



HiSysLink API 开发参考

文档版本 00B08
发布日期 2019-04-30

版权所有 © 上海海思技术有限公司 2019。保留一切权利。

非经本公司书面许可，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。

商标声明



HISILICON、海思和其他海思商标均为海思技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受海思公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，海思公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

上海海思技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址： <http://www.hisilicon.com/cn/>

客户服务邮箱： support@hisilicon.com



前 言

概述

该文档主要指导用户双核应用开发。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。

产品版本

与本文档相对应的产品版本如下。

| 产品名称 | 产品版本 |
|---------|--------|
| Hi3559A | V100ES |
| Hi3559A | V100 |
| Hi3559C | V100 |
| Hi3556A | V100 |
| Hi3519A | V100 |
| Hi3516C | V500 |
| Hi3516D | V300 |
| Hi3516A | V300 |
| Hi3559 | V200 |
| Hi3556 | V200 |

读者对象

本文档（本指南）主要适用于以下工程师：

- 技术支持工程师
- 软件开发工程师



修订记录

修订记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

文档版本 00B08 (2019-04-30)

第 8 次临时版本发布。

4.1 小节, HI_IPCMSG_DestroyMessage 和 HI_IPCMSG_SendSync 【注意】涉及修改

文档版本 00B07 (2018-11-13)

第 7 次临时版本发布。

4.2 小节, HI_DATAFIFO_Close、HI_DATAFIFO_CMD 涉及修改

文档版本 00B06 (2018-06-15)

第 6 次临时版本发布。

4.1 小节, 新增 HI_IPCMSG_SendOnly

4.2 小节, HI_DATAFIFO_Write 的【注意】涉及修改

文档版本 00B05 (2018-05-15)

第 5 次临时版本发布。

5.2 小节, HI_DATAFIFO_PARAMS_S 【注意】涉及修改

5.3 小节, 表 5-1 涉及修改

文档版本 00B04 (2018-01-10)

第 4 次临时版本发布。

4.1 小节, HI_IPCMSG_DestroyMessage、HI_IPCMSG_SendAsync 和 HI_IPCMSG_SendSync 涉及修改

4.2 小节, HI_DATAFIFO_CMD 涉及修改

5.1 小节, HI_IPCMSG_CONNECT_S 涉及修改, 删除 CPU_ID_E

5.2 小节涉及修改

文档版本 00B03 (2017-09-20)

第 3 次临时版本发布。

5.3 小节, 表 5-1 涉及更新。

文档版本 00B02 (2017-05-27)

第 2 次临时版本发布。

5.1 小节, HI_IPCMSG_CONNECT_S 【定义】和【成员】涉及修改, 新增 CPU_ID_E



文档版本 00B01 (2017-04-28)

第 1 次临时版本发布。



目 录

| | |
|-------------------|----|
| 1 概述..... | 1 |
| 1.1 IPCMSG..... | 1 |
| 1.2 DATAFIFO..... | 2 |
| 2 重要概念..... | 3 |
| 3 功能描述..... | 4 |
| 3.1 IPCMSG..... | 4 |
| 3.2 DATAFIFO..... | 4 |
| 4 API 参考 | 5 |
| 4.1 IPCMSG..... | 5 |
| 4.2 DATAFIFO..... | 17 |
| 5 数据类型..... | 25 |
| 5.1 IPCMSG..... | 25 |
| 5.2 DATAFIFO..... | 30 |
| 5.3 错误码..... | 34 |



插图目录

| | |
|----------------------|---|
| 图 1-1 双核通讯示意图..... | 1 |
| 图 1-2 双核数据传输示意图..... | 2 |



表格目录

| | |
|------------------------------|----|
| 表 5-1 IPCMSG API 错误码 | 34 |
| 表 5-2 DATAFIFO API 错误码 | 34 |



