



CV180X & CV181X IPCM 客制化通信 使用手册

Version:

Release date: 2023-09-18

©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	2
2	IPCM_PORT_COMMON	3
2.1	功能概述	3
2.2	设计概述	3
2.3	API 参考	4
2.3.1	ipcm_port_common_init	4
2.3.2	ipcm_port_common_uninit	5
2.3.3	ipcm_get_buff_to_pos	5
2.3.4	ipcm_release_buff_by_pos	6
2.3.5	ipcm_inv_data	7
2.3.6	ipcm_flush_data	7
2.3.7	ipcm_data_lock	8
2.3.8	ipcm_data_unlock	9
2.3.9	ipcm_get_buff	10
2.3.10	ipcm_release_buff	10
2.3.11	ipcm_data_packed	11
2.3.12	ipcm_get_msg_data	12
2.3.13	ipcm_get_data_by_pos	12
2.3.14	ipcm_common_send_msg	13
2.3.15	ipcm_get_user_addr	14
2.3.16	ipcm_pool_reset	15
2.3.17	get_param_bin_addr	15
2.3.18	get_param_bak_bin_addr	16
2.3.19	get_pq_bin_addr	17
2.4	数据类型	17
2.4.1	基础类型	17
2.4.2	BlockConfig	18
2.4.3	MsgData	18
2.4.4	PortType	19
2.4.5	MsgType	20
2.4.6	MsgParam	20
2.4.7	MsgPtr	21
2.5	错误码	22
3	IPCM_ANON	23
3.1	功能概述	23
3.2	设计概述	23
3.3	API 参考	24
3.3.1	ipcm_anon_init	24
3.3.2	ipcm_anon_uninit	24

3.3.3	ipcm_anon_register_handle	25
3.3.4	ipcm_anon_send_msg	26
3.3.5	ipcm_anon_send_param	26
3.3.6	ipcm_anon_get_user_addr	27
3.4	数据类型	28
3.4.1	ipcm_anon_msg_t	28
3.4.2	ANON_MSGPROC_FN	29
3.5	错误码	29

修订记录

Revision	Date	Description
v1.0	2023/09/18	初稿

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/>

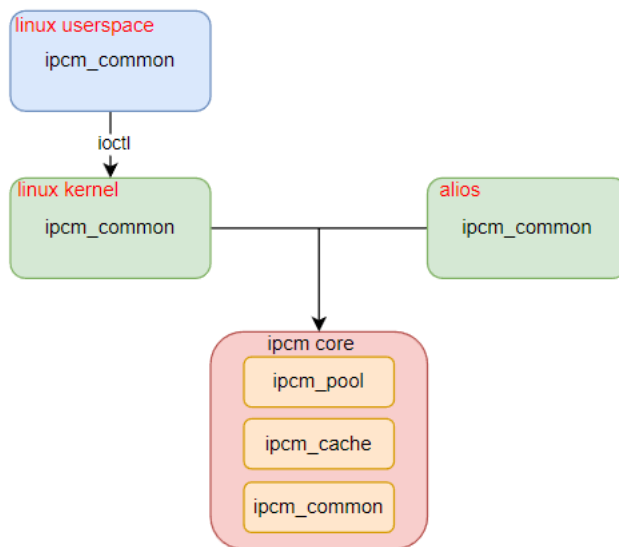
技术论坛 <https://developer.sophgo.com/forum/index.html>

2 IPCM_PORT_COMMON

2.1 功能概述

为 ipcm 提供访问 share memory、操作 cache、获取 param/param_bak/pqbin addr 的接口。

2.2 设计概述



- ipcm_common 接口分别在 linux userspace、linux kernel、alios 都有实现
- linux kernel、alios 直接调用 ipcm core 提供的接口
- linux userspace 则会通过 ioctl 下发到 kernel 进行操作

2.3 API 参考

2.3.1 ipcm_port_common_init

【描述】

ipcm port common 初始化。

【语法】

- alios:

```
s32 ipcm_port_common_init(BlockConfig *config, u32 num);
```

- linux:

```
s32 ipcm_port_common_init(void);
```

【参数】

alios:

参数名称	描述	输入/输出
config	share memory pool 配置指针	输入
num	share memory pool 配置个数	

linux:

无

【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: ipcm_common.h, ipcm_port_common.h

【注意】

无

【举例】

无

【相关主题】

无

2.3.2 ipcm_port_common_uninit

【描述】

ipcm port common 去初始化。

【语法】

```
s32 ipcm_port_common_uninit(void);
```

【参数】

无

【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: ipcm_common.h, ipcm_port_common.h

【注意】

无

【举例】

无

【相关主题】

无

2.3.3 ipcm_get_buff_to_pos

【描述】

从 share memory 中申请一块空闲 pool, 返回 pool 相对 share memory 起始 pool 的偏移。

【语法】

```
u32 ipcm_get_buff_to_pos(u32 size);
```

【参数】

参数名称	描述	输入/输出
size	申请 pool 的长度	输入

【返回值】

返回值	描述
U32_MAX	失败
其他	成功, 返回相对 share memory 起始 pool 的偏移

【需求】

- 头文件: ipcm_common.h, ipcm_port_common.h

【注意】

无

【举例】

无

【相关主题】

无

2.3.4 ipcm_release_buff_by_pos

【描述】

释放指定偏移地址的 share memory pool。

【语法】

```
s32 ipcm_release_buff_by_pos(u32 pos);
```

【参数】

参数名称	描述	输入/输出
pos	需要释放的 pool 的偏移地址	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: ipcm_common.h, ipcm_port_common.h

【注意】

无

【举例】

无

【相关主题】

无

2.3.5 ipcm_inv_data

【描述】

使指定地址的 cache 无效。

【语法】

```
s32 ipcm_inv_data(void *data, u32 size);
```

【参数】

参数名称	描述	输入/输出
data	数据地址，需要 64byte 对齐	输入
size	数据长度，需要 64byte 对齐	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: ipcm_common.h, ipcm_port_common.h

【注意】

无

【举例】

无

【相关主题】

无

2.3.6 ipcm_flush_data

【描述】

将指定地址的内容 flush 到 ddr。

【语法】

```
s32 ipcm_flush_data(void *data, u32 size);
```

【参数】

参数名称	描述	输入/输出
data	数据地址，需要 64byte 对齐	输入
size	数据长度，需要 64byte 对齐	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: ipcm_common.h, ipcm_port_common.h

【注意】

无

【举例】

无

【相关主题】

无

2.3.7 ipcm_data_lock

【描述】

大小核访问互斥锁上锁，同一个 lock id，只有拿到 lock 的核才能访问。

【语法】

```
s32 ipcm_data_lock(u8 lock_id);
```

【参数】

参数名称	描述	输入/输出
lock_id	锁 id 号；当前支持 5 组 data lock，lock id 的范围是 [0,4]	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: ipcm_common.h, ipcm_port_common.h

【注意】

无

【举例】

无

【相关主题】

无

2.3.8 ipcm_data_unlock

【描述】

大小核访问互斥锁解锁。

【语法】

```
s32 ipcm_data_unlock(u8 lock_id);
```

【参数】

参数名称	描述	输入/输出
lock_id	锁 id 号; 当前支持 5 组 data lock, lock id 的范围是 [0,4]	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: ipcm_common.h, ipcm_port_common.h

【注意】

无

【举例】

无

【相关主题】

无

2.3.9 ipcm_get_buff

【描述】

从 share memory 中申请一块空闲 pool，返回 pool 的地址。

【语法】

```
void *ipcm_get_buff(u32 size);
```

【参数】

参数名称	描述	输入/输出
size	申请 pool 的长度	输入

【返回值】

返回值	描述
NULL	失败
其他	返回 pool 的地址。

【需求】

- 头文件: ipcm_common.h, ipcm_port_common.h

【注意】

无

【举例】

无

【相关主题】

无

2.3.10 ipcm_release_buff

【描述】

释放指定地址的 share memory pool。

【语法】

```
s32 ipcm_release_buff(void *data);
```

【参数】

参数名称	描述	输入/输出
data	pool 的地址	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: ipcm_common.h, ipcm_port_common.h

