CV180X & CV181X Software CviSysLink User Guide

Version: 1.2.0

Release date: 2023/09

Contents

Revision History

Revision	Date	Description
xxx	xxx	xxx

1 Disclaimer

Terms and Conditions

The document and all information contained herein remain the CVITEK Co., Ltd's (
"CVITEK") confidential information, and should not disclose to any third party or use it in
any way without CVITEK's prior written consent. User shall be liable for any damage and
loss caused by unauthority use and disclosure.

CVITEK reserves the right to make changes to information contained in this document at any
time and without notice.

All information contained herein is provided in "AS IS" basis, without warranties of any kind,
expressed or implied, including without limitation mercantability, non-infringement and fitness
for a particular purpose. In no event shall CVITEK be liable for any third party's software
provided herein, User shall only seek remedy against such third party. CVITEK especially
claims that CVITEK shall have no liable for CVITEK's work result based on Customer's
specification or published shandard.

Contact Us

Address Building 1, Yard 9, FengHao East Road, Haidian District, Beijing, 100094,
China

Building T10, UpperCoast Park, Huizhanwan, Zhancheng Community, Fuhai
Street, Baoan District, Shenzhen, 518100, China

Phone +86-10-57590723 +86-10-57590724

Website https://www.sophgo.com/

Forum https://developer.sophgo.com/forum/index.html

2 Document Overview

This document is mainly used to guide users to use the multimedia dual-system inter-core communication module to improve development efficiency; it describes the main functions and development reference of IPCMSG and DATAFIFO. Users can use IPCMSG and DATAFIFO to solve the dual-core message communication and data transfer problems. There is no guarantee that this document will be up to date, so please use the documentation in the latest released SDK.

2.1 Objective

Guides users to develop using the Multimedia Dual System Inter-core Communication Module API.

2.2 Scope of application

This document covers the dual-system inter-core communication module in the SOPHGO Multimedia Software Development SDK. Applicable groups: technical support engineers, software development engineers

2.3 Terms and definitions

Table 2.1: List of terms and definitions

Serial No.	Term	Description of definitions
1.	IPCM	The driver layer for cross-core message communication, which implements message sending and receiving through interrupts and shared memory, provides a standard device read/write interface to the upper layers.
2.	IPCMSG	The package of IPCM provides the user with Message interface and realizes the information transfer between the two cores by sending and receiving of Message.
3.	Physical Address	Absolute address on the DDR, visible on both ends of the dual-core dual system.
4.	Virtual Address	Each system handles this differently, and on Linux, virtual addresses are only visible within the same process.
5.	Ring Buffer	In order to facilitate address offset, the data stored in the Ring Buffer requires a fixed length. Therefore, in general, the Ring Buffer only stores Pointers to stream data, but not stream data.

3 System Overview

CviSysLink contains two modules: IPCMSG and DATAFIFO. The former is used for cross-core communication and the latter is used for cross-core data transfer.

3.1   System Architecture

Figure 3.1 shows the overall architecture of CviSysLink.

There are two main modules that make up the program:

- IPCMSG

- DATAFIFO
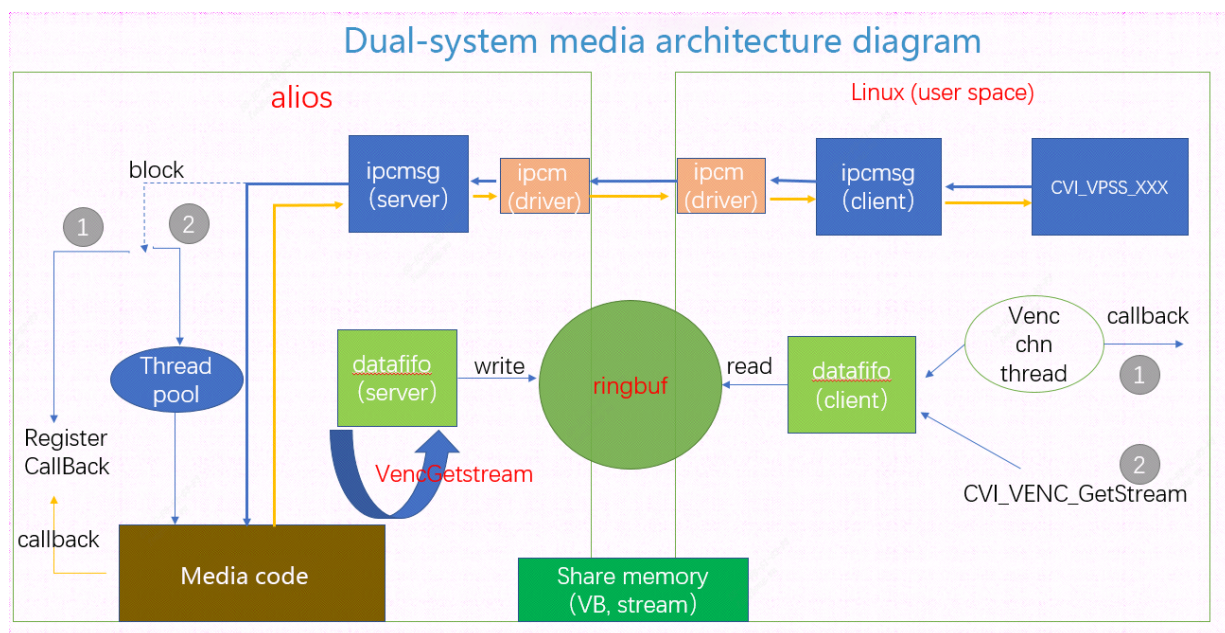
Fig. 3.1: CviSysLink System Architecture

3.1.1   IPCMSG

This module is designed to solve the problem of communication between modules deployed in two systems in a dual-core dual-system environment, and the amount of data communication should not be too large (no more than 1024 bytes at a time).

Figure 3.2 is IPCMSG flow chart.
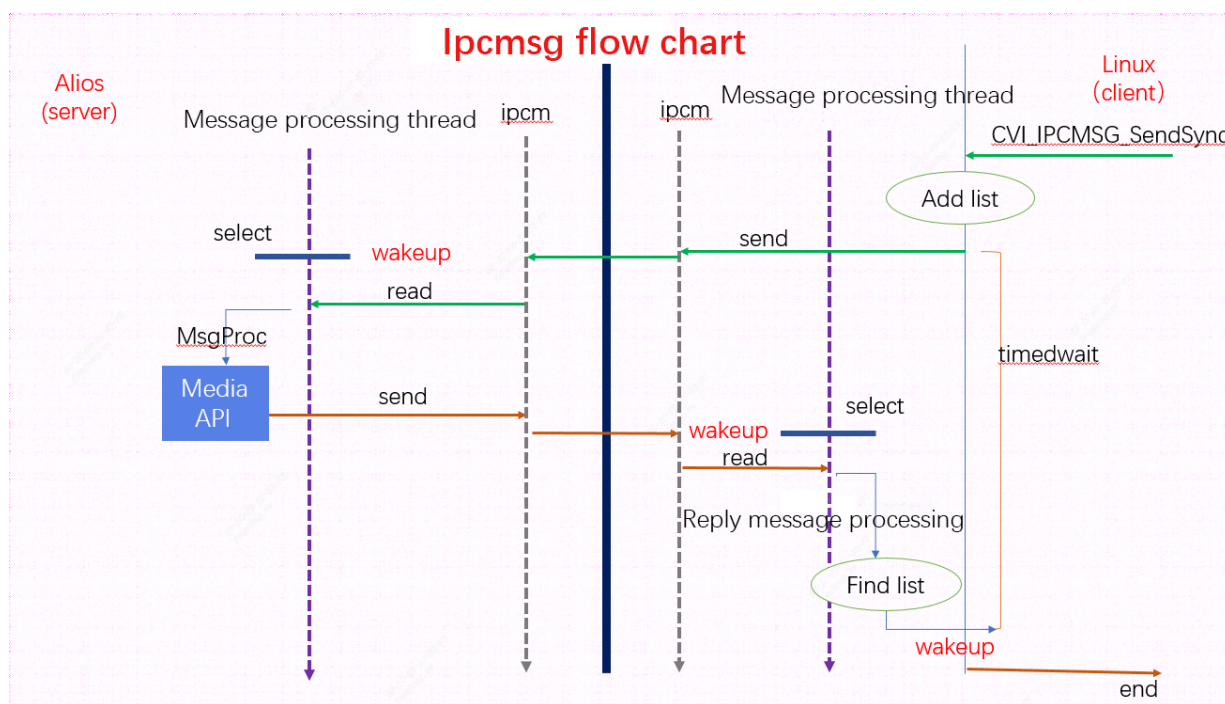
Fig. 3.2: IPCMSG flowchart

