



保密等级	A	TXW81x USB 方案开发指南	文件编号	TX-0000
发行日期	2023-12-12		文件版本	V1.1
<div>TXW81x USB 方案开发指南</div> <div></div>				
		珠海泰芯半导体有限公司 Zhuhai Taixin Semiconductor Co.,Ltd	珠海市高新区港湾一号科技园港 11 栋 3 楼	
版权所有 侵权必究 Copyright © 2023 by Tai Xin All rights reserved				

保密等级	A	TXW81x USB 方案开发指南	文件编号	TX-0000
发行日期	2023-12-12		文件版本	V1.1

# 责任与版权

## 责任限制

由于产品版本升级或者其他原因，本文档会不定期更新。除非另行约定，泰芯半导体有限公司对本文档所有内容不提供任何担保或授权。

客户应在遵守法律、法规和安全要求的前提下进行产品设计，并做充分验证。泰芯半导体有限公司对应用帮助或客户产品设计不承担任何义务。客户应对其使用泰芯半导体有限公司的产品和应用自行负责。

在适用法律允许的范围内，泰芯半导体有限公司在任何情况下，都不对因使用本文档相关内容及本文档描述的产品而产生的损失和损害进行超过购买支付价款的赔偿（除在涉及人身伤害的情况中根据适用的法律规定的损害赔偿外）。

## 版权申明

泰芯半导体有限公司保留随时修改本文档中任何信息的权利，无需提前通知且不承担任何责任。


未经泰芯半导体有限公司书面同意，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。除非获得相关权利人的许可，否则，任何人不能以任何形式对前述软件进行复制、分发、修改、摘录、反编译、反汇编、解密、反向工程、出租、转让、分许可等侵犯本文档描述的享有版权的软件版权的行为，但是适用法禁止此类限制的除外。




珠海泰芯半导体有限公司  
Zhuhai Taixin Semiconductor Co., Ltd

珠海市高新区港湾一号科技园港 11 栋 3 楼

版权所有 侵权必究  
Copyright © 2023 by Tai Xin All rights reserved

保密等级	A	TXW81x USB 方案开发指南	文件编号	TX-0000																								
发行日期	2023-12-12		文件版本	V1.1																								
<div>修订记录</div> <table><tr><th>日期</th><th>版本</th><th>描 述</th><th>修订人</th></tr><tr><td>2023-12-12</td><td>V1.1</td><td>添加 v2.5.1 USB 改动描述</td><td>TX</td></tr><tr><td>2023-11-25</td><td>V1.0</td><td>初始版本</td><td>TX</td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr></table>					日期	版本	描 述	修订人	2023-12-12	V1.1	添加 v2.5.1 USB 改动描述	TX	2023-11-25	V1.0	初始版本	TX												
日期	版本	描 述	修订人																									
2023-12-12	V1.1	添加 v2.5.1 USB 改动描述	TX																									
2023-11-25	V1.0	初始版本	TX																									
		珠海泰芯半导体有限公司 Zhuhai Taixin Semiconductor Co.,Ltd	珠海市高新区港湾一号科创园港 11 栋 3 楼																									
版权所有 侵权必究 Copyright © 2023 by Tai Xin All rights reserved																												

保密等级	A	TXW81x USB 方案开发指南	文件编号	TX-0000
发行日期	2023-11-25		文件版本	V1.0
<div>目录</div> <div>TXW81x USB 方案开发指南..... 1</div> <div>1. 概述..... 1</div> <div>2. USB Common..... 2</div> <div>3. USB HOST..... 5</div> <div>    3.1. USB Host 相关配置..... 5</div> <div>    3.2. USB Host 工作流程..... 5</div> <div>4. USB Device..... 6</div> <div>    4.1. USB Device 相关配置..... 6</div> <div>    4.2. USB Device 工作流程..... 7</div>				
		珠海泰芯半导体有限公司 Zhuhai Taixin Semiconductor Co.,Ltd	珠海市高新区港湾一号科技园港 11 栋 3 楼	
版权所有 侵权必究 Copyright © 2023 by Tai Xin All rights reserved				

## 1. 概述

本文为使用方案软件设计开发人员而写，目的帮助您快速入门 USB 方案开发。

本文档主要适用于以下工程师：

- 技术支持工程师
- 方案软件开发工程师

本文档适用的产品范围：

型号	封装	包装
TXW81x		

## 2. USB Common

TXW81x USB 支持 DEVICE 和 HOST 模式，支持 USB2.0 HighSpeed 传输，共有 EP0/1/2/3 四对收发端点，除去 EP0 枚举外还有 EP1/2/3 收发 6 个端点。