【注意】

无

【举例】

无

【相关主题】

无

2.3.11 ipcm_data_packed

【描述】

将指定地址的 share memory pool 打包到待发送的 msg 结构体中。

【语法】

```
s32 ipcm_data_packed(void *data, u32 len, MsgData *msg);
```

【参数】

参数名称	描述	输入/输出
data	pool 的地址	输入
len	pool 的长度	输入
msg	待发送的 msg 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: ipcm_common.h, ipcm_port_common.h

【注意】

无

【举例】

无

【相关主题】

无

2.3.12 ipcm_get_msg_data

【描述】

获取接收的 msg 中 share memory pool 的地址。

【语法】

```
void *ipcm_get_msg_data(MsgData *msg);
```

【参数】

参数名称	描述	输入/输出
msg	接收的 msg 指针	输入

【返回值】

返回值	描述
NULL	失败
其他	返回 msg 对应的 share memory pool 地址

【需求】

- 头文件: ipcm_common.h, ipcm_port_common.h

【注意】

无

【举例】

无

【相关主题】

无

2.3.13 ipcm_get_data_by_pos

【描述】

获取指定偏移的 share memory pool 的地址。

【语法】

```
void *ipcm_get_data_by_pos(u32 data_pos);
```

【参数】

参数名称	描述	输入/输出
data_pos	share memory pool 的偏移	输入

【返回值】

返回值	描述
NULL	失败
其他	成功, 返回 share memory pool 的地址

【需求】

- 头文件: ipcm_common.h, ipcm_port_common.h

【注意】

无

【举例】

无

【相关主题】

无

2.3.14 ipcm_common_send_msg

【描述】

通过 mailbox 发送消息。

【语法】

```
s32 ipcm_common_send_msg(MsgData *msg);
```

【参数】

参数名称	描述	输入/输出
msg	待发送的 msg 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: ipcm_common.h, ipcm_port_common.h

【注意】

无

【举例】

无

【相关主题】

无

2.3.15 ipcm_get_user_addr

【描述】

返回对应物理地址的虚拟地址，物理地址的范围必须是 alios 管理的地址。

在 **linux** 侧，将从 alios 拿到的物理地址转成虚拟地址；

在 **alios** 侧，因为 alios 没有开 mmc，所以直接返回 alios 的物理地址。

【语法】

```
void *ipcm_get_user_addr(u32 paddr);
```

【参数】

参数名称	描述	输入/输出
paddr	待转换的物理地址	输入

【返回值】

返回值	描述
NULL	失败，物理地址不在 alios 管理的范围
其他	成功，返回对应物理地址的虚拟地址

【需求】

- 头文件: ipcm_common.h, ipcm_port_common.h

【注意】

无

【举例】

无

【相关主题】

无

2.3.16 ipcm_pool_reset

【描述】

将 share memory pool 全部复位为 free 状态，一般在系统第一次启动时 ipcm 驱动会调用。

【语法】

```
s32 ipcm_pool_reset(void);
```

【参数】

无

【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: ipcm_common.h, ipcm_port_common.h

【注意】

无

【举例】

无

【相关主题】

无

2.3.17 get_param_bin_addr

【描述】

获取参数区域的物理地址。

【语法】

```
u32 get_param_bin_addr(void);
```

【参数】

无

【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: ipcm_common.h, ipcm_port_common.h

【注意】

无

【举例】

无

【相关主题】

无

2.3.18 get_param_bak_bin_addr

【描述】

获取参数备份区域的物理地址。

【语法】

```
u32 get_param_bak_bin_addr(void);
```

【参数】

无

【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: ipcm_common.h, ipcm_port_common.h

【注意】

无

【举例】

无

【相关主题】

无

2.3.19 get_pq_bin_addr

【描述】

获取 pq 参数的物理地址。

【语法】

```
u32 get_pq_bin_addr(void);
```

【参数】

无

【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: ipcm_common.h, ipcm_port_common.h

【注意】

无

【举例】

无

【相关主题】

无

2.4 数据类型

2.4.1 基础类型

```
#ifndef U32_MAX
#define U32_MAX ((u32)~0U)
#endif
typedef unsigned char u8;
typedef signed char s8;
typedef unsigned short u16;
typedef signed short s16;
typedef unsigned int u32;
typedef int s32;
```