3.1.2   DATAFIFO

DATAFIFO provides a mechanism to deal with frequent transfer of large amounts of data (such as codec) where IPMSG is not efficient or feasible. The general process is shown in *Figure 3.3*. The pointer to the stream data is maintained in the DATAFIFO. After receiving the stream data, the writer sends the pointer to the RingBuf in the DATAFIFO. When the reader wants to read data, DATAFIFO passes the stored data pointer to the reader. In dual-core data transmission, one end can only write, and the other end can only read.
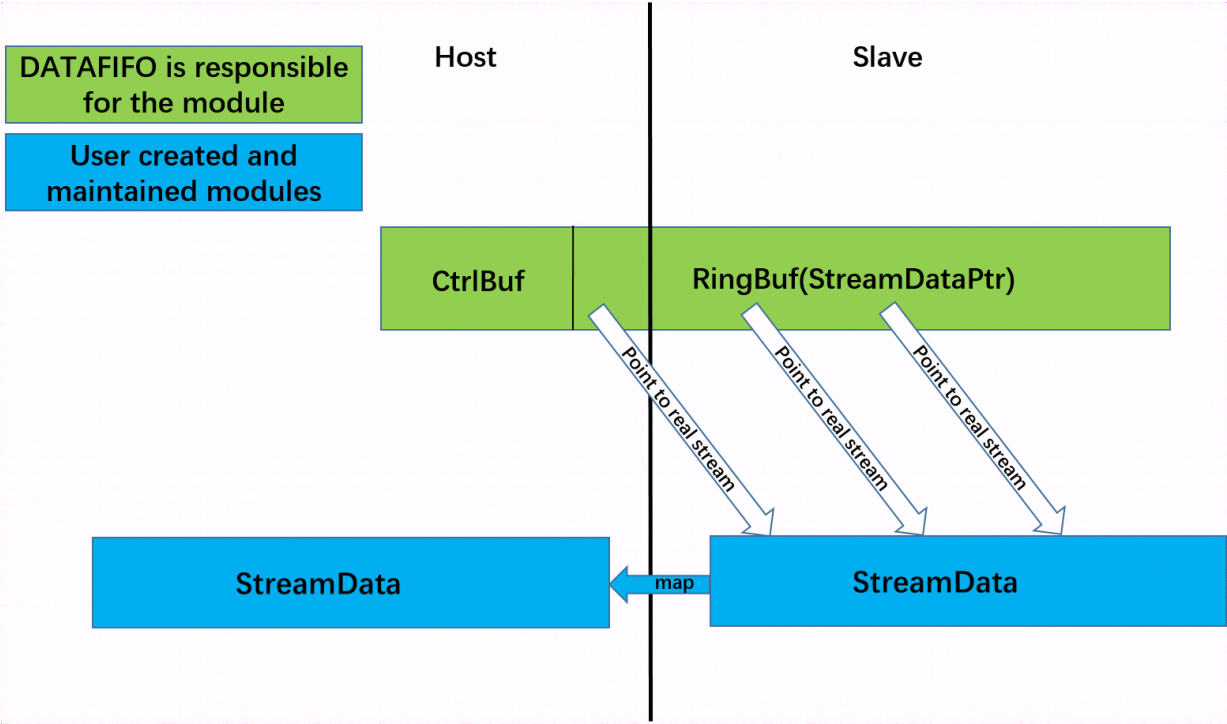
Fig. 3.3: Dual-core data transmission diagram

4 Function Overview

4.1 Function overview

4.1.1 IPCMSG

IPCMSG module contains the functions of message creation and destruction, service addition and deletion, connection establishment, disconnection, message sending, etc. Connection establishment supports blocking and non-blocking connection establishment. Message sending Supports synchronous message sending and asynchronous message sending. Synchronous messages support a timeout mechanism. Whether synchronous or asynchronous messages are sent, if the reply message is longer than 60 seconds, the reply message is discarded.