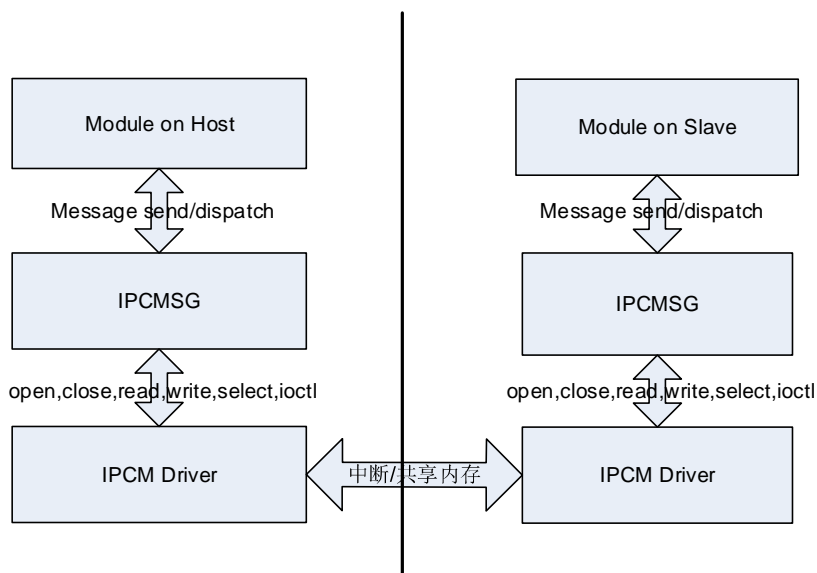
1 概述

HiSysLink 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。

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）。在性能方面，高优先级消息端到端延时在微秒级别，普通优先级消息为毫秒级别。

图1-1 双核通讯示意图

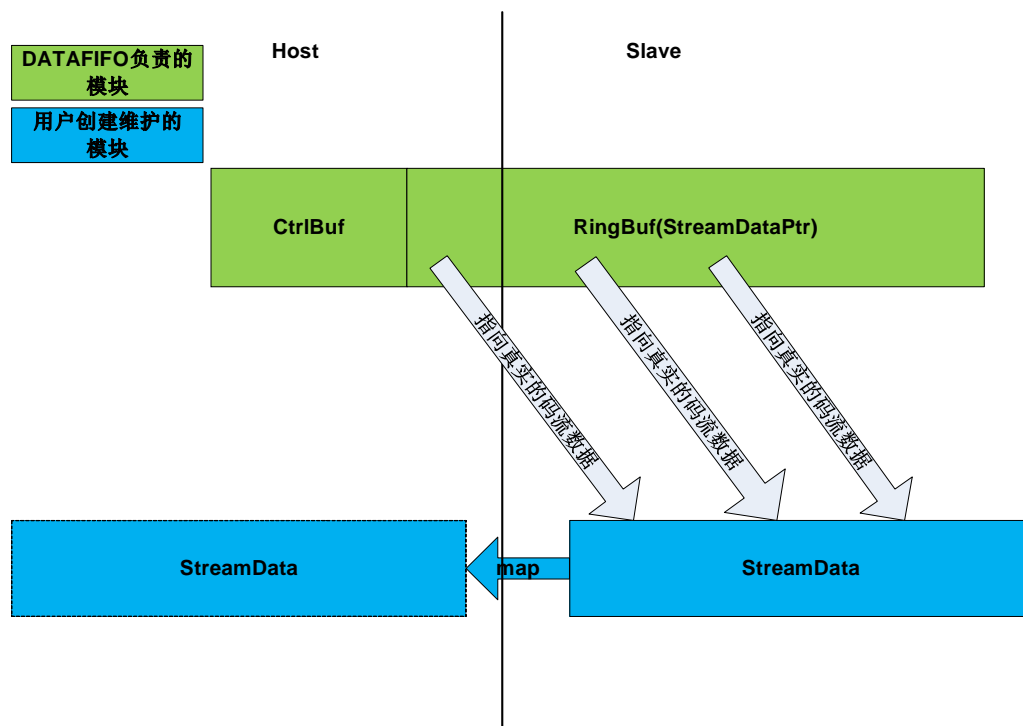




1.2 DATAFIFO

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 1-2 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1-2 双核数据传输示意图





