



TXW81x 数据流方案开发指南



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

保密等级	A	TXW81x 数据流方案开发指南	文件编号	TX-0000
发行日期	2023-11-10		文件版本	V1.0

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

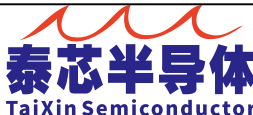
保密等级	A	TXW81x 数据流方案开发指南	文件编号	TX-0000
发行日期	2023-11-10		文件版本	V1.0

修订记录

日期	版本	描 述	修订人
2023-11-10	V1.0	初始版本	TX

 泰芯半导体 TaiXin Semiconductor	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 数据流方案开发指南	文件编号	TX-0000
发行日期	2023-11-10		文件版本	V1.0
<div>目录</div> <div>TXW81x 数据流方案开发指南..... 1</div> <div>1. 概述..... 1</div> <div>2. SDK 视频音频数据流开发指南以及 API 说明.....2</div> <div>2.1. 接收与发送通用接口.....2</div> <div>2.1.1. open_stream_available.....2</div> <div>2.1.2. close_tream.....3</div> <div>2.1.3. enable_stream.....3</div> <div>2.1.4. streamSrc_bind_streamDest 与 streamSrc_unbind_streamDeset..... 4</div> <div>2.2. 接收方常用的 api 接口以及说明.....4</div> <div>2.2.1. recv_real_data.....4</div> <div>2.2.2. get_stream_real_data.....4</div> <div>2.2.3. get_stream_real_data_len.....4</div> <div>2.3. 发送方常用的 api 接口以及说明.....5</div> <div>2.3.1. send_data_to_stream.....5</div> <div>2.3.2. set_stream_real_data_len.....5</div> <div>2.3.3. set_stream_data_time.....5</div> <div>2.3.4. broadcast_cmd_to_destStream.....5</div> <div>2.3.5. get_src_data_f.....5</div> <div>2.3.6. force_del_data.....6</div> <div>3. stream_frame 的接收以及发送简单流程图.....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创园港 11 栋 3 楼	
版权所有 侵权必究 Copyright © 2023 by Tai Xin All rights reserved				

1. 概述

本文主要描述视频开发流程。

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. SDK 视频音频数据流开发指南以及 API 说明

SDK 是基于 stream_frame 的框架去实现数据流管理，所有任务都可以通过框架获取所需要的数据(音频、视频)，这里主要涉及接收方以及发送方概念。

2.1. 接收与发送通用接口

2.1.1. open_stream_available

- `void *open_stream_available(const char *name, int data_count, int recv_count, stream_priv_func func, void *priv)`: 无论发送还是接收，都需要用到这个接口。
- **Name**: 流的名称，这个要是唯一。
- **data_count**: 生成数据源头节点数量(这个是发送方需要考虑的，节点数量影响可以缓冲多少个节点，比如是图片，在内存足够的情况下，最多可以发送data_count张图片)。
- **recv_count**: 接收的数据的节点数量(这个是接收方考虑的，这个节点数量是代表接收缓冲区接收的节点数，如果接收满了，下一次发送的数据就不会被接收，所以如果需要接收所有的数据：(1)足够内存以及节点，(2)处理足够快，释放足够快)。
- **func**: 注册的一个回调函数，在流运行过程中，会有各种各样事件可能需要处理，这里结合代码宏举个例子：

以图片的流(video_app.c)说明：

- **STREAM_OPEN_EXIT**: 流第一次open_stream_available成功后最后调用，再这里是bind流以及将硬件jpeg模块打开，创建接收任务。
- **STREAM_OPEN_FAIL**: 是创建失败才会调用。
- **STREAM_DATA_DIS**: 是结合stream_data_dis_mem使用，一般是为每一个节点分配内存空间。
- **STREAM_DATA_DESTORY**: 是在close_stream后，并且等待回收内存的时候调用，主