USB device 硬件抽象层请查看 usb\_device.h，此抽象包含了 USB HOST 和 DEVICE。

```
68:
69: struct usb_device {
70:     ... struct dev_obj dev;
71:     ... uint32 obj;
72:     ... int32 (*open)(struct usb_device *p_usb_d, struct usb_device_cfg *p_usbdev_cfg);
73:     ... int32 (*close)(struct usb_device *p_usb_d);
74:     ... int32 (*read)(struct usb_device *p_usb_d, int8 ep, int8 *buff, uint32 len, int8 sync);
75:     ... int32 (*write)(struct usb_device *p_usb_d, int8 ep, int8 *buff, uint32 len, int8 sync);
76:     ... int32 (*ioctl)(struct usb_device *p_usb_d, uint32 cmd, uint32 param1, uint32 param2);
77:     ... int32 (*request_irq)(struct usb_device *p_usb_d, usbdev_irq_hdl irqhdl, uint32 data);
78: };
79:
80: int32 usb_device_open(struct usb_device *p_usb_d, struct usb_device_cfg *p_usbdev_cfg);
81:
82: int32 usb_device_close(struct usb_device *p_usb_d);
83:
84: int32 usb_device_write(struct usb_device *p_usb_d, int8 ep, int8 *buff, uint32 len, int8 sync);
85:
86: int32 usb_device_read(struct usb_device *p_usb_d, int8 ep, int8 *buff, uint32 len, int8 sync);
87:
88: int32 usb_device_ioctl(struct usb_device *p_usb_d, uint32 cmd, uint32 param1, uint32 param2);
89:
90: int32 usb_device_request_irq(struct usb_device *p_usb_d, usbdev_irq_hdl handle, uint32 data);
91:
```

使用步骤如下：

- 1) 和其他外设一样，使用前在 device.c 中先将设备加入系统设备链表

```
351:
352: #if USB_EN
353: #if USB_HOST_EN
354:     ... hgsub20_host_attach(HG_USBDEV_DEVID, &usb20_dev);
355: #else
356:     ... hgsub20_dev_attach(HG_USBDEV_DEVID, &usb20_dev);
357: #endif
358: #endif
359:
```

- 2) 获取设备句柄(dev\_get)、打开设备(usb\_device\_open)、配置设备(usb\_device\_ioctl)、中断请求(usb\_device\_request\_irq)

```
196: ... usb_test.dev = (struct usb_device *) dev_get(HG_USBDEV_DEVID);
197: ... if (usb_test.dev) {
198:     ... usb_host_task_init(usb_test.dev);
199:     ... if (!usb_device_open(usb_test.dev, NULL)) {
200:         ... usb_host_uvc_ioctl(usb_test.dev, USB_HOST_IO_CMD_INSERT_HUB, &msgbuf, 0);
201:         ... printf("%s:%d\n", FUNCTION, LINE);
202:         ... uint32 usb_host_bus_irq(uint32 irq, uint32 param1, uint32 param2, uint32 param3);
203:         ... usb_device_request_irq(usb_test.dev, usb_host_bus_irq, (int32)usb_test.dev);
204:     }
205: }
```

详细说明：

- usb\_device\_ioctl 使用说明

```

24: enum usb_dev_io_cmd {
25:     ... USB_DEV_IO_CMD_AUTO_TX_NULL_PKT_ENABLE,
26:     ... USB_DEV_IO_CMD_AUTO_TX_NULL_PKT_DISABLE,
27:     ... USB_DEV_IO_CMD_REMOTE_WAKEUP,
28:
29:     ... /* this function need call after attatch & before open
30:         ... * msg[1:0]::vid
31:         ... * msg[3:2]::pid
32:         ... */
33:     ... USB_DEV_IO_CMD_SET_ID,
34: };
35:

```

- **【USB\_DEV\_IO\_CMD\_AUTO\_TX\_NULL\_PKT\_ENABLE】**：
- **【USB\_DEV\_IO\_CMD\_AUTO\_TX\_NULL\_PKT\_DISABLE】**：USB自动发送空包：当发送包长和MaxPacketSize对齐时，自动补发空包作为短包结束开关。
- **【USB\_DEV\_IO\_CMD\_REMOTE\_WAKEUP】**：USB Device远程唤醒Host
- **【USB\_DEV\_IO\_CMD\_SET\_ID】**：USB Device VID/PID设置，此动作需要在usb\_device\_open后马上执行。
- usb\_device\_request\_irq使用说明