2 重要概念

- **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。
- **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。
- **物理地址**
DDR 上的绝对地址，在双核双系统两端都可见。
- **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，而在 Huawei LiteOS 上虚拟地址和物理地址一样。
- **MMZ**
在双系统环境中，MMZ 内存是分配在两个系统占用内存之外的，所以在 MMZ 上读写数据，都不会影响两个系统运行。
- **映射**
此文档中的映射说的都是物理地址到虚拟地址的映射。
- **Ring Buffer 和码流数据**
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 功能描述

3.1 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。

3.2 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。



4 API 参考

4.1 IPCMSG

该功能模块提供以下 API:

- [HI_IPCMSG_CreateMessage](#): 创建消息。
- [HI_IPCMSG_CreateRespMessage](#): 创建回复消息。
- [HI_IPCMSG_DestroyMessage](#): 销毁消息。
- [HI_IPCMSG_AddService](#): 添加服务。
- [HI_IPCMSG_DelService](#): 删除服务。
- [HI_IPCMSG_TryConnect](#): 非阻塞连接。
- [HI_IPCMSG_Connect](#): 阻塞方式建立连接。
- [HI_IPCMSG_Disconnect](#): 断开连接。
- [HI_IPCMSG_IsConnected](#): 获取是否连接状态。
- [HI_IPCMSG_SendAsync](#): 发送异步消息。
- [HI_IPCMSG_SendSync](#): 发送同步消息。
- [HI_IPCMSG_Run](#): 消息处理函数。
- [HI_IPCMSG_SendOnly](#): 仅发送消息的函数。

HI_IPCMSG_CreateMessage

【描述】

创建消息。

【语法】

```
HI_IPCMSG_MESSAGE_S* HI_IPCMSG_CreateMessage(HI_U32 u32Module, HI_U32  
u32CMD, HI_VOID *pBody, HI_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 |
|----------------------|----------|
| HI_IPCMSG_MESSAGE_S* | 消息结构体指针。 |
| HI_NULL | 消息创建失败。 |

【芯片差异】

无。

【需求】

- 头文件：hi_comm_ipcmsg.h、hi_ipcmsg.h
- 库文件：libipcmsg_big-little.a、libipcmsg_single.a

【注意】

无

【举例】

无

【相关主题】

[HI_IPCMSG_DestroyMessage](#)

HI_IPCMSG_CreateRespMessage

【描述】

创建回复消息。

【语法】

```
HI_IPCMSG_MESSAGE_S* HI_IPCMSG_CreateRespMessage(HI_IPCMSG_MESSAGE_S  
*pstRequest, HI_S32 s32RetVal, HI_VOID *pBody, HI_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 |
|----------------------|----------|
| HI_IPCMSG_MESSAGE_S* | 消息结构体指针。 |
| HI_NULL | 消息创建失败。 |

【芯片差异】

无。

【需求】

- 头文件：hi_comm_ipcmsg.h、hi_ipcmsg.h
- 库文件：libipcmsg_big-little.a、libipcmsg_single.a

【注意】

无

【举例】

无

【相关主题】

[HI_IPCMSG_DestroyMessage](#)

HI_IPCMSG_DestroyMessage

【描述】

销毁消息。

【语法】

```
HI_VOID HI_IPCMSG_DestroyMessage (HI_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 |
|---------|----|
| HI_VOID | 无 |

【芯片差异】

无。

【需求】

- 头文件：hi_comm_ipcmmsg.h、hi_ipcmmsg.h
- 库文件：libipcmmsg_big-little.a、libipcmmsg_single.a

【注意】

不支持同一个消息重复销毁，否则会导致系统异常。

【举例】

无

【相关主题】

- [HI_IPCMMSG_CreateMessage](#)
- [HI_IPCMMSG_CreateRespMessage](#)

HI_IPCMMSG_AddService

【描述】

添加服务。

【语法】

```
HI_S32 HI_IPCMMSG_AddService(const HI_CHAR* pszServiceName, const  
HI\_IPCMMSG\_CONNECT\_S\* pstConnectAttr);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|----------------|--|-------|
| pszServiceName | 服务的名称指针。服务名称最大长度： HI_IPCMMSG_MAX_SERVICENAME_LEN。 | 输入 |
| pstConnectAttr | 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，其值为 错误码 。 |