4.1.2 DATAFIFO

Module for cross-core data transfer, data first in, first out. Since the two systems transmit data through memory sharing, one end is responsible for writing data, and the other end is responsible for reading data. DataFifo maintains four Pointers of write head, write tail, read head, and read tail at both ends.Perform operations in a circular Buffer on one end to complete data transfer. DATAFIFO mainly includes the opening and closing of channels, the writing and reading of data, and other control commands.

5 IPCMSG

5.1   API Reference

The function module provides the following API:

- *CVI_IPCMSG_CreateMessage* : Create a message.
- *CVI_IPCMSG_CreateRespMessage* : Create a reply message.
- *CVI_IPCMSG_DestroyMessage* : Destroy the message.
- *CVI_IPCMSG_AddService* : Add a service.
- *CVI_IPCMSG_DelService* : Delete the service.
- *CVI_IPCMSG_TryConnect* : Non-blocking connection.
- *CVI_IPCMSG_Connect* : Block mode to establish connection.
- *CVI_IPCMSG_Disconnect* : Disconnect the connection.
- *CVI_IPCMSG_IsConnected* : Get the connection status.
- *CVI_IPCMSG_SendAsync* : Send asynchronous messages.
- *CVI_IPCMSG_SendSync* : Send synchronous messages.
- *CVI_IPCMSG_Run* : Message handler function
- *CVI_IPCMSG_SendOnly* : function that sends only messages

5.1.1   CVI_IPCMSG_CreateMessage

【Description】

Create a message.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
u32Module	Module ID. Created by the user to distinguish between different messages for different modules.	Input
u32CMD	u32CMD command ID. Created by the user to distinguish between different commands under the same module.	Input
pBody	Message body pointer.	Input
u32BodyLen	Message body size.	Input

【Return Value】

Return Value	Description
CVI_IPCMSG_MESSAGE_S *	Message structure pointer.
NULL	Message creation failed.

【Requirement】

- Header files: cvi_comm_ipcmsg.h、cvi_ipcmsg.h

- Library files: cvilink.a

【Note】

None

【Example】

None

【Related Topic】

CVI_IPCMSG_DestroyMessage

5.1.2 CVI_IPCMSG_CreateRespMessage

【Description】

Create a reply message.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
pstRequest	Pointer to the request message.	Input
s32RetVal	Reply Return value.	Input
pBody	Pointer to reply message in the message body.	Input
u32BodyLen	The body size of the reply message.	Input

【Return Value】

Return Value	Description
CVI_IPCMSG_MESSAGE_S *	Message structure pointer.
NULL	Message creation failed.

【Requirement】

- Header files: cvi_comm_ipcmsg.h、cvi_ipcmsg.h

- Library files: cvilink.a

【Note】

None

【Example】

None

【Related Topic】

CVI_IPCMSG_DestroyMessage

5.1.3   CVI_IPCMSG_DestroyMessage

【Description】

Destroy the message.

【Syntax】

```
CVI_VOID CVI_IPCMSG_DestroyMessage(CVI_IPCMSG_MESSAGE_S *pstMsg);
```

【Parameter】

Parameter Name	Description	Input/Output
pstMsg	Message pointer.	Input

【Return Value】

Return Value	Description
CVI_VOID	None

【Requirement】

- Header files: cvi_comm_ipcmsg.h、cvi_ipcmsg.h

- Library files: cvilink.a

【Note】

None

【Example】

None

【Related Topic】

CVI_IPCMSG__CreateMessage CVI_IPCMSG__CreateRespMessage

5.1.4 CVI_IPCMSG_AddService

【Description】

Add a service.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
pszServiceName	Name pointer to the service.	Input
pstConnectAttr	The property structure that connects to the peer server.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure, please refer *Error Codes*.