目前支持的usb irq如下：

```

9:
10: enum usb_dev_irqs {
11:     ... USB_DEV_RESET_IRQ,
12:     ... USB_DEV_SUSPEND_IRQ,
13:     ... USB_DEV_RESUME_IRQ,
14:     ... USB_DEV_SOF_IRQ,
15:     ... USB_DEV_CTL_IRQ,
16:     ... USB_EP_RX_IRQ,
17:     ... USB_EP_TX_IRQ,
18:     ... USB_CONNECT,
19:     ... USB_DISCONNECT,
20:     ... USB_BABBLE,
21:     ... USB_XACT_ERR,
22: };
23:

```

- **【USB\_DEV\_RESET\_IRQ】**：usb device收到总线reset信号
- **【USB\_DEV\_SUSPEND\_IRQ】**：usb device收到总线suspend信号
- **【USB\_DEV\_RESUME\_IRQ】**：usb device收到总线resume信号
- **【USB\_DEV\_SOF\_IRQ】**：usb device收到总线sof信号
- **【USB\_DEV\_CTL\_IRQ】**：usb 端点 0 收发中断

- **【USB\_EP\_RX\_IRQ】**：usb 端点收到数据中断
- **【USB\_EP\_TX\_IRQ】**：usb 端点发送数据完成中断
- **【USB\_CONNECT】**：usb host发现设备连接上来中断
- **【USB\_DISCONNECT】**：usb host发现设备连接断开中断
- **【USB\_BABBLE】**：usb host遇到babble
- **【USB\_XACT\_ERR】**：usb host遇到xact error