【芯片差异】

无。

【需求】

- 头文件：hi_comm_ipcmmsg.h、hi_ipcmmsg.h
- 库文件：libipcmmsg_big-little.a、libipcmmsg_single.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 servic 只能对应一个 client。

【举例】

无。

【相关主题】

[HI_IPCMMSG_DelService](#)

HI_IPCMMSG_DelService

【描述】

删除服务。

【语法】

```
HI_S32 HI_IPCMMSG_DelService(const HI_CHAR *pszServiceName);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|----------------|--|-------|
| pszServiceName | 服务的名称指针。服务名称最大长度： HI_IPCMMSG_MAX_SERVICENAME_LEN。 | 输入 |

【返回值】



| 返回值 | 描述 |
|-----|------------------------------|
| 0 | 成功。 |
| 非 0 | 失败，其值为 错误码 。 |

【芯片差异】

无。

【需求】

- 头文件：hi_comm_ipcmmsg.h、hi_ipcmmsg.h
- 库文件：libipcmmsg_big-little.a、libipcmmsg_single.a

【注意】

无

【举例】

无

【相关主题】

[HI_IPCMMSG_AddService](#)

HI_IPCMMSG_TryConnect

【描述】

非阻塞方式建立连接。

【语法】

```
HI_S32 HI_IPCMMSG_TryConnect(HI_S32 *ps32Id, const HI_CHAR *pszServiceName,  
HI\_IPCMMSG\_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。 |



| 返回值 | 描述 |
|-----|------------------------------|
| 非 0 | 失败，其值为 错误码 。 |

【芯片差异】

无。

【需求】

- 头文件：hi_comm_ipcmmsg.h、hi_ipcmmsg.h
- 库文件：libipcmmsg_big-little.a、libipcmmsg_single.a

【注意】

无

【举例】

无

【相关主题】

[HI_IPCMMSG_Disconnect](#)

HI_IPCMMSG_Connect

【描述】

阻塞方式建立连接。

【语法】

```
HI_S32 HI_IPCMMSG_Connect(HI_S32 *ps32Id, const HI_CHAR *pszServiceName,  
HI\_IPCMMSG\_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 | 消息处理函数。 | 输入 |

【返回值】

| 返回值 | 描述 |
|-----|------------------------------|
| 0 | 成功。 |
| 非 0 | 失败，其值为 错误码 。 |



【芯片差异】

无。

【需求】

- 头文件：hi_comm_ipcmmsg.h、hi_ipcmmsg.h
- 库文件：libipcmmsg_big-little.a、libipcmmsg_single.a

【注意】

无

【举例】

无

【相关主题】

[HI_IPCMMSG_Disconnect](#)

HI_IPCMMSG_Disconnect

【描述】

断开连接。

【语法】

```
HI_S32 HI_IPCMMSG_Disconnect (HI_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，其值为 错误码 。 |

【芯片差异】

无。

【需求】

- 头文件：hi_comm_ipcmmsg.h、hi_ipcmmsg.h



- 库文件: libipcmmsg_big-little.a、libipcmmsg_single.a

【注意】

无

【举例】

无

【相关主题】

- [HI_IPCMMSG_TryConnect](#)
- [HI_IPCMMSG_Connect](#)

HI_IPCMMSG_IsConnected

【描述】

消息通信是否连接状态。

【语法】

```
HI_BOOL HI_IPCMMSG_IsConnected(HI_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 |
|----------|--------|
| HI_TRUE | 连接状态。 |
| HI_FALSE | 非连接状态。 |

【芯片差异】

无。

【需求】

- 头文件: hi_comm_ipcmmsg.h、hi_ipcmmsg.h
- 库文件: libipcmmsg_big-little.a、libipcmmsg_single.a

【注意】

无

【举例】



无

【相关主题】

- [HI_IPCMSG_Connect](#)
- [HI_IPCMSG_TryConnect](#)

HI_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】

```
HI_S32 HI_IPCMSG_SendAsync(HI_S32 s32Id, HI_IPCMSG_MESSAGE_S *pstMsg,  
HI_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，其值为 错误码 。 |

【芯片差异】

无。

【需求】

- 头文件：hi_comm_ipcmsg.h、hi_ipcmsg.h
- 库文件：libipcmsg_big-little.a、libipcmsg_single.a

