



CV180X & CV181X 软件 CviSysLink 使用手册

Version: 1.2.0

Release date: 2023/09

©2022 北京晶视智能科技有限公司
本文件所含信息归北京晶视智能科技有限公司所有。
未经授权，严禁全部或部分复制或披露该等信息。

目录

1	声明	1
2	文档概述	2
2.1	目的	2
2.2	适用范围	2
2.3	术语和定义	2
3	模块概述	3
3.1	模块架构	3
3.1.1	IPCMSG	4
3.1.2	DATAFIFO	4
4	功能概述	6
4.1	功能概述	6
4.1.1	IPCMSG	6
4.1.2	DATAFIFO	6
5	IPCMSG	7
5.1	API 参考	7
5.1.1	CVI_IPCMSG_CreateMessage	7
5.1.2	CVI_IPCMSG_CreateRespMessage	8
5.1.3	CVI_IPCMSG_DestroyMessage	9
5.1.4	CVI_IPCMSG_AddService	10
5.1.5	CVI_IPCMSG_DelService	11
5.1.6	CVI_IPCMSG_TryConnect	12
5.1.7	CVI_IPCMSG_Connect	12
5.1.8	CVI_IPCMSG_Disconnect	13
5.1.9	CVI_IPCMSG_IsConnected	14
5.1.10	CVI_IPCMSG_SendAsync	15
5.1.11	CVI_IPCMSG_SendSync	16
5.1.12	CVI_IPCMSG_Run	17
5.1.13	CVI_IPCMSG_SendOnly	17
5.2	数据类型	18
5.2.1	CVI_IPCMSG_MAX_CONTENT_LEN	18
5.2.2	CVI_IPCMSG_PRIVDATA_NUM	19
5.2.3	CVI_IPCMSG_INVALID_MSGID	19
5.2.4	CVI_IPCMSG_MAX_SERVICENAME_LEN	19
5.2.5	CVI_IPCMSG_CONNECT_S	20
5.2.6	CVI_IPCMSG_MESSAGE_S	21
5.2.7	CVI_IPCMSG_HANDLE_FN_PTR	22
5.2.8	CVI_IPCMSG_RESPHANDLE_FN_PTR	22

5.3	错误码	23
6	DATAFIFO	24
6.1	API 参考	24
6.1.1	CVI_DATAFIFO_Open	24
6.1.2	CVI_DATAFIFO_OpenByAddr	25
6.1.3	CVI_DATAFIFO_Close	26
6.1.4	CVI_DATAFIFO_Read	27
6.1.5	CVI_DATAFIFO_Write	27
6.1.6	CVI_DATAFIFO_CMD	28
6.2	数据类型	29
6.2.1	CVI_DATAFIFO_HANDLE	29
6.2.2	CVI_DATAFIFO_INVALID_HANDLE	30
6.2.3	CVI_DATAFIFO_RELEASESTREAM_FN_PTR	30
6.2.4	CVI_DATAFIFO_OPEN_MODE_E	30
6.2.5	CVI_DATAFIFO_PARAMS_S	31
6.2.6	CVI_DATAFIFO_CMD_E	32
6.3	错误码	32

1 声明



法律声明

本数据手册包含北京晶视智能科技有限公司（下称“晶视智能”）的保密信息。未经授权，禁止使用或披露本数据手册中包含的信息。如您未经授权披露全部或部分保密信息，导致晶视智能遭受任何损失或损害，您应对因之产生的损失/损害承担责任。

本文件内信息如有更改，恕不另行通知。晶视智能不对使用或依赖本文件所含信息承担任何责任。本数据手册和本文件所含的所有信息均按“原样”提供，无任何明示、暗示、法定或其他形式的保证。晶视智能特别声明未做任何适销性、非侵权性和特定用途适用性的默示保证，亦对本数据手册所使用、包含或提供的任何第三方的软件不提供任何保证；用户同意仅向该第三方寻求与此相关的任何保证索赔。此外，晶视智能亦不对任何其根据用户规格或符合特定标准或公开讨论而制作的可交付成果承担责任。

联系我们

地址 北京市海淀区丰豪东路 9 号院中关村集成电路设计园（ICPARK）1 号楼

深圳市宝安区福海街道展城社区会展湾云岸广场 T10 栋

电话 +86-10-57590723 +86-10-57590724

邮编 100094（北京）518100（深圳）

官方网站 <https://www.sophgo.com/>

技术论坛 <https://developer.sophgo.com/forum/index.html>

2 文档概述

本文档主要用于指导用户使用多媒体双系统核间通信模块进行开发以提高开发效率，文档描述了 IPCMSG 和 DATAFIFO 的主要功能和开发参考。用户可使用 IPCMSG 和 DATAFIFO 这两个模块解决双核消息通信和数据传输问题。我们不保证此文档更改后能及时告知用户，因此请使用最新版 released SDK 中的文档。

2.1 目的

指导用户使用多媒体双系统核间通信模块 API 进行开发。

2.2 适用范围

此文件涵盖了算能多媒体软件开发 SDK 中的双系统核间通信模块。适用人群：技术支持工程师、软件开发工程师。

2.3 术语和定义

表 2.1: 术语和定义列表

序号	术语	定义说明
1.	IPCM	跨核消息通信的驱动层，通过中断和共享内存来实现消息的发送接收，向上层提供标准的设备读写接口。
2.	IPCMSG	对 IPCM 的封装，给用户提供了 Message 接口，通过 Message 的收发实现双核间的信息传递。
3.	物理地址	DDR 上的绝对地址，在双核双系统两端都可见。
4.	虚拟地址	每个系统处理的方式不一样，在 Linux 上，虚拟地址只有在同一进程内可见。
5.	Ring Buffer	为了方便地址的偏移，规定存储在 Ring Buffer 里的数据要求定长，所以一般情况下，Ring Buffer 里只存储指向码流数据的指针，不存储码流数据。