### 3. USB HOST

#### 3.1. USB Host 相关配置

方案配置看 project\_config.h 配置：USB\_HOST\_EN 和 USB\_EN 均使能。

#### 3.2. USB Host 工作流程

USB Host 软件配置和工作流程如下。

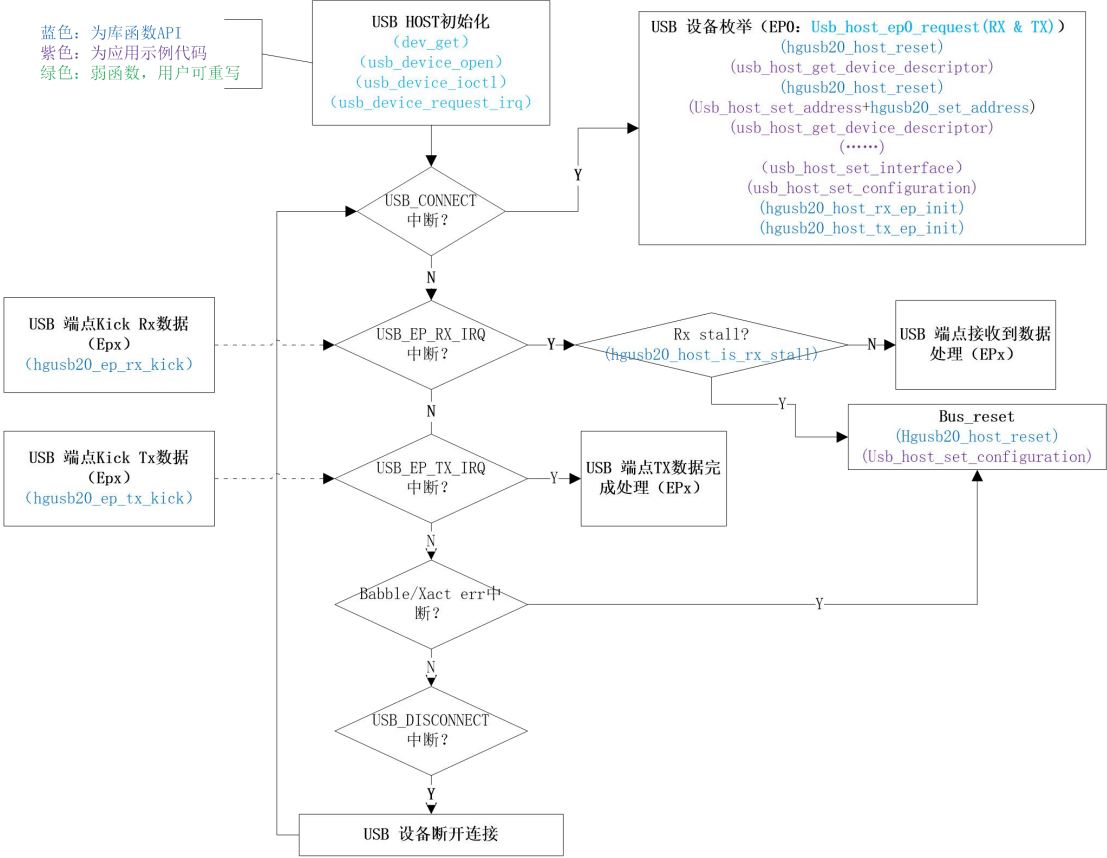


图3. 2. 1 USB Host工作流程图

## 4. USB Device

### 4.1. USB Device 相关配置

方案配置看 project\_config.h 配置：USB\_EN 使能。

Device 配置已经抽象到了一个结构体：struct usb\_device\_cfg，此结构体在 usb\_device\_open（）的时候传入，USB 后台枚举时会提取相应信息做枚举，结构体定义如下：

```
4:
5: struct usb_device_cfg {
6:     ...uint16_t vid; /*usb.device.VID*/
7:     ...uint16_t pid; /*usb.device.PID*/
8:     ...uint8_t speed; /*not used*/
9:     ...uint8_t *p_device_descriptor; /*usb.device.descriptor*/
10:    ...uint8_t *p_config_desc_hs; /*usb.config.descriptor*/
11:    ...uint8_t *p_config_desc_fs; /*usb.config.descriptor*/
12:    ...uint16_t config_desc_len; /*usb.config.descriptor.total.len*/
13:    ...uint8_t interface_num;
14:
15:    ...uint8_t ep_nums;
16:    ...struct usb_device_ep_cfg *ep_cfg[8]; /*must same with config_desc*/
17: };
18:
19: typedef uint32_t (*usbdev_irq_hdl)(uint32_t irq, uint32_t param1, uint32_t param2, uint32_t param3);
20:
```

【vid】：厂商 ID

【pid】：产品 ID

【speed】：暂未使用

【p\_device\_descriptor】：设备描述符

【p\_config\_desc】：配置描述符

【config\_desc\_len】：配置描述符长度

【interface\_num】：接口数量

【ep\_nums】：端点数量（EP0 不计入）

【ep\_cfg】：端点配置：端点号、传输方向、传输类型、最大包大小

下面是自定义 USB 设备类配置示例：

```

11:
12: static const struct usb_device_cfg usb_dev_wifi_cfg = {
13:     ....vid.....=USB_VID,
14:     ....pid.....=USB_PID,
15:     ....speed.....=USB_SPEED_HIGH,
16:     ....p_device_descriptor.=(uint8_t*)tbl_usb_wifi_device_descriptor,
17:
18:     ....p_config_desc_hs.=(uint8_t*)tbl_usb_wifi_config_all_descriptor_wifi_hs,
19:     ....p_config_desc_fs.=(uint8_t*)tbl_usb_wifi_config_all_descriptor_wifi_fs,
20:     ....config_desc_len.=sizeof(tbl_usb_wifi_config_all_descriptor_wifi_hs),
21:     ....interface_num.....=1,
22:
23:     ....ep_nums.....=2,
24:     ....ep_cfg[0].ep_id.....=USB_WIFI_RX_EP,
25:     ....ep_cfg[0].ep_type.....=USB_ENDPOINT_XFER_BULK,
26:     ....ep_cfg[0].ep_dir_tx.....=0,
27:     ....ep_cfg[0].max_packet_size_hs.....=USB_WIFI_EP_MAX_PKT_SIZE_HS,
28:     ....ep_cfg[0].max_packet_size_fs.....=USB_WIFI_EP_MAX_PKT_SIZE_FS,
29:     ....ep_cfg[1].ep_id.....=USB_WIFI_TX_EP,
30:     ....ep_cfg[1].ep_type.....=USB_ENDPOINT_XFER_BULK,|
31:     ....ep_cfg[1].ep_dir_tx.....=1,
32:     ....ep_cfg[1].max_packet_size_hs.....=USB_WIFI_EP_MAX_PKT_SIZE_HS,
33:     ....ep_cfg[1].max_packet_size_fs.....=USB_WIFI_EP_MAX_PKT_SIZE_FS,
34: };
35:

```

图4.1.1 自定义USB设备类配置示例

## 4.2. USB Device 工作流程

- USB Device配置和工作流程如：

➤ 图 4.2.1 USB Device软件配置和工作主流程

➤ 图 4.2.2 USB Device EP0 枚举流程

- i. v2.5.0 SDK 枚举采用“后台”枚举流程
- ii. v2.5.1 以后SDK 开放了设备枚举流程， 用户需实现以下函数：
  - i) usb\_device\_ep0\_setup\_irq: 处理获取各种描述符请求、类请求、厂商请求等信息
  - ii) usb\_device\_ep0\_rx\_irq: 数据阶段已接收完成的数据处理
  - iii) usb\_device\_ep0\_tx\_irq: 数据阶段已发送完成处理

常规情况下，简单的 USB Device 方案，用户只需要配置好 struct usb\_device\_cfg 即可实现；

如果配置 usb\_device\_cfg 无法满足用户需求，则可以重写绿色部分的函数实现用户想要实现的功能，此动作需要要求熟悉 USB 协议，非必要不建议使用。