【注意】

无



【举例】

无

【相关主题】

[HI_IPCMSG_SendSync](#)

HI_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】

```
HI_S32 HI_IPCMSG_SendSync(HI_S32 s32Id, HI_IPCMSG_MESSAGE_S *pstMsg,  
HI_IPCMSG_MESSAGE_S **ppstMsg, HI_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，其值为 错误码 。 |

【芯片差异】

无

【需求】

- 头文件：hi_comm_ipcmsg.h、hi_ipcmsg.h
- 库文件：libipcmsg_big-little.a、libipcmsg_single.a

【注意】

本接口超时的情况下，内部会调用 [HI_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】

[HI_IPCMSG_SendAsync](#)

HI_IPCMSG_Run

【描述】

消息处理函数。

【语法】

```
HI_VOID HI_IPCMSG_Run(HI_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 |
|---------|----|
| HI_VOID | 无。 |

【芯片差异】

无

【需求】

- 头文件：hi_comm_ipcmmsg.h、hi_ipcmmsg.h
- 库文件：libipcmmsg_big-little.a、libipcmmsg_single.a

【注意】

无

【举例】

无

【相关主题】

无



HI_IPCMSG_SendOnly

【描述】

仅发送消息给对端，不接收对端的返回值。

【语法】

```
HI_S32 HI_IPCMSG_SendOnly(HI_S32 s32Id, HI_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，其值为 错误码 。 |

【芯片差异】

无

【需求】

- 头文件：hi_comm_ipcmsg.h、hi_ipcmsg.h
- 库文件：libipcmsg_big-little.a、libipcmsg_single.a

【注意】

无

【举例】

无

【相关主题】

无

4.2 DATAFIFO

该功能模块提供以下 API：

- [HI_DATAFIFO_Open](#)：打开数据通路。



- [HI_DATAFIFO_OpenByAddr](#): 通过物理地址打开通路。
- [HI_DATAFIFO_Close](#): 关闭通路。
- [HI_DATAFIFO_Read](#): 读取数据。
- [HI_DATAFIFO_Write](#): 写入数据。
- [HI_DATAFIFO_CMD](#): 其他控制命令。

HI_DATAFIFO_Open

【描述】

打开数据通路。

【语法】

```
HI_S32 HI_DATAFIFO_Open(HI\_DATAFIFO\_HANDLE *Handle, HI\_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，其值为 错误码 。 |

【芯片差异】

无

【需求】

- 头文件: [hi_datafifo.h](#)
- 库文件: [libdatafifo_big-little.a](#)、[libdatafifo_single.a](#)

【注意】

无

【举例】

无

【相关主题】



[HI_DATAFIFO_Close](#)

HI_DATAFIFO_OpenByAddr

【描述】

通过物理地址打开数据通路。

【语法】

```
HI_S32 HI_DATAFIFO_OpenByAddr(HI_DATAFIFO_HANDLE *Handle,  
HI_DATAFIFO_PARAMS_S *pstParams, HI_U32 u32PhyAddr);
```

【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，其值为 错误码 。 |

【芯片差异】

无。

【需求】

- 头文件：hi_datafifo.h
- 库文件：libdatafifo_big-little.a、libdatafifo_single.a

【注意】

无

【举例】

无

【相关主题】

[HI_DATAFIFO_Close](#)



HI_DATAFIFO_Close

【描述】

关闭数据通路。

【语法】

```
HI_S32 HI_DATAFIFO_Close(HI\_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，其值为 错误码 。 |

【芯片差异】

无

【需求】

- 头文件：hi_datafifo.h
- 库文件：libdatafifo_big-little.a、libdatafifo_single.a

【注意】

关闭 DataFifo 的时候为了保证读写两端数据正常的释放，用户需要保证读端要读完 DataFifo 中存在的数据，写端写完数据后需要额外调用一次 [HI_DATAFIFO_Write](#)(Handle, NULL) 触发写端的数据释放和读指针更新。

【举例】

无

【相关主题】

- [HI_DATAFIFO_Open](#)
- [HI_DATAFIFO_OpenByAddr](#)

HI_DATAFIFO_Read

【描述】

读取数据。



【语法】