【Requirement】

- Header files: cvi_comm_ipcmsg.h、cvi_ipcmsg.h

- Library files: cvilink.a

【Note】

The Service can add multiple, but different Service cannot use the same port number. The client and service communicate through the same port number, so a service can only correspond to one client.

【Example】

None

【Related Topic】

CVI_IPCMSG_DelService

5.1.5   CVI_IPCMSG_DelService

【Description】

Delete the service.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
pszServiceName	Name pointer to the service.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure, please refer *Error Codes*.

【Requirement】

- Header files: cvi_comm_ipcmsg.h、cvi_ipcmsg.h

- Library files: cvilink.a

【Note】

The Service can add multiple, but different Service cannot use the same port number. The client and service communicate through the same port number, so a service can only correspond to one client.

【Example】

None

【Related Topic】

CVI_IPCMSG_AddService

5.1.6 CVI_IPCMSG_TryConnect

【Description】

Non-blocking connection.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
ps32Id	Message communication ID pointer.	Output
pszServiceName	Name pointer to the service.	Input
pfnMessageHandle	Message handling callback function.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure, please refer *Error Codes*.

【Requirement】

- Header files: cvi_comm_ipcmsg.h、cvi_ipcmsg.h

- Library files: cvilink.a

【Note】

None

【Example】

None

【Related Topic】

CVI_IPCMSG_Connect

5.1.7 CVI_IPCMSG_Connect

【Description】

Block mode to establish connection.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
ps32Id	Message communication ID pointer.	Output
pszServiceName	Name pointer to the service.	Input
pfnMessageHan-dle	Message handling callback function.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure, please refer *Error Codes*.

【Requirement】

- Header files: cvi_comm_ipcmsg.h、cvi_ipcmsg.h

- Library files: cvilink.a

【Note】

None

【Example】

None

【Related Topic】

CVI_IPCMSG_Disconnect

5.1.8 CVI_IPCMSG_Disconnect

【Description】

Disconnect the connection.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
s32Id	Message communication ID.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure, please refer *Error Codes*.

【Requirement】

- Header files: cvi_comm_ipcmsg.h、cvi_ipcmsg.h

- Library files: cvilink.a

【Note】

None

【Example】

None

【Related Topic】

CVI_IPCMSG_Connect CVI_IPCMSG_TryConnect

5.1.9   CVI_IPCMSG_IsConnected

【Description】

Whether the message communication is connected.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
s32Id	Message communication ID.	Input

【Return Value】

Return Value	Description
CVI_TRUE	Connection status.
CVI_FALSE	Disconnected status.

【Requirement】

- Header files: cvi_comm_ipcmsg.h、cvi_ipcmsg.h

- Library files: cvilink.a

【Note】

None

【Example】

None

【Related Topic】

CVI_IPCMSG_Connect CVI_IPCMSG_TryConnect

5.1.10   CVI_IPCMSG_SendAsync

【Description】

Send asynchronous messages. This is a non-blocking interface that sends a message to the peer and then returns without waiting for the message command to be processed. If this interface is invoked to send a reply message, the peer does not need to reply, otherwise the peer must reply.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
s32Id	Message service ID.	Input
pstMsg	Message pointer.	Input
pfnRespHandle	Message reply handler function. Can be NULL when sending a reply message, but otherwise is not allowed.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure, please refer *Error Codes*.

【Requirement】

- Header files: cvi_comm_ipcmsg.h、cvi_ipcmsg.h

- Library files: cvilink.a

【Note】

None

【Example】

None

【Related Topic】

CVI_IPCMSG_SendSync

5.1.11   CVI_IPCMSG_SendSync

【Description】

Send synchronous messages. This interface blocks until the message command is processed and then returns.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
s32Id	Message service ID.	Input
pstMsg	Message pointer.	Input
ppstMsg	A pointer to the reply message pointer.	Output
s32TimeoutMs	The timeout period. Unit: ms.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure, please refer *Error Codes*.