要为了回收节点分配的内存，一般与STREAM_DATA_DIS分配内存对应。

- **STREAM_DATA_FREE:** 节点释放的时候使用，一般就是bind的流接收到数据后，使用完就会调用free_data，所有流用完该节点，代表节点数据已经可以被释放的时候调用，实现内容主要看jpeg流有哪些空间需要释放(如果是接收流，不会有这个事件)。
- **STREAM_DATA_FREE_END:** 与STREAM_DATA_FREE相似，只是执行时间更加推迟一点，并且节点是归还到可用节点链表中。
- **STREAM_SEND_DATA_FINISH:** 是send_data_to_stream调用发送完成后的事件，代表data已经发送完毕，如果发送完成后需要执行通知或者其他行为可以在这里实现。
- **STREAM_CLOSE_EXIT:** 这个是close_stream结束的事件，注意，这里代表close结束，但是流不一定被回收释放，可能要经过 5-10ms才实际被回收，但这个事件来后，这个流就不要再用了，至于申请的节点空间等，都会在后台释放，节点释放事件是(STREAM_DATA_DESTROY)。

还有更多的事件的宏定义可以看 stream_frame.h，不是所有事件都要响应，根据流的特性去实现，一般产生 data 数据的流需要响应事件比较多，接收流响应事件会比较少。

2.1.2. close_tream

- **int close_stream(struct stream_s *s);**

关闭流的时候使用，与open_stream_available成对使用，这个接口实际不会关闭，需要等到所有open_stream_available对应都关闭后，才会真正关闭，所以建议是同一个任务或者同一个事件调用open_stream_available与close_stream成对使用。

2.1.3. enable_stream

- **int enable_stream(struct stream_s *s,int enable);**

是否使能流，这个主要是影响接收，发送暂时没有影响。

2.1.4. streamSrc_bind_streamDest 与

streamSrc_unbind_streamDest

- `int streamSrc_bind_streamDest(struct stream_s *s, const char *name);`
`int streamSrc_unbind_streamDest(struct stream_s *s, const char *name);`

分别是绑定以及解绑流，现在阶段只有open_stream和close_stream调用了。

2.2. 接收方常用的 api 接口以及说明

2.2.1. recv_real_data

- `struct data_structure *recv_real_data(struct stream_s *s);`

该api是接收数据，只要源头的流有发送数据过来，就可以从struct stream_s *s接收到数据，然后通过struct data_structure的类型去解析对应的数据，解析这个数据会分别用到：`void *get_stream_real_data(struct data_structure* data);`和
`uint32_t get_stream_real_data_len(struct data_structure* data);`

2.2.2. get_stream_real_data

- `void *get_stream_real_data(struct data_structure* data);`

获取实际数据的内容(这个也要根据类型来判断，大部分就是实际的数据buf，但jpeg是有区别)，比如音频，get_stream_real_data获取到就是一个音频的pcm数据，如果是图片，则由于内存存在，图片是以节点形式存在，需要特定解析。

2.2.3. get_stream_real_data_len

- `uint32_t get_stream_real_data_len(struct data_structure* data);`

获取data的长度，如果是音频，就是返回音频的长度，如果是jpeg图片，则返回图片的大小

2.3. 发送方常用的 api 接口以及说明

2.3.1. send_data_to_stream

- `int send_data_to_stream(struct data_structure *data);`

发送data数据到每一个bind的流(已经创建并且使能)，只要接收方有空间，并且接收，就可以在接收流中通过recv_real_data接收到对应的data数据。

2.3.2. set_stream_real_data_len

- `uint32_t set_stream_real_data_len(struct data_structure* data,uint32_t len);`

是设置data的数据长度，接收方通过get_stream_real_data_len获取，一般每次新数据都要设置一次，比如图片，每次生成都不一样，所以发送之前要先设置好。

2.3.3. set_stream_data_time

- `uint32_t set_stream_data_time(struct data_structure* data,uint32_t timestamp);`

与 2.3.2 同样，这个设置时间戳。

2.3.4. broadcast_cmd_to_destStream

- `void broadcast_cmd_to_destStream(stream *s,uint32_t cmd)`

这个是广播命令，会向所有bind的流发送cmd的命令，暂时少用，作为拓展，比如播放视频可以暂停，等操作，需要对应流实现响应对应事件STREAM_SEND_CMD。

