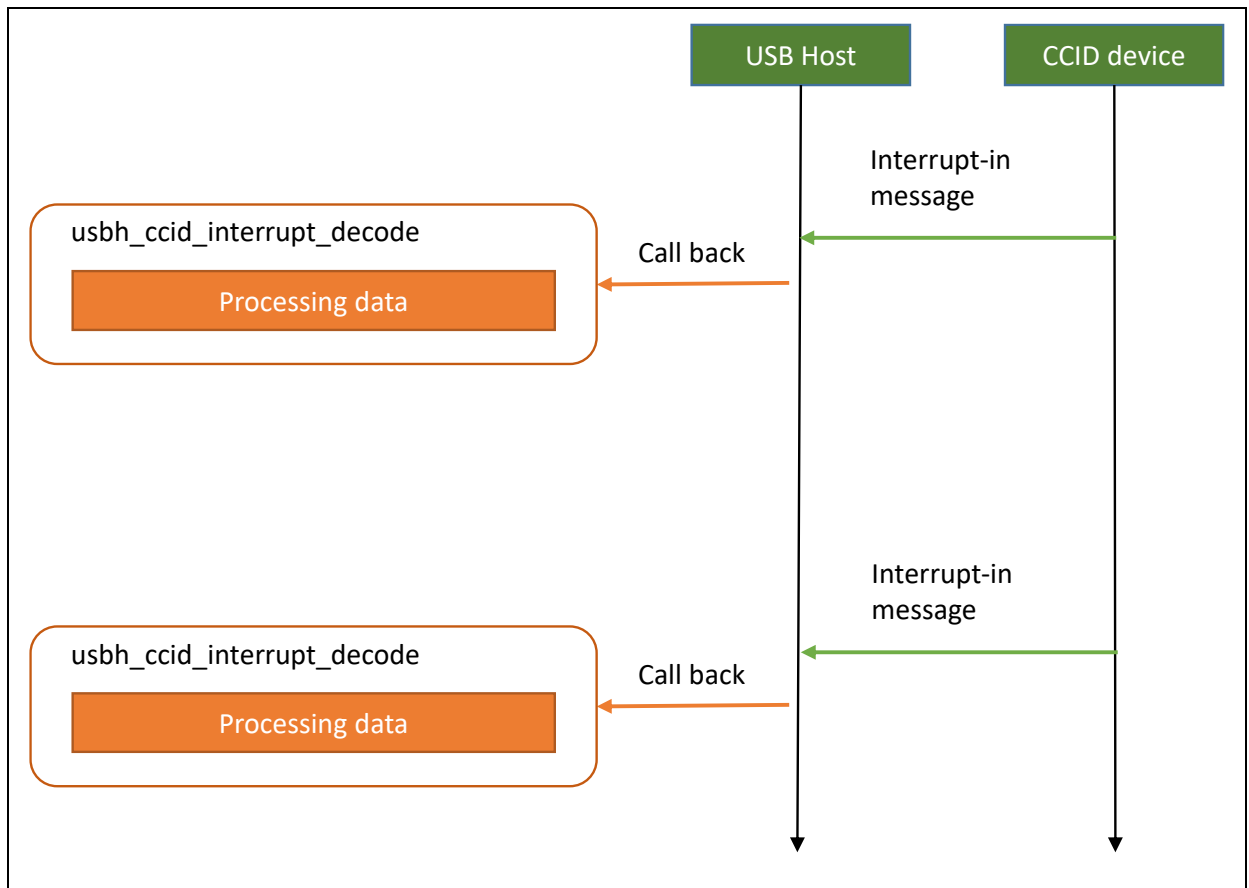
1.4.5 USB interrupt-IN flow

CCID装置会透过interrupt端点，发送数据给USB host端。本章节将说明这部分的代码流程。

如下图，当收到CCID装置端的数据后，系统会呼叫usbh_ccid_interrupt_decode，用户可以在此函数内处理收到的数据。

图 6. interrupt-IN flow

```
usbh_hid_keyboard.c  usbh_ccid_class.c  usbh_ccid_class.h  main.c
745  /**
746   * @brief user routine to decode ccid interrupt-in message
747   * @param data: interrupt-in message.
748   * @retval none.
749   */
750 void usbh_ccid_interrupt_decode(uint8_t *data)
751 {
752     //user code begin
753     //user code end
754 }
755
```