```
HI_S32 HI_DATAFIFO_Read(HI\_DATAFIFO\_HANDLE Handle, HI_VOID **ppData);
```

【参数】

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

【返回值】

| 返回值 | 描述 |
|-----|------------------------------|
| 0 | 成功。 |
| 非 0 | 失败，其值为 错误码 。 |

【芯片差异】

无。

【需求】

- 头文件：hi_datafifo.h
- 库文件：libdatafifo_big-little.a、libdatafifo_single.a

【注意】

无

【举例】

无

【相关主题】

- [HI_DATAFIFO_Write](#)
- [HI_DATAFIFO_CMD](#)

HI_DATAFIFO_Write

【描述】

写入数据。

【语法】

```
HI_S32 HI_DATAFIFO_Write(HI\_DATAFIFO\_HANDLE Handle, HI_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。 | 输入 |

【返回值】

| 返回值 | 描述 |
|-----|------------------------------|
| 0 | 成功。 |
| 非 0 | 失败，其值为 错误码 。 |

【芯片差异】

无

【需求】

- 头文件：hi_datafifo.h
- 库文件：libdatafifo_big-little.a、libdatafifo_single.a

【注意】

当 pData 为 NULL 时，触发写端的数据释放回调函数，同时更新写端的读尾指针。

【举例】

无

【相关主题】

- [HI_DATAFIFO_Read](#)
- [HI_DATAFIFO_CMD](#)

HI_DATAFIFO_CMD

【描述】

其他操作。

【语法】

```
HI_S32 HI_DATAFIFO_CMD(HI\_DATAFIFO\_HANDLE Handle, HI\_DATAFIFO\_CMD\_E enCMD,  
HI_VOID *pArg);
```

【参数】

| 参数名称 | 描述 | 输入/输出 |
|--------|---------|-------|
| Handle | 数据通路句柄。 | 输入 |



| 参数名称 | 描述 | 输入/输出 |
|-------|------------|-------|
| enCMD | 操作命令。 | 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，其值为 错误码 。 |

【芯片差异】

无

【需求】

- 头文件：hi_datafifo.h
- 库文件：libdatafifo_big-little.a、libdatafifo_single.a

【注意】

控制命令和对应参数：

| 命令 | 参数以及说明 |
|--|--|
| DATAFIFO_CMD_GET_PHY_ADDR | 返回 DATAFIFO 的物理地址，HI_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 HI_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 HI_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，HI_U32 类型。 |
| DATAFIFO_CMD_GET_AVAIL_READ_LEN | 返回可读取的数据个数，HI_U32 类型。 |



【举例】

无

【相关主题】

无



5 数据类型

5.1 IPCMSG

相关数据类型定义如下：

- [HI_IPCMSG_MAX_CONTENT_LEN](#)：定义消息体最长度。
- [HI_IPCMSG_PRIVDATA_NUM](#)：定义消息体中私有数据最大个数。
- [HI_IPCMSG_INVALID_MSGID](#)：定义无效消息 ID。
- [HI_IPCMSG_MAX_SERVICENAME_LEN](#)：定义服务名称的最大长度。
- [HI_IPCMSG_PORT_HIMPP](#)：定义 HIMPP 使用的服务端口号。
- [HI_IPCMSG_PORT_FAKEFS](#)：定义 FAKEFS 使用的服务端口号。
- [HI_IPCMSG_CONNECT_S](#)：定义连接对端服务器的结构体。
- [HI_IPCMSG_MESSAGE_S](#)：定义视频输入设备的接口模式。
- [HI_IPCMSG_HANDLE_FN_PTR](#)：定义视频设备的输入模式。
- [HI_IPCMSG_RESPHANDLE_FN_PTR](#)：定义回复消息的处理函数。

HI_IPCMSG_MAX_CONTENT_LEN

【说明】

定义消息体最长度。

【定义】

```
#define HI_IPCMSG_MAX_CONTENT_LEN (1024)
```

【注意事项】

无

【相关数据类型及接口】

[HI_DATAFIFO_Close](#)