2.3.5. get_src_data_f

- `struct data_structure *get_src_data_f(struct stream_s *s);`

这个是从stream_s *s中获取一个可用的节点，等待填充以及配置完毕后，再调用

send_data_to_stream发送到bind的各个流。

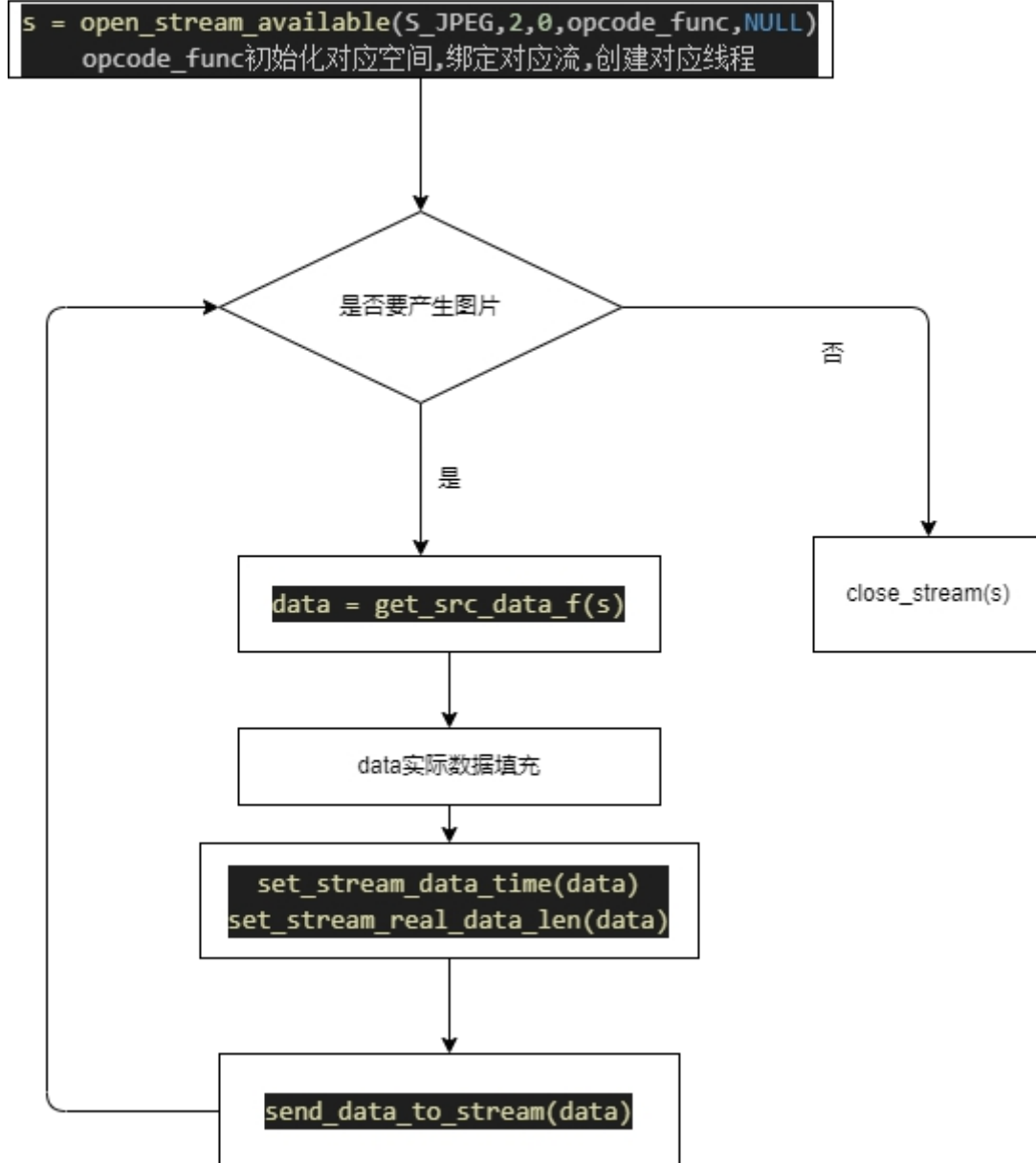
2.3.6. force_del_data

- `int force_del_data(struct data_structure *data);`

这个是一般发送方使用，因为调用get_src_data_f后，如果数据出现异常不需要了，则可以通过这个接口force_del_data删除，不会出现节点丢失，因为get_src_data_f获取的data不释放，后续就会获取不到。

3. stream_frame 的接收以及发送简单流程图

发送方流程(可以参考video_app.c)



接收方流程