【Requirement】

- Header files: cvi_comm_ipcmsg.h、cvi_ipcmsg.h

- Library files: cvilink.a

【Note】

In case of timeout, CVI_IPCMSG_DestroyMessage is called internally to destroy *ppstMsg (reply message). Since the same message cannot be destroyed repeatedly, the interface does not need to destroy the reply message after the timeout.

【Example】

None

【Related Topic】

CVI_IPCMSG_SendAsync

5.1.12 CVI_IPCMSG_Run

【Description】

Message handling function.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
s32Id	Message service ID.	Input

【Return Value】

Return Value	Description
CVI_VOID	Success.

【Requirement】

- Header files: cvi_comm_ipcmsg.h、cvi_ipcmsg.h
- Library files: cvilink.a

【Note】

None

【Example】

None

【Related Topic】

None

5.1.13 CVI_IPCMSG_SendOnly

【Description】

Only send the message to the peer, do not receive the return value of the peer.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
s32Id	Message Service ID.	Input
pstRequest	A pointer to a message structure.	Input

【Return Value】

Return Value	Description
0	Success.
Non 0	Failure, please refer *Error Codes*.

【Requirement】

- Header files: cvi_comm_ipcmsg.h、cvi_ipcmsg.h

- Library files: cvilink.a

【Note】

None

【Example】

None

【Related Topic】

None

5.2   Data Types

5.2.1   CVI_IPCMSG_MAX_CONTENT_LEN

【Description】

Define the maximum length of the message body.

【Define】

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

【Member】

None

【Note】

None

【Related Data types and Interfaces】

None

5.2.2 CVI_IPCMSG_PRIVDATA_NUM

【Description】

Define the maximum number of private data in the message body.

【Define】

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

【Member】

None

【Note】

None

【Related Data types and Interfaces】

None

5.2.3 CVI_IPCMSG_INVALID_MSGID

【Description】

Define invalid message ID.

【Define】

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

【Note】

None

【Related Data types and Interfaces】

None

5.2.4 CVI_IPCMSG_MAX_SERVICENAME_LEN

【Description】

Define the maximum length of the service name.

【Define】

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

【Note】

None

【Related Data types and Interfaces】

None

5.2.5 CVI_IPCMSG_CONNECT_S

【Description】

Define the structure to connect to the peer server.

【Define】

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

【Member】

Member Name	Description
u32RemoteId	Represents the enumeration value of the connected remote CPU. 0: main CPU, which is the CPU that runs the main application. 1: Slave CPU, the one running the media driver.
u32Port	Custom port number for message communication.
u32Priority	Priority of message delivery. Value range: 0: normal priority; 1: High priority. The default is 0.

【Note】

If high-priority message transmission is required, u32Priority at both sender and receiver needs to be specified as 1. High priority messages use interrupt to transmit messages. If the frequency of sending high priority messages is very high, it may cause the degradation of the overall performance of the system.

【Related Data types and Interfaces】

None

5.2.6   CVI_IPCMSG_MESSAGE_S

【Description】

Define the message structure.

【Define】

```
/**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;
```

【Member】

Member Name	Description
bIsResp	Indicates whether the message replies to the message. CVI_TRUE: reply; CVI_FALSE: Do not reply
u64Id	Message ID.
u32Module	Module ID.
u32CMD	Message ID.
s32RetVal	Return value.
as32PrivData	Private data.
pBody	Message body pointer.

【Note】

None

【Related Data types and Interfaces】

None

5.2.7 CVI_IPCMSG_HANDLE_FN_PTR

【Description】

Define the message reply handler function.

【Define】

```
typedef void (*CVI_IPCMSG_HANDLE_FN_PTR)(CVI_S32 s32Id, CVI_IPCMSG_MESSAGE_S␣
↪*pstMsg);
```

【Member】

Member Name	Description
s32Id	Message service ID.
pstMsg	Message body pointer.

【Note】

None

【Related Data types and Interfaces】

None

5.2.8 CVI_IPCMSG_RESPHANDLE_FN_PTR

【Description】

Define the I80 instruction.

【Define】

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

【Member】

Member Name	Description
pstMsg	Message body pointer.