3 模块概述

CviSysLink 包含两个模块：IPCMSG 和 DATAFIFO。前者用于跨核通讯，后者用于跨核数据传输。

3.1 模块架构

图 3.1 为 CviSysLink 的整体架构。

主要有两个模块构成：

- IPCMSG
- DATAFIFO

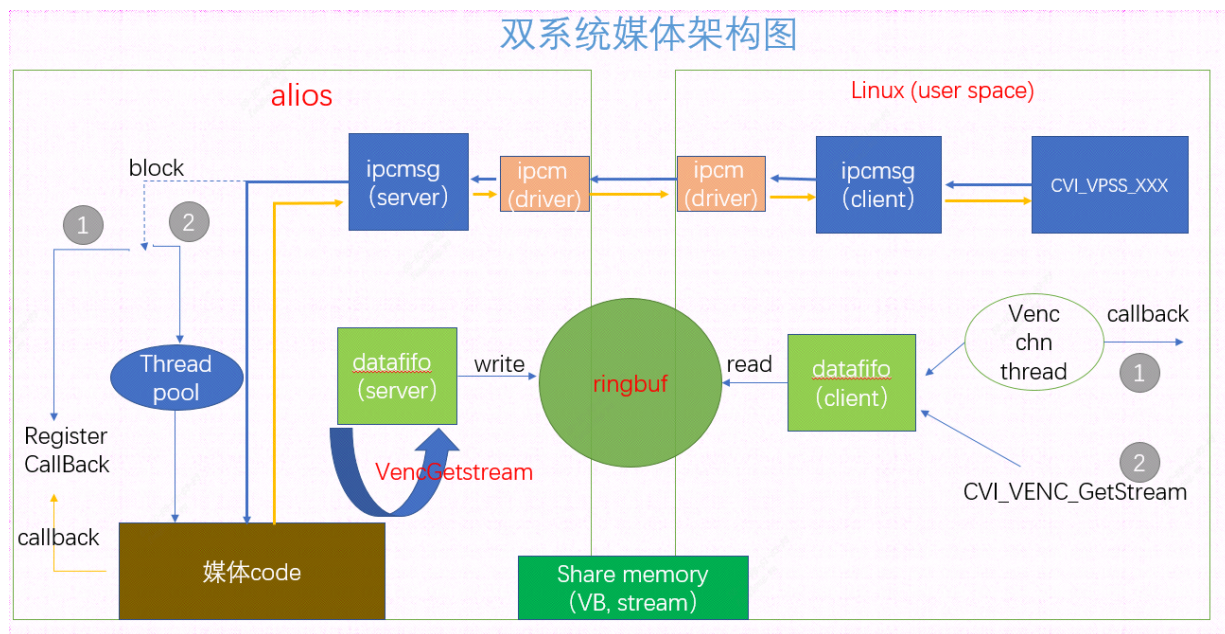


图 3.1: CviSysLink 系统架构

3.1.1 IPCMSG

此模块旨在解决在双核双系统环境下，在两个系统部署的模块间通信的问题，且通信的数据量不能太大（一次发送不能超过 1024 字节）。

图 3.2 为 IPCMSG 流程图。

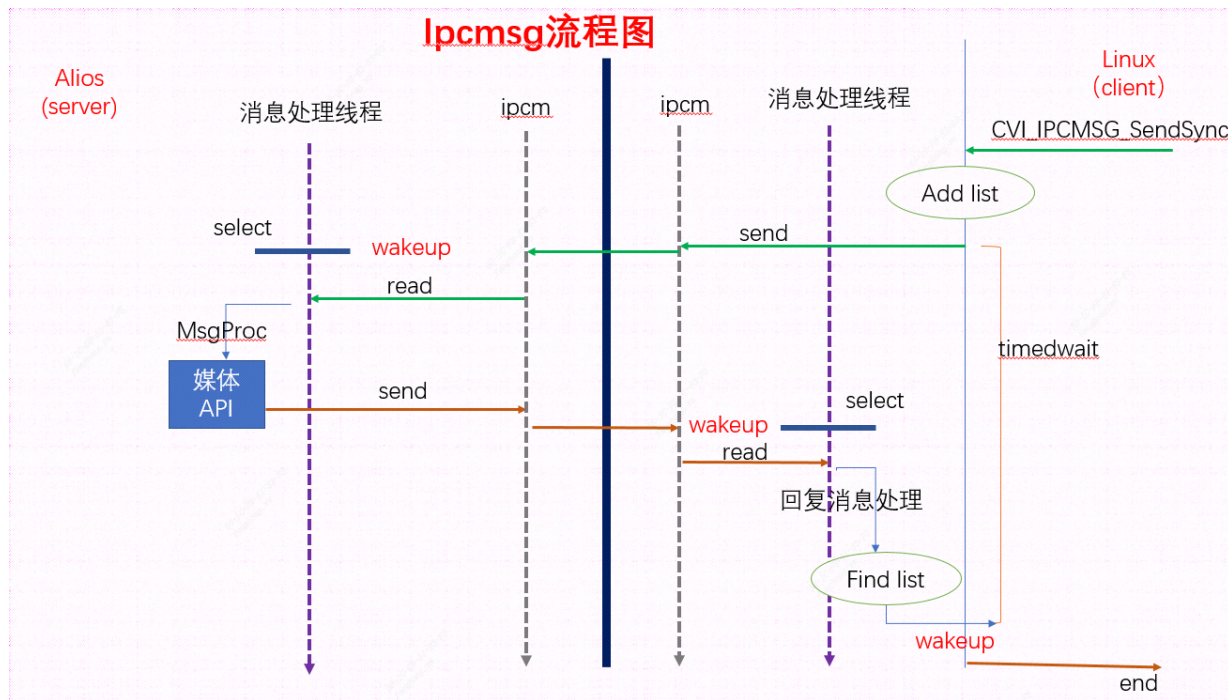


图 3.2: IPCMSG 流程图

3.1.2 DATAFIFO