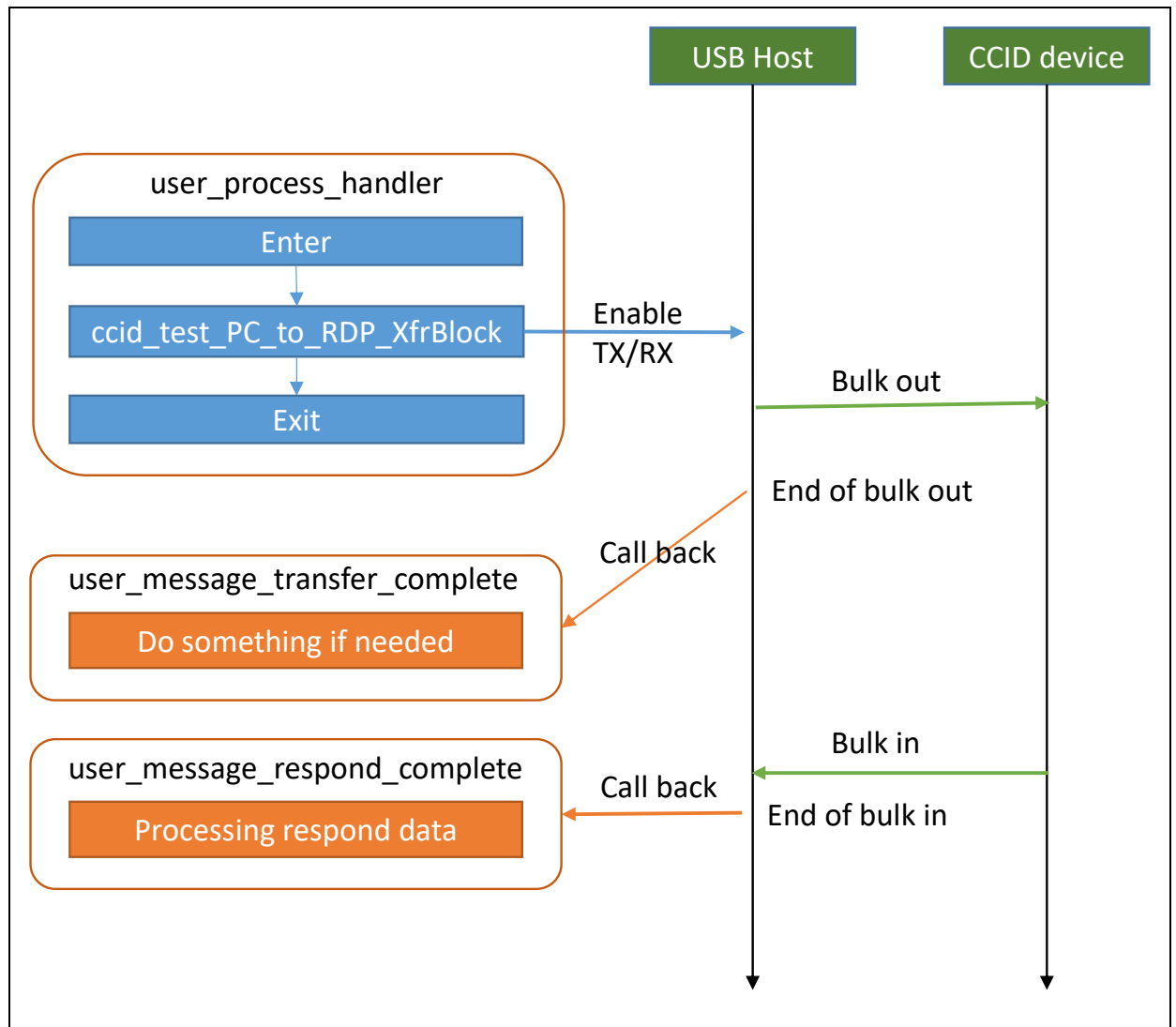
1.4.6 CCID bulk transfer example

本章节描述如何建构并发送CCID中的PC_to_RDP_XfrBlock，提供使用者参考。如下图。

图 7. PC_to_RDP_XfrBlock

```

void ccid_test_PC_to_RDP_XfrBlock(void)
{
    #define TEST_BLOCK_LEN (usbh_ccid.ccid.class_desc.dwMaxCCIDMessageLength-10)
    cmd_xfr_buf[0]=0x6F;
    cmd_xfr_buf[1]=(TEST_BLOCK_LEN >> 0)& 0xFF;
    cmd_xfr_buf[2]=(TEST_BLOCK_LEN >> 8)& 0xFF;
    cmd_xfr_buf[3]=(TEST_BLOCK_LEN >>16)& 0xFF;
    cmd_xfr_buf[4]=(TEST_BLOCK_LEN >>24)& 0xFF;
    cmd_xfr_buf[5]=0;
    cmd_xfr_buf[6]=1;
    cmd_xfr_buf[7]=0xFF;
    ccid_start_transmission(usbh_ccid.puhost, (uint8_t*)cmd_xfr_buf, TEST_BLOCK_LEN+10 );
    ccid_start_reception (usbh_ccid.puhost, (uint8_t*)reponse_buf, 10 );
}
  