【Note】

None

【Related Data types and Interfaces】

None

5.3   Error Codes

Errore code	Macro definition	Description
0x1901	CVI_IPCMSG_EINVAL	Invalid parameter setting
0x1902	CVI_IPCMSG_ETIMEOUT	Function timeout
0x1903	CVI_IPCMSG_ENOOP	Failed to load the IPC driver
0x1904	CVI_IPCMSG_EINTER	Internal error
0x1905	CVI_IPCMSG_ENULL_PTR	Null pointer
0x00000000	CVI_SUCCESS	Success
0xFFFFFFFF	CVI_FAILURE	Failure

6 DATAFIFO

6.1 API Reference

The function module provides the following API:

- *CVI_DATAFIFO_Open* : Open data path.

- *CVI_DATAFIFO_OpenByAddr* : Open path through the physical address.

- *CVI_DATAFIFO_Close* : Close path.

- *CVI_DATAFIFO_Read* : Read data.

- *CVI_DATAFIFO_Write* : Write data.

- *CVI_DATAFIFO_CMD* : Other control command.

6.1.1 CVI_DATAFIFO_Open

【Description】

　　Create a message.

【Syntax】

```
CVI_S32 CVI_DATAFIFO_Open(CVI_DATAFIFO_HANDLE *Handle, CVI_DATAFIFO_PARAMS_S␣
↪*pstParams);
```

【Parameter】

Parameter Name	Description	Input/Output
Handle	Data path handle.	Output
pstParams	Data path parameter pointer.	Input

【Return Value】

Return Value	Description
0	Success
Non 0	Failure, please refer *Error Codes*.

【Requirement】

- Header files: cvi_comm_ipcmsg.h、cvi_ipcmsg.h

- Library files: cvilink.a

【Note】

None

【Example】

None

【Related Topic】

CVI_DATAFIFO_Close

6.1.2   CVI_DATAFIFO_OpenByAddr

【Description】

Open the data path by physical address.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
Handle	Data path handle.	Output
pstParams	Data path parameter pointer.	Input
u32PhyAddr	Physical address of the data cache.	Input

【Return Value】

Return Value	Description
0	Success
Non 0	Failure, please refer *Error Codes*.

【Requirement】

- Header files: cvi_comm_ipcmsg.h、cvi_ipcmsg.h

- Library files: cvilink.a

【Note】

None

【Example】

None

【Related Topic】

CVI_DATAFIFO_Close

6.1.3 CVI_DATAFIFO_Close

【Description】

Close the data path.

【Syntax】

```
CVI_S32 CVI_DATAFIFO_Close(CVI_DATAFIFO_HANDLE Handle);
```

【Parameter】

Parameter Name	Description	Input/Output
Handle	Data path handle.	Input

【Return Value】

Return Value	Description
0	Success
Non 0	Failure, please refer *Error Codes*.

【Requirement】

- Header files: cvi_comm_ipcmsg.h、cvi_ipcmsg.h

- Library files: cvilink.a

【Note】

None

【Example】

None

【Related Topic】

CVI_DATAFIFO_Open CVI_DATAFIFO_OpenByAddr

6.1.4 CVI_DATAFIFO_Read

【Description】

Read data.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
Handle	Data path handle.	Input
ppData	A pointer that reads a data pointer.	Output

【Return Value】

Return Value	Description
0	Success。
Non 0	Failure, please refer *Error Codes*.

【Requirement】

- Header files: cvi_comm_ipcmsg.h、cvi_ipcmsg.h
- Library files: cvilink.a

【Note】

None

【Example】

None

【Related Topic】

CVI_DATAFIFO_Write CVI_DATAFIFO_CMD

6.1.5 CVI_DATAFIFO_Write

【Description】

Write data.

【Syntax】

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

【Parameter】

Parameter Name	Description	Input/Output
Handle	ata path handle.	Input
pData	Written data.	Output

【Return Value】

Return Value	Description
0	Success。
Non 0	Failure, please refer *Error Codes*.

【Requirement】

- Header files: cvi_comm_ipcmsg.h、cvi_ipcmsg.h

- Library files: cvilink.a

【Note】

None

【Example】

None

【Related Topic】

CVI_DATAFIFO_Read CVI_DATAFIFO_CMD

6.1.6 CVI_DATAFIFO_CMD

【Description】

Other control command.