在频繁传递大数据量时（如编解码），用 IPMSG 无论在效率还是可行性上都做不到，DATAFIFO 提供了处理这种问题的机制。大致流程如图 3.3 所示。在 DATAFIFO 里维护了码流数据的指针，写端申请到码流数据后，将指针送到到 DATAFIFO 里的 RingBuf 中；当读端要读数据时，DATAFIFO 会将存储的数据指针传给读端。在双核数据传输时，一端只能写入，另一端只能读取。

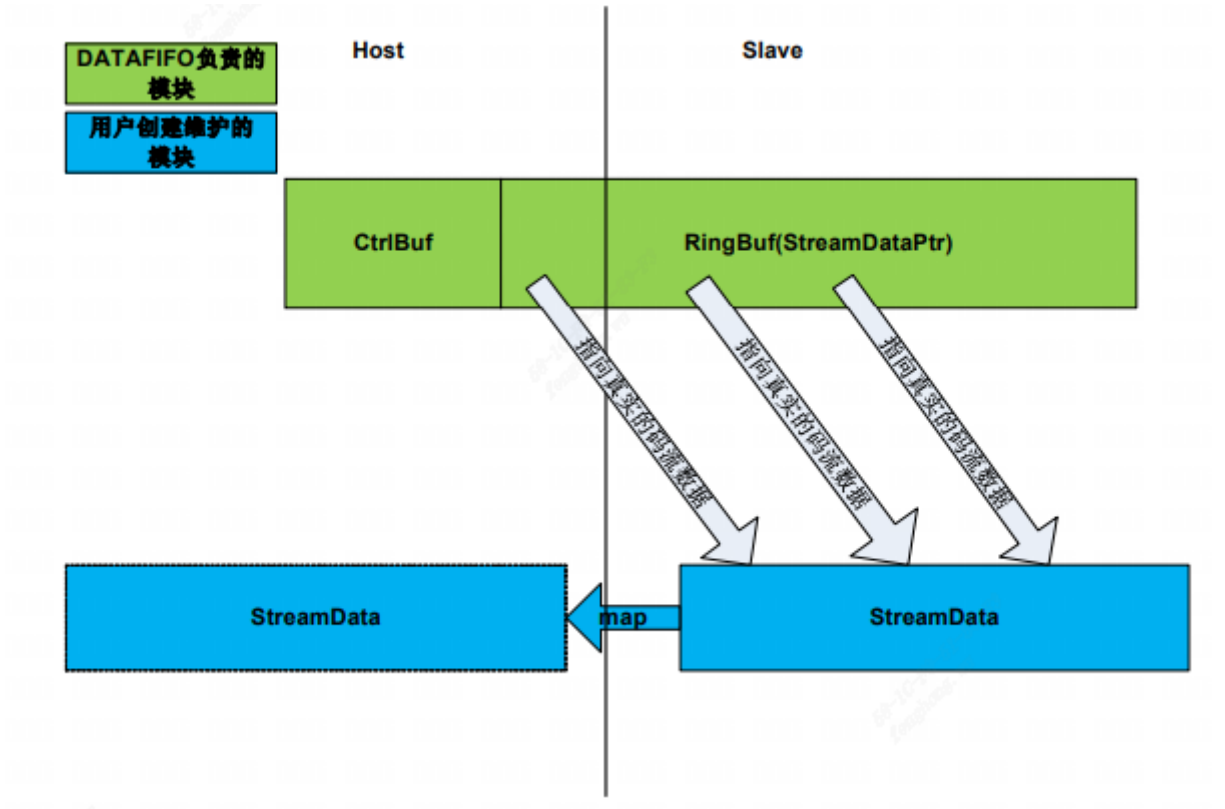


图 3.3: 双核数据传输示意图

4 功能概述

4.1 功能概述

4.1.1 IPCMSG

IPCMSG 模块包含消息的创建和销毁，服务的添加和删除，建立连接，断开连接，发送消息等功能。建立连接支持阻塞和非阻塞建立连接；消息发送支持同步消息和异步消息发送两种方式。同步消息支持超时机制。无论发送同步消息还是异步消息，如果回复消息大于 60 秒，则回复消息会被丢弃。