```



1.4.7 USB trace log

示例除了有正常的枚举流程外，还展示对CCID装置发出PC_to_RDR_XfrBlock和class request的部分，提供使用者参考。如下图。

图 8.抓包截图

Trace View

Transfer 0	F	Control	ADDR	ENDP	bRequest	wValue	wIndex	Descriptors	Time	Time Stamp	
	S	GET	0	0	GET_DESCRIPTOR	DEVICE type	0x0000	DEVICE Descriptor	102.750 us	1 . 518 171 216	
Transfer 1	F	Control	ADDR	ENDP	bRequest	wValue	wIndex	Descriptors	Time	Time Stamp	
	S	GET	0	0	GET_DESCRIPTOR	DEVICE type	0x0000	DEVICE Descriptor	4.001 ms	1 . 518 273 966	
Transfer 2	F	Control	ADDR	ENDP	bRequest	wValue	wIndex	wLength	Time	Time Stamp	
	S	SET	0	0	SET_ADDRESS	New address 1	0x0000	0	100.750 us	1 . 522 275 350	
Transfer 3	F	Control	ADDR	ENDP	bRequest	wValue	wIndex	Descriptors	Time	Time Stamp	
	S	GET	1	0	GET_DESCRIPTOR	CONFIGURATION type, Index 0	0x0000	CONFIGURATION Descriptor	100.082 us	1 . 522 376 100	
Transfer 4	F	Control	ADDR	ENDP	bRequest	wValue	wIndex	Descriptors	Time	Time Stamp	
	S	GET	1	0	GET_DESCRIPTOR	CONFIGURATION type, Index 0	0x0000	9 Descriptors	225.984 us	1 . 522 476 182	
Transfer 5	F	Control	ADDR	ENDP	bRequest	wValue	wIndex	Descriptors	Time	Time Stamp	
	S	GET	1	0	GET_DESCRIPTOR	STRING type, Index 1	Language ID 0x0409	Circle	2.239 ms	1 . 522 702 166	
Transfer 6	F	Control	ADDR	ENDP	bRequest	wValue	wIndex	Descriptors	Time	Time Stamp	
	S	GET	1	0	GET_DESCRIPTOR	STRING type, Index 2	Language ID 0x0409	CIR615 CL & 1S	2.539 ms	1 . 524 941 450	
Transfer 7	F	Control	ADDR	ENDP	bRequest	wValue	wIndex	Descriptors	Time	Time Stamp	
	S	GET	1	0	GET_DESCRIPTOR	STRING type, Index 3	Language ID 0x0409	0056-000457	2.151 ms	1 . 527 480 550	
Transfer 8	F	Control	ADDR	ENDP	bRequest	wValue	wIndex	wLength	Time	Time Stamp	
	S	SET	1	0	SET_CONFIGURATION	New Configuration 1	0x0000	0	1.853 ms	1 . 529 531 316	