【Syntax】

```
CVI_S32 CVI_DATAFIFO_CMD(CVI_DATAFIFO_HANDLE Handle, CVI_DATAFIFO_CMD_E enCMD,␣
↪CVI_VOID *pArg);
```

【Parameter】

Parameter Name	Description	Input/Output
Handle	Data path handle.	Input
pArg	Parameters, see 【Note】 for details.	Input/Output

【Return Value】

Return Value	Description
0	Success。
Non 0	Failure, please refer *Error Codes*.

【Requirement】

- Header files: cvi_comm_ipcmsg.h、cvi_ipcmsg.h

- Library files: cvilink.a

【Note】

DATAFIFO_CMD_GET_PHY_ADDR: Return the physical address of the DATAFIFO, CVI_U32 type. DATAFIFO_CMD_READ_DONE: After the reader has finished using the data, it needs to call the header and tail pointer of the update reader. No return value, and the argument can be CVI_NULL. DATAFIFO_CMD_WRITE_DONE: After the writer has finished writing the data, it needs to call the write end pointer of the update writer. No return value, and the argument can be CVI_NULL. DATAFIFO_CMD_SET_DATA_RELEASE_CALLBACK: Data release callback function. DATAFIFO_CMD_GET_AVAIL_WRITE_LEN: Return the number of data that can be written, CVI_U32 type. DATAFIFO_CMD_GET_AVAIL_READ_LEN : Return the number of data that can be read, CVI_U32 type.

【Example】

None

【Related Topic】

None

6.2   Data Types

6.2.1   CVI_DATAFIFO_HANDLE

【Description】

Define the DATAFIFO handle.

【Define】

```
typedef CVI_U64 CVI_DATAFIFO_HANDLE;
```

【Member】

None

【Note】

None

【Related Data types and Interfaces】

None

6.2.2 CVI_DATAFIFO_INVALID_HANDLE

【Description】

Define the data path invalid handle.

【Define】

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

【Note】

None

【Related Data types and Interfaces】

None

6.2.3 CVI_DATAFIFO_RELEASESTREAM_FN_PTR

【Description】

Define the data path code stream release function.

【Define】

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

【Note】

None

【Related Data types and Interfaces】

None

6.2.4 CVI_DATAFIFO_OPEN_MODE_E

【Description】

Define the data path open mode.

【Define】

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

【Member】

Member Name	Description
DATAFIFO_READER	Read out, only read data.
DATAFIFO_WRITER	Write, only write data.

【Note】

None

【Related Data types and Interfaces】

None

6.2.5   CVI_DATAFIFO_PARAMS_S

【Description】

Define the data path configuration parameters.

【Define】

```
/** 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;
```

【Member】

Member Name	Description
u32EntriesNum	The number of data in the loop Buffer
u32CacheLineSize	Size of each data item.
bDataReleaseByWriter	Whether the writer needs to release the data.
enOpenMode	The role of opening the path.

【Note】

DataFifo has a fixed unit length (item) for each read and write, the length and number of items for each item are specified by users,

Due to the need to retain an item to assist with buf management, the actual number available will be one less.

【Related Data types and Interfaces】

None

6.2.6 CVI_DATAFIFO_CMD_E

【Description】

Define the control type of the data path.

【Define】

```
/** 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;
```

【Member】

Member Name	Description
DATAFIFO_CMD_GET_PHY_ADDR	Get the physical address of the data path.
DATAFIFO_CMD_READ_DONE	Notify that the read is complete.
DATAFIFO_CMD_WRITE_DONE	The notification write is complete.
DATAFIFO_CMD_SET_DATA_RELEASE_CALLBACK	Set the data release callback function.
DATAFIFO_CMD_GET_AVAIL_WRITE_LEN	Get the length of data that can be written to.
DATAFIFO_CMD_GET_AVAIL_READ_LEN	Get the length of data that can be read.

【Note】

None

【Related Data types and Interfaces】

None

6.3 Error Codes

Errore code	Macro definition	Description
0x00000000	CVI_SUCCESS	Success
0xFFFFFFFF	CVI_FAILURE	Failure