4.1.2 DATAFIFO

用于跨核数据传输的模块，数据先进入的先取出。由于两个系统上通过内存共享方式进行数据的传输，一端负责写入数据，另一端负责读取数据，DataFifo 内部维护了读写两端的写头、写尾、读头、读尾四个指针。在一端循环 Buffer 中进行操作来完成数据的传输。DATAFIFO 主要包含通路的打开、关闭、数据的写入和读出，以及其他控制命令。

5 IPCMSG

5.1 API 参考

该功能模块提供以下 API:

- CVI_IPCMSG_CreateMessage : 创建消息。
 - CVI_IPCMSG_CreateRespMessage : 创建回复消息。
 - CVI_IPCMSG_DestroyMessage : 销毁消息。
 - CVI_IPCMSG_AddService : 添加服务。
 - CVI_IPCMSG_DelService : 删除服务。
 - CVI_IPCMSG_TryConnect : 非阻塞连接。
 - CVI_IPCMSG_Connect : 阻塞方式建立连接。
 - CVI_IPCMSG_Disconnect : 断开连接。
 - CVI_IPCMSG_IsConnected : 获取是否连接状态。
 - CVI_IPCMSG_SendAsync : 发送异步消息。
 - CVI_IPCMSG_SendSync : 发送同步消息。
 - CVI_IPCMSG_Run : 消息处理函数。
 - CVI_IPCMSG_SendOnly : 仅发送消息的函数
-

5.1.1 CVI_IPCMSG_CreateMessage

【描述】

创建消息。

【语法】

```
CVI_IPCMSG_MESSAGE_S *CVI_IPCMSG_CreateMessage(CVI_U32 u32Module, CVI_U32  
→u32CMD,  
CVI_VOID *pBody, CVI_U32 u32BodyLen)
```

【参数】

参数名称	描述	输入/输出
u32Module	模块 ID。由用户创建，用于区分不同模块的不同消息。	输入
u32CMD	u32CMD 命令 ID。由用户创建，用于区分同一模块下的不同命令。	输入
pBody	消息体指针。	输入
u32BodyLen	消息体大小。	输入

【返回值】

返回值	描述
CVI_IPCMSG_MESSAGE_*	消息结构体指针。
NULL	消息创建失败。

【需求】

- 头文件：cvi_comm_ipcmsg.h、cvi_ipcmsg.h
- 库文件：cvilink.a

【注意】

无

【举例】

无

【相关主题】

[CVI_IPCMSG_DestroyMessage](#)

5.1.2 CVI_IPCMSG_CreateRespMessage

【描述】

创建回复消息。

【语法】

```
CVI_IPCMSG_MESSAGE_S *CVI_IPCMSG_CreateRespMessage(CVI_IPCMSG_MESSAGE_S_
↪ *pstRequest,
CVI_S32 s32RetVal, CVI_VOID *pBody, CVI_U32 u32BodyLen);
```

【参数】

参数名称	描述	输入/输出
pstRequest	请求消息的指针。	输入
s32RetVal	回复返回值。	输入
pBody	回复消息的消息体指针。	输入
u32BodyLen	回复消息的消息体大小。	输入

【返回值】

返回值	描述
CVI_IPCMSG_MESSAGE_S *	消息结构体指针。
NULL	消息创建失败。

【需求】

- 头文件: cvi_comm_ipcmsg.h、cvi_ipcmsg.h
- 库文件: cvilink.a

【注意】

无

【举例】

无

【相关主题】

[CVI_IPCMSG_DestroyMessage](#)

5.1.3 CVI_IPCMSG_DestroyMessage

【描述】

销毁消息。

【语法】

CVI_VOID CVI_IPCMSG_DestroyMessage(CVI_IPCMSG_MESSAGE_S *pstMsg);

【参数】

参数名称	描述	输入/输出
pstMsg	消息指针。	输入

【返回值】

返回值	描述
CVI_VOID	无

【需求】

- 头文件: `cvi_comm_ipcmsg.h`、`cvi_ipcmsg.h`
- 库文件: `cvilink.a`

【注意】

无

【举例】

无

【相关主题】

`CVI_IPCMSG_CreateMessage` `CVI_IPCMSG_CreateRespMessage`

5.1.4 CVI_IPCMSG_AddService

【描述】

添加服务。

【语法】

```
CVI_S32 CVI_IPCMSG_AddService(const CVI_CHAR *pszServiceName, const CVI_IPCMSG_
→CONNECT_S *pstConnectAttr);
```

【参数】

参数名称	描述	输入/输出
<code>pszServiceName</code>	服务的名称指针。	输入
<code>pstConnectAttr</code>	连接对端服务器的属性结构体。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败, 请参见 错误码 。

【需求】

- 头文件: `cvi_comm_ipcmsg.h`、`cvi_ipcmsg.h`
- 库文件: `cvilink.a`

【注意】

Service 可以添加多个, 但不同的 service 不能使用相同的端口号。client 跟 service 是通过相同的端口号来通信的, 因此一个 service 只能对应一个 client。

【举例】

无

【相关主题】[CVI_IPCMSG_DelService](#)

5.1.5 CVI_IPCMSG_DelService

【描述】

删除服务。

【语法】

```
CVI_S32 CVI_IPCMSG_DelService(const CVI_CHAR *pszServiceName);
```

【参数】

参数名称	描述	输入/输出
pszServiceName	服务的名称指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见 错误码 。

【需求】

- 头文件：cvi_comm_ipcmsg.h、cvi_ipcmsg.h
- 库文件：cvilink.a

【注意】

Service 可以添加多个，但不同的 service 不能使用相同的端口号。client 跟 service 是通过相同的端口号来通信的，因此一个 service 只能对应一个 client。