Transfer 9	F	Control	ADDR	ENDP	bRequest	wValue	wIndex	Descriptors	Time	Time Stamp	
	S	GET	1	0	GET_DESCRIPTOR	REPORT_DESCRIPTOR type	0x0001	REPORT Descriptor	188.416 us	1 . 531 484 666	
Transfer 10	F	Control	ADDR	ENDP	bRequest	wValue	wIndex	wLength	Time	Time Stamp	
	S	SET	1	0	SET_IDLE	0x0000	0x0001	0	67.000 us	1 . 531 673 082	
Transfer 11	F	Control	ADDR	ENDP	bRequest	wValue	wIndex	wLength	Time	Time Stamp	
	S	SET	1	0	SET_PROTOCOL	0x0000	0x0001	0x0000	8.003 sec	1 . 531 740 082	
Transfer 12	F	Bulk	ADDR	ENDP	SmartCard CCID	XfrBlock	Seq: 1	BW: 255 LevelParameter: 0x0000 Data: 512 bytes	Time	Time Stamp	
	S	OUT	1	1					51.500 us	9 . 534 712 966	
Transfer 13	F	Bulk	ADDR	ENDP	SmartCard CCID	DataBlock	Seq: 1	ICCStatus: No ICC CmdStatus: Error ErrCode: CMD_ABORTED ChainParam: 0x00	Time	Time Stamp	
	S	IN	1	1					5.250 us	9 . 534 764 466	
Transfer 14	F	Control	ADDR	ENDP	SmartCard CCID	Request	bSeq	bSlot	Interface	Time	Time Stamp
	S	SET	1	0		ABORT	0	0	0x0000	5.526 sec	9 . 534 769 716

2 版本历史

表 1. 文档版本历史

日期	版本	变更
2024.3.8	2.0.0	初始版本

重要通知 - 请仔细阅读

买方自行负责对本文所述雅特力产品和服务的选择和使用，雅特力概不承担与选择或使用本文所述雅特力产品和服务相关的任何责任。

无论之前是否有过任何形式的表示，本文档不以任何方式对任何知识产权进行任何明示或默示的授权或许可。如果本文档任何部分涉及任何第三方产品或服务，不应被视为雅特力授权使用此类第三方产品或服务，或许可其中的任何知识产权，或者被视为涉及以任何方式使用任何此类第三方产品或服务或其中任何知识产权的保证。

除非在雅特力的销售条款中另有说明，否则，雅特力对雅特力产品的使用和/或销售不做任何明示或默示的保证，包括但不限于有关适销性、适合特定用途（及其依据任何司法管辖区的法律的对应情况），或侵犯任何专利、版权或其他知识产权的默示保证。

雅特力产品并非设计或专门用于下列用途的产品：（A）对安全性有特别要求的应用，例如：生命支持、主动植入设备或对产品功能安全有要求的系统；（B）航空应用；（C）航天应用或航天环境；（D）武器，且/或（E）其他可能导致人身伤害、死亡及财产损害的应用。如果采购商擅自将其用于前述应用，即使采购商向雅特力发出了书面通知，风险及法律责任仍将由采购商单独承担，且采购商应独立负责在前述应用中满足所有法律和法规要求。

经销的雅特力产品如有不同于本文档中提出的声明和/或技术特点的规定，将立即导致雅特力针对本文所述雅特力产品或服务授予的任何保证失效，并且不应以任何形式造成或扩大雅特力的任何责任。

© 2024 雅特力科技 保留所有权利