HI_IPCMSG_PRIVDATA_NUM

【说明】



定义消息体中私有数据最大个数。

【定义】

```
#define HI_IPCMSG_PRIVDATA_NUM (8)
```

【注意事项】

无

【相关数据类型及接口】

[HI_DATAFIFO_Close](#)

HI_IPCMSG_INVALID_MSGID

【说明】

定义无效消息 ID。

【定义】

```
#define HI_IPCMSG_INVALID_MSGID (0xFFFFFFFFFFFFFFFF)
```

【注意事项】

无

【相关数据类型及接口】

无

HI_IPCMSG_MAX_SERVICENAME_LEN

【说明】

定义服务名称的最大长度。

【定义】

```
#define HI_IPCMSG_MAX_SERVICENAME_LEN (16)
```

【注意事项】

无

【相关数据类型及接口】

无

HI_IPCMSG_PORT_HIMPP

【说明】

定义 HIMPP 使用的服务端口号。

【定义】

```
#define HI_IPCMSG_PORT_HIMPP (1)
```



【注意事项】

无

【相关数据类型及接口】

无

HI_IPCMSG_PORT_FAKEFS

【说明】

定义 FAKEFS 使用的服务端口号。

【定义】

```
#define HI_IPCMSG_PORT_FAKEFS (2)
```

【注意事项】

无

【相关数据类型及接口】

无

HI_IPCMSG_CONNECT_S

【说明】

定义连接对端服务器的结构体。

【定义】

```
typedef struct hiIPCMSG_CONNECT_S  
{  
    HI_U32 u32RemoteId;  
    HI_U32 u32Port;  
    HI_U32 u32Priority;  
} HI_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。 取值范围：[0, 512]。 |



| 成员名称 | 描述 |
|-------------|---|
| 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

HI_IPCMSG_MESSAGE_S

【说明】

定义消息结构体。

【定义】

```
typedef struct hiIPCMSG_MESSAGE_S
{
    HI_BOOL bIsResp;    /**<Identify the response messgae*/
    HI_U64 u64Id;        /**<Message ID*/
    HI_U32 u32Module;    /**<Module ID, user-defined*/
    HI_U32 u32CMD;        /**<CMD ID, user-defined*/
    HI_S32 s32RetVal;    /**<Retrun Value in response message*/
    HI_S32 as32PrivData[HI_IPCMSG_PRIVDATA_NUM]; /**<Private data, can be
modify directly after ::HI_IPCMSG_CreateMessage
or ::HI_IPCMSG_CreateRespMessage*/
    HI_VOID* pBody;        /**<Message body*/
    HI_U32 u32BodyLen;    /**<Length of pBody*/
} HI_IPCMSG_MESSAGE_S
```

【成员】



| 成员名称 | 描述 |
|--------------|--|
| bIsResp | 标示该消息是否回复消息： <ul style="list-style-type: none">• HI_TRUE: 回复;• HI_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。。 |
| u32BodyLen | 消息体长度，单位字节。 |

【注意事项】

无

【相关数据类型及接口】

无

HI_IPCMSG_HANDLE_FN_PTR

【说明】

定义消息回复处理函数。

【定义】

```
typedef void (*HI_IPCMSG_HANDLE_FN_PTR) (HI_S32 s32Id, HI_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】



HI_IPCMSG_DestroyMessage

HI_IPCMSG_RESPHANDLE_FN_PTR

【说明】

定义回复消息的处理函数。

【定义】

```
typedef void (*HI_IPCMSG_RESPHANDLE_FN_PTR) (HI_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

5.2 DATAFIFO

相关数据类型定义如下：

- [HI_DATAFIFO_HANDLE](#)：定义 DATAFIFO 的句柄。
- [MAX_NAME_LEN](#)：定义数据通路名称最大长度。
- [HI_DATAFIFO_INVALID_HANDLE](#)：定义数据通路无效句柄。
- [HI_DATAFIFO_RELEASESTREAM_FN_PTR](#)：定义数据通路码流释放函数。
- [HI_DATAFIFO_OPEN_MODE_E](#)：定义数据通路打开模式。
- [HI_DATAFIFO_PARAMS_S](#)：定义数据通路配置参数。
- [HI_DATAFIFO_CMD_E](#)：定义数据通路的控制类型。

HI_DATAFIFO_HANDLE

【说明】

定义 DATAFIFO 的句柄。

【定义】