【举例】

无

【相关主题】[CVI_IPCMSG_AddService](#)

5.1.6 CVI_IPCMSG_TryConnect

【描述】

非阻塞方式建立连接。

【语法】

```
CVI_S32 CVI_IPCMSG_TryConnect(CVI_S32 *ps32Id, const CVI_CHAR *pszServiceName,
    CVI_IPCMSG_HANDLE_FN_PTR pfnMessageHandle);
```

【参数】

参数名称	描述	输入/输出
ps32Id	消息通信 ID 指针。	输出
pszServiceName	服务名称指针。	输入
pfnMessageHandle	消息处理回调函数。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见 错误码 。

【需求】

- 头文件：cvi_comm_ipcmsg.h、cvi_ipcmsg.h
- 库文件：cvilink.a

【注意】

无

【举例】

无

【相关主题】

[CVI_IPCMSG_Connect](#)

5.1.7 CVI_IPCMSG_Connect

【描述】

阻塞方式建立连接。

【语法】

```
CVI_S32 CVI_IPCMSG_Connect(CVI_S32 *ps32Id, const CVI_CHAR *pszServiceName,  
    CVI_IPCMSG_HANDLE_FN_PTR pfnMessageHandle);
```

【参数】

参数名称	描述	输入/输出
ps32Id	消息通信 ID 指针。	输出
pszServiceName	服务名称指针。	输入
pfnMessageHandle	消息处理回调函数。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见 错误码 。

【需求】

- 头文件：cvi_comm_ipcmsg.h、cvi_ipcmsg.h
- 库文件：cvilink.a

【注意】

无

【举例】

无

【相关主题】

[CVI_IPCMSG_Disconnect](#)

5.1.8 CVI_IPCMSG_Disconnect

【描述】

断开连接。

【语法】

```
CVI_S32 CVI_IPCMSG_Disconnect(CVI_S32 s32Id);
```

【参数】

参数名称	描述	输入/输出
s32Id	消息通信 ID。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见 错误码 。

【需求】

- 头文件：cvi_comm_ipcmsg.h、cvi_ipcmsg.h
- 库文件：cvilink.a

【注意】

无

【举例】

无

【相关主题】

[CVI_IPCMSG_Connect](#) [CVI_IPCMSG_TryConnect](#)

5.1.9 CVI_IPCMSG_IsConnected

【描述】

消息通信是否连接状态。

【语法】

```
CVI_BOOL CVI_IPCMSG_IsConnected(CVI_S32 s32Id);
```

【参数】

参数名称	描述	输入/输出
s32Id	消息通信 ID。	输入

【返回值】

返回值	描述
CVI_TRUE	连接状态。
CVI_FALSE	非连接状态。

【需求】

- 头文件：cvi_comm_ipcmsg.h、cvi_ipcmsg.h
- 库文件：cvilink.a

【注意】

无

【举例】

无

【相关主题】CVI_IPCMSG_Connect CVI_IPCMSG_TryConnect

5.1.10 CVI_IPCMSG_SendAsync

【描述】

发送异步消息。这个接口是非阻塞接口，发送消息到对端后就返回了，不会等待消息命令的处理过程。如果调用此接口发送回复消息，则不需要对端回复，否则对端必须回复。

【语法】

```
CVI_S32 CVI_IPCMSG_SendAsync(CVI_S32 s32Id, CVI_IPCMSG_MESSAGE_S *pstMsg,
    CVI_IPCMSG_RESPHANDLE_FN_PTR pfnRespHandle);
```

【参数】

参数名称	描述	输入/输出
s32Id	消息服务 ID。	输入
pstMsg	消息指针。	输入
pfnRespHandle	消息回复处理函数。在发送回复消息时可以为 NULL，其他情况不允许为 NULL。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见 错误码 。

【需求】

- 头文件：cvi_comm_ipcmsg.h、cvi_ipcmsg.h
- 库文件：cvilink.a

【注意】

无

【举例】

无

【相关主题】CVI_IPCMSG_SendSync

5.1.11 CVI_IPCMSG_SendSync

【描述】

发送同步消息。这个接口会阻塞等待对端消息命令处理完成后再返回。

【语法】

```
CVI_S32 CVI_IPCMSG_SendSync(CVI_S32 s32Id, CVI_IPCMSG_MESSAGE_S *pstMsg,  
    CVI_IPCMSG_MESSAGE_S **ppstMsg, CVI_S32 s32TimeoutMs);
```

【参数】

参数名称	描述	输入/输出
s32Id	消息服务 ID。	输入
pstMsg	消息指针。	输入
ppstMsg	回复消息的指针的指针。	输出
s32TimeoutMs	超时时间。单位：ms。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见 错误码 。

【需求】

- 头文件：cvi_comm_ipcmsg.h、cvi_ipcmsg.h
- 库文件：cvilink.a

【注意】

本接口超时的情况下，内部会调用 CVI_IPCMSG_DestroyMessage 将 *ppstMsg（回复消息）销毁一次，由于同一个消息不能重复销毁，所以本接口超时退出后不必再做销毁回复消息的处理。

【举例】

无

【相关主题】

[CVI_IPCMSG_SendAsync](#)