```
typedef HI_U32 HI_DATAFIFO_HANDLE;
```

【注意事项】



无

【相关数据类型及接口】

无

MAX_NAME_LEN

【说明】

定义数据通路名称最大长度。

【定义】

```
#define MAX_NAME_LEN (16)
```

【注意事项】

无

【相关数据类型及接口】

无

HI_DATAFIFO_INVALID_HANDLE

【说明】

定义数据通路无效句柄。

【定义】

```
#define HI_DATAFIFO_INVALID_HANDLE (-1)
```

【注意事项】

无

【相关数据类型及接口】

无

HI_DATAFIFO_RELEASESTREAM_FN_PTR

【说明】

定义数据通路码流释放函数。

【定义】

```
typedef void (*HI_DATAFIFO_RELEASESTREAM_FN_PTR)(void *pStream)
```

【注意事项】

无

【相关数据类型及接口】



无

HI_DATAFIFO_OPEN_MODE_E

【说明】

定义数据通路打开模式。

【定义】

```
typedef enum hiDATAFIFO_OPEN_MODE_E
{
    DATAFIFO_READER,
    DATAFIFO_WRITER,
} HI_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

HI_DATAFIFO_PARAMS_S

【说明】

定义数据通路配置参数。

【定义】

```
typedef struct hiDATAFIFO_PARAMS_S
{
    HI_U32 u32EntriesNum; /**<The number of items in the ring buffer*/
    HI_U32 u32CacheLineSize; /**<Item size*/
    HI_BOOL bDataReleaseByWriter; /**<Whether the data buffer release by writer*/
    HI_DATAFIFO_OPEN_MODE_E enOpenMode; /**<READER or WRITER*/
} HI_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】

u32EntriesNum 和 u32CacheLineSize 并没有做取值范围做限制，只要 MMZ 内存足够大，DATAFIFO 就可以创建成功。因此，需要用户保证这 2 个参数在合理的范围之内。

【相关数据类型及接口】

无

HI_DATAFIFO_CMD_E

【说明】

定义数据通路的控制类型。

【定义】

```
typedef enum hiDATAFIFO_CMD_E
{
    DATAFIFO_CMD_GET_PHY_ADDR, /**<Get the physic address of ring buffer*/
    DATAFIFO_CMD_READ_DONE, /**<When the read buffer read over, the reader
should call this function to notify the writer*/
    DATAFIFO_CMD_WRITE_DONE, /**<When the writer buffer is write done, the
writer should call this function*/
    DATAFIFO_CMD_SET_DATA_RELEASE_CALLBACK, /**<When bDataReleaseByWriter
is HI_TRUE, the writer should call this to register release callback*/
    DATAFIFO_CMD_GET_AVAIL_WRITE_LEN, /**<Get available write length*/
    DATAFIFO_CMD_GET_AVAIL_READ_LEN /**<Get available read length*/
} HI_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。 |



| 成员名称 | 描述 |
|--|-------------|
| 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

5.3 错误码

IPCMSG API 错误码如表 5-1 所示。

表5-1 IPCMSG API 错误码

| 错误代码 | 宏定义 | 描述 |
|------------|---------------------|--------|
| 0x1901 | HI_IPCMSG_EINVAL | 配置参数无效 |
| 0x1902 | HI_IPCMSG_ETIMEOUT | 超时错误 |
| 0x1903 | HI_IPCMSG_ENOOP | 驱动打开失败 |
| 0x1904 | HI_IPCMSG_EINTER | 内部错误 |
| 0x1905 | HI_IPCMSG_ENULL_PTR | 空指针错误 |
| 0x00000000 | HI_SUCCESS | 成功 |
| 0xFFFFFFFF | HI_FAILURE | 失败 |

DATAFIFO API 错误码如表 5-2 所示。

表5-2 DATAFIFO API 错误码

| 错误代码 | 宏定义 | 描述 |
|------------|------------|----|
| 0x00000000 | HI_SUCCESS | 成功 |
| 0xFFFFFFFF | HI_FAILURE | 失败 |