5.1.12 CVI_IPCMSG_Run

【描述】

消息处理函数。

【语法】

```
CVI_VOID CVI_IPCMSG_Run(CVI_S32 s32Id);
```

【参数】

参数名称	描述	输入/输出
s32Id	消息服务 ID。	输入

【返回值】

返回值	描述
CVI_VOID	成功。

【需求】

- 头文件：cvi_comm_ipcmsg.h、cvi_ipcmsg.h
- 库文件：cvilink.a

【注意】

无

【举例】

无

【相关主题】

无

5.1.13 CVI_IPCMSG_SendOnly

【描述】

仅发送消息给对端，不接收对端的返回值。

【语法】

```
CVI_S32 CVI_IPCMSG_SendOnly(CVI_S32 s32Id, CVI_IPCMSG_MESSAGE_S *pstRequest);
```

【参数】

参数名称	描述	输入/输出
s32Id	消息服务 ID。	输入
pstRequest	消息结构体的指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见 错误码 。

【需求】

- 头文件：cvi_comm_ipcmsg.h、cvi_ipcmsg.h
- 库文件：cvilink.a

【注意】

无

【举例】

无

【相关主题】

无

5.2 数据类型

5.2.1 CVI_IPCMSG_MAX_CONTENT_LEN

【说明】

定义消息体最大长度。

【定义】

```
#define CVI_IPCMSG_MAX_CONTENT_LEN (1024)
```

【成员】

无

【注意事项】

无

【相关数据类型及接口】

无

5.2.2 CVI_IPCMSG_PRIVDATA_NUM

【说明】

定义消息体中私有数据最大个数。

【定义】

```
#define CVI_IPCMSG_PRIVDATA_NUM (8)
```

【成员】

无

【注意事项】

无

【相关数据类型及接口】

无

5.2.3 CVI_IPCMSG_INVALID_MSGID

【说明】

定义无效消息 ID。

【定义】

```
#define CVI_IPCMSG_INVALID_MSGID (0xFFFFFFFFFFFFFFFF)
```

【注意事项】

无

【相关数据类型及接口】

无

5.2.4 CVI_IPCMSG_MAX_SERVICENAME_LEN

【说明】

定义服务名称的最大长度。

【定义】

```
#define CVI_IPCMSG_MAX_SERVICENAME_LEN (16)
```

【注意事项】

无

【相关数据类型及接口】

无

5.2.5 CVI_IPCMSG_CONNECT_S

【说明】

定义连接对端服务器的结构体。

【定义】

```
typedef struct cviIPCMSG_CONNECT_S {  
    CVI_U32 u32RemoteId;  
    CVI_U32 u32Port;  
    CVI_U32 u32Priority;  
} CVI_IPCMSG_CONNECT_S;
```

【成员】

成员名称	描述
u32RemoteId	标示连接远端 CPU 的枚举值。0: 主 CPU, 运行主要应用程序的 CPU; 1: 从 CPU, 运行媒体驱动的那个 CPU。
u32Port	消息通信用的自定义 port 号。
u32Priority	消息传递的优先级。取值范围: 0: 普通优先级; 1: 高优先级。默认为 0

【注意事项】

如果需要采用高优先级的消息传输, 那么发送和接收端的 u32Priority 都需要指定为 1; 高优先级的消息采用中断的方式传输消息, 如果发送高优先级的频率很高, 可能会造成系统整体性能的下降。

【相关数据类型及接口】

无

5.2.6 CVI_IPCMSG_MESSAGE_S

【说明】

定义消息结构体。

【定义】

```
/**Message structure*/
typedef struct cviIPCMSG_MESSAGE_S {
    CVI_BOOL bIsResp; /**<Identify the response messgae*/
    CVI_U64 u64Id;      /**<Message ID*/
    CVI_U32 u32Module;  /**<Module ID, user-defined*/
    CVI_U32 u32CMD;     /**<CMD ID, user-defined*/
    CVI_S32 s32RetVal;  /**<Retrun Value in response message*/
    CVI_U32 u32BodyLen; /**<Length of pBody*/
    /**<Private data, can be modify directly after ::CVI_IPCMSG_CreateMessage
    or ::CVI_IPCMSG_CreateRespMessage*/
    CVI_S32 as32PrivData[CVI_IPCMSG_PRIVDATA_NUM];
    CVI_VOID *pBody; /**<Message body*/
#ifdef __arm__
    CVI_U32 u32VirAddrPadding;
#endif
} CVI_IPCMSG_MESSAGE_S;
```

【成员】

成员名称	描述
bIsResp	标示该消息是否回复消息：CVI_TRUE：回复；CVI_FALSE：不回复
u64Id	消息 ID。
u32Module	模块 ID。
u32CMD	消息 ID。
s32RetVal	返回值。
as32PrivData	私有数据。
pBody	消息体指针。

【注意事项】

无

【相关数据类型及接口】

无

5.2.7 CVI_IPCMSG_HANDLE_FN_PTR

【说明】

定义消息回复处理函数。

【定义】

```
typedef void (*CVI_IPCMSG_HANDLE_FN_PTR)(CVI_S32 s32Id, CVI_IPCMSG_MESSAGE_S_
→*pstMsg);
```

【成员】

成员名称	描述
s32Id	消息服务 ID。
pstMsg	消息体指针。

【注意事项】

无

【相关数据类型及接口】

无

5.2.8 CVI_IPCMSG_RESPHANDLE_FN_PTR

【说明】

定义定义 I80 指令

【定义】

```
typedef void (*CVI_IPCMSG_RESPHANDLE_FN_PTR)(CVI_IPCMSG_MESSAGE_S *pstMsg);
```

【成员】

成员名称	描述
pstMsg	消息体指针。

【注意事项】

无

【相关数据类型及接口】

无

5.3 错误码

错误代码	宏定义	描述
0x1901	CVI_IPCMSG_EINVAL	参数设置无效
0x1902	CVI_IPCMSG_ETIMEOUT	函数运行超时
0x1903	CVI_IPCMSG_ENOOP	IPC 驱动加载失败
0x1904	CVI_IPCMSG_EINTER	内部错误
0x1905	CVI_IPCMSG_ENULL_PTR	空指针
0x00000000	CVI_SUCCESS	成功
0xFFFFFFFF	CVI_FAILURE	失败

6 DATAFIFO

6.1 API 参考

该功能模块提供以下 API:

- `CVI_DATAFIFO_Open` : 打开数据通路。
 - `CVI_DATAFIFO_OpenByAddr` : 通过物理地址打开通路。
 - `CVI_DATAFIFO_Close` : 关闭通路。
 - `CVI_DATAFIFO_Read` : 读取数据。
 - `CVI_DATAFIFO_Write` : 写入数据。
 - `CVI_DATAFIFO_CMD` : 其他控制命令。
-

6.1.1 CVI_DATAFIFO_Open

【描述】

创建消息。

【语法】

```
CVI_S32 CVI_DATAFIFO_Open(CVI_DATAFIFO_HANDLE *Handle, CVI_DATAFIFO_
↪PARAMS_S *pstParams);
```

【参数】

参数名称	描述	输入/输出
Handle	数据通路句柄。	输出
pstParams	数据通路参数指针。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败, 请参见 错误码 。

【需求】

- 头文件: cvi_comm_ipcmmsg.h、cvi_ipcmmsg.h
- 库文件: cvilink.a

【注意】

无

【举例】

无

【相关主题】

[CVI_DATAFIFO_Close](#)

6.1.2 CVI_DATAFIFO_OpenByAddr

【描述】

通过物理地址打开数据通路。

【语法】

```
CVI_S32 CVI_DATAFIFO_OpenByAddr(CVI_DATAFIFO_HANDLE *Handle,
    CVI_DATAFIFO_PARAMS_S *pstParams, CVI_U64 u64PhyAddr);
```

【参数】

参数名称	描述	输入/输出
Handle	数据通路句柄。	输出
pstParams	数据通路参数指针。	输入
u32PhyAddr	数据缓存的物理地址。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败, 请参见 错误码 。

【需求】

- 头文件: cvi_comm_ipcmmsg.h、cvi_ipcmmsg.h
- 库文件: cvilink.a

【注意】

无

【举例】

无

【相关主题】[CVI_DATAFIFO_Close](#)

6.1.3 CVI_DATAFIFO_Close

【描述】

关闭数据通路。

【语法】

`CVI_S32 CVI_DATAFIFO_Close(CVI_DATAFIFO_HANDLE Handle);`

【参数】

参数名称	描述	输入/输出
Handle	数据通路句柄。	输入

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见 错误码 。

【需求】

- 头文件：cvi_comm_ipcmmsg.h、cvi_ipcmmsg.h
- 库文件：cvilink.a

【注意】

无

【举例】

无

【相关主题】[CVI_DATAFIFO_Open](#) [CVI_DATAFIFO_OpenByAddr](#)

6.1.4 CVI_DATAFIFO_Read

【描述】

读取数据。

【语法】

```
CVI_S32 CVI_DATAFIFO_Read(CVI_DATAFIFO_HANDLE Handle, CVI_VOID **ppData);
```

【参数】

参数名称	描述	输入/输出
Handle	数据通路句柄。	输入
ppData	读取数据指针的指针。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见 错误码 。

【需求】

- 头文件：cvi_comm_ipcmmsg.h、cvi_ipcmmsg.h
- 库文件：cvilink.a

【注意】

无

【举例】

无

【相关主题】

CVI_DATAFIFO_Write CVI_DATAFIFO_CMD

6.1.5 CVI_DATAFIFO_Write

【描述】

写入数据。

【语法】

```
CVI_S32 CVI_DATAFIFO_Write(CVI_DATAFIFO_HANDLE Handle, CVI_VOID *pData);
```

【参数】

参数名称	描述	输入/输出
Handle	数据通路句柄。	输入
pData	写入的数据。	输出

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见 错误码 。

【需求】

- 头文件：cvi_comm_ipcmmsg.h、cvi_ipcmmsg.h
- 库文件：cvilink.a

【注意】

无

【举例】

无

【相关主题】

[CVI_DATAFIFO_Read CVI_DATAFIFO_CMD](#)

6.1.6 CVI_DATAFIFO_CMD

【描述】

其他操作。

【语法】

```
CVI_S32 CVI_DATAFIFO_CMD(CVI_DATAFIFO_HANDLE Handle, CVI_DATAFIFO_CMD_EN_
→enCMD, CVI_VOID *pArg);
```

【参数】

参数名称	描述	输入/输出
Handle	数据通路句柄。	输入
pArg	参数，详见【注意】。	输入/输出

【返回值】

返回值	描述
0	成功。
非 0	失败，请参见 错误码 。

【需求】

- 头文件: `cvi_comm_ipcmmsg.h`、`cvi_ipcmmsg.h`
- 库文件: `cvilink.a`

【注意】

`DATAFIFO_CMD_GET_PHY_ADDR`: 返回 `DATAFIFO` 的物理地址, `CVI_U32` 类型。`DATAFIFO_CMD_READ_DONE`: 读端使用完数据后, 需要调用这个更新读端的头尾指针。无返回值, 参数可以为 `CVI_NULL`。`DATAFIFO_CMD_WRITE_DONE`: 写端写完数据后, 需要调用这个更新写端的写尾指针。无返回值, 参数可以为 `CVI_NULL`。`DATAFIFO_CMD_SET_DATA_RELEASE_CALLBACK`: 数据释放回调函数。`DATAFIFO_CMD_GET_AVAIL_WRITE_LEN`: 返回可写入的数据个数, `CVI_U32` 类型。`DATAFIFO_CMD_GET_AVAIL_READ_LEN`: 返回可读取的数据个数, `CVI_U32` 类型。

【举例】

无

【相关主题】

无

6.2 数据类型

6.2.1 CVI_DATAFIFO_HANDLE

【说明】

定义 `DATAFIFO` 的句柄。

【定义】

```
typedef CVI_U64 CVI_DATAFIFO_HANDLE;
```

【成员】

无

【注意事项】

无

【相关数据类型及接口】

无

6.2.2 CVI_DATAFIFO_INVALID_HANDLE

【说明】

定义数据通路无效句柄。

【定义】

```
#define CVI_DATAFIFO_INVALID_HANDLE (-1)
```

【注意事项】

无

【相关数据类型及接口】

无

6.2.3 CVI_DATAFIFO_RELEASESTREAM_FN_PTR

【说明】

定义数据通路码流释放函数。

【定义】

```
typedef void (*CVI_DATAFIFO_RELEASESTREAM_FN_PTR)(void *pStream);
```

【注意事项】

无

【相关数据类型及接口】

无

6.2.4 CVI_DATAFIFO_OPEN_MODE_E

【说明】

定义数据通路打开模式。

【定义】

```
typedef enum cviDATAFIFO_OPEN_MODE_E {  
    DATAFIFO_READER,  
    DATAFIFO_WRITER  
} CVI_DATAFIFO_OPEN_MODE_E;
```

【成员】

成员名称	描述
DATAFIFO_READER	读出角色，只读取数据。
DATAFIFO_WRITER	写入角色，只写入数据。

【注意事项】

无

【相关数据类型及接口】

无

6.2.5 CVI_DATAFIFO_PARAMS_S

【说明】

定义数据通路配置参数。

【定义】

```
/** DATAFIFO parameters */
typedef struct cviDATAFIFO_PARAMS_S {
    CVI_U32 u32EntriesNum; /**< The number of items in the ring buffer*/
    CVI_U32 u32CacheLineSize; /**< Item size*/
    CVI_BOOL bDataReleaseByWriter; /**< Whether the data buffer release by writer*/
    CVI_DATAFIFO_OPEN_MODE_E enOpenMode; /**< READER or WRITER*/
} CVI_DATAFIFO_PARAMS_S;
```

【成员】

成员名称	描述
u32EntriesNum	循环 Buffer 的数据个数
u32CacheLineSize	每个数据项的大小。
bDataReleaseByWriter	是否需要写入者释放数据。
enOpenMode	打开通路的角色。

【注意事项】

Datafifo 每次读写都是固定单位长度 (item), 每个 item 的长度, 多少个 item 都是由用户指定, 由于需要保留一个 item 辅助 buf 管理, 实际可用的个数会少一个。

【相关数据类型及接口】

无

6.2.6 CVI_DATAFIFO_CMD_E

【说明】

定义数据通路的控制类型。

【定义】

```
/** DATAFIFO advanced function */
typedef enum cviDATAFIFO_CMD_E {
    DATAFIFO_CMD_GET_PHY_ADDR, /**<Get the physic address of ring buffer*/
    /**<When the read buffer read over, the reader should
    call this function to notify the writer*/
    DATAFIFO_CMD_READ_DONE,
    DATAFIFO_CMD_WRITE_DONE, /**<When the writer buffer is write done, the writer should
    call this function*/
    /**<When bDataReleaseByWriter is CVI_TRUE, the writer should call this
    to register release callback*/
    DATAFIFO_CMD_SET_DATA_RELEASE_CALLBACK,
    DATAFIFO_CMD_GET_AVAIL_WRITE_LEN, /**<Get available write length*/
    DATAFIFO_CMD_GET_AVAIL_READ_LEN, /**<Get available read length*/
    DATAFIFO_CMD_SHOW_POINTER
} CVI_DATAFIFO_CMD_E;
```

【成员】

成员名称	描述
DATAFIFO_CMD_GET_PHY_ADDR	获取数据通路的物理地址。
DATAFIFO_CMD_READ_DONE	通知读取完成。
DATAFIFO_CMD_WRITE_DONE	通知写入完成。
DATAFIFO_CMD_SET_DATA_RELEASE_CALLBACK	设置数据释放回调函数。
DATAFIFO_CMD_GET_AVAIL_WRITE_LEN	获取可以写入数据的长度。
DATAFIFO_CMD_GET_AVAIL_READ_LEN	获取可以读取数据的长度。

【注意事项】

无

【相关数据类型及接口】

无

6.3 错误码

错误代码	宏定义	描述
0x00000000	CVI_SUCCESS	成功
0xFFFFFFFF	CVI_FAILURE	失败