2.4.2 BlockConfig

【说明】

定义 share memory pool 结构体

【定义】

```
typedef struct _BlockConfig {
    u32 size;
    u32 num;
} BlockConfig;
```

【成员】

成员名称	描述
size	pool 长度, 需要 64byte 对齐
num	对应长度的 pool 个数

【注意事项】

- size 需要 64byte 对齐
- size 最大不能超过 4095
- 所有 pool 加起来总 size 不能超过 (CVIMMAP_SHARE_MEM_SIZE-4)

【相关数据类型及接口】

无

2.4.3 MsgData

【说明】

定义 mailbox 消息结构体

【定义】

```
typedef struct _MsgData {
    u8 port_id;
    u8 msg_id : 7;
    u8 func_type : 1;
    union resv_t resv;
    MsgParam msg_param;
} __attribute__((packed)) __attribute__((aligned(0x8))) MsgData;
```

【成员】

成员名称	描述
port_id	port 号
msg_id	msg 号
func_type	msg 数据类型 0:msg_param 存的是 share memory pool 的 pos 和 size 1: msg_param 存的是 msg 的内容, msg 内容不超过 4bytes
resv	保留
msg_param	msg 内容

【注意事项】

无

【相关数据类型及接口】

无

2.4.4 PortType

【说明】

定义不同功能的 port 号

【定义】

```
typedef enum _PortType {  
    PORT_SYSTEM = 0,  
    PORT_MSG,  
    // PORT_VIRTTTY,  
    // PORT_SHAREFS,  
#ifdef __IPCM_UT__  
    PORT_SYSTEM_UT,  
    PORT_MSG_UT,  
    // PORT_VIRTTTY_UT,  
    // PORT_SHAREFS_UT,  
#endif  
    PORT_ANON_ST,  
    PORT_BUTT = 255,  
} PortType;
```

【成员】

成员名称	描述
PORT_SYSTEM	system port 号, 内部使用
PORT_MSG	msg port 号, ipcmsg 内部使用
PORT_SYSTEM_UT	UT 开启后, system ut port 号, 内部使用
PORT_MSG_UT	UT 开启后, msg ut port 号, 内部使用
PORT_ANON_ST	anon port 起始号, ipcmanon api 使用

【注意事项】

- port id 目前开放的范围是 [PORT_ANON_ST, PORT_BUTT), 其他 port id 为内部使用。

【相关数据类型及接口】

无

2.4.5 MsgType

【说明】

定义 msg param 的类别

【定义】

```
typedef enum _MsgType {  
    MSG_TYPE_SHM = 0, // msg_param is share memory addr  
    MSG_TYPE_RAW_PARAM, // msg_param is the param  
} MsgType;
```

【成员】

成员名称	描述
MSG_TYPE_SHM	msg 长度超过 4byte, param 存的是 share memory pool 的 pos 和 size
MSG_TYPE_RAW_PARAM	msg 内容不超过 4bytes, param 存的是 msg 的内容

【注意事项】

无

【相关数据类型及接口】

无

2.4.6 MsgParam

【说明】

定义 msg 的内容

【定义】

```
typedef union _MsgParam {  
    MsgPtr msg_ptr;  
    unsigned int param;  
} MsgParam;
```

【成员】

成员名称	描述
msg_ptr	当 msg 长度超过 4byte, 存放 share memory pool 的 pos 和 size
param	当 msg 内容不超过 4bytes, 存放 msg 的内容

【注意事项】

无

【相关数据类型及接口】

无

2.4.7 MsgPtr

【说明】

当 msg 长度超过 4byte, 定义 share memory pool 的 pos 和 size

【定义】

```
typedef struct _MsgPtr {  
    u32 data_pos : 20;  
    u32 remaining_rd_len : 12;  
} MsgPtr;
```

【成员】

成员名称	描述
data_pos	share memory pool 的偏移
remaining_rd_len	share memory pool 剩余未读取的数据长度; msg send 时保存 msg 数据的实际长度

【注意事项】

无

【相关数据类型及接口】

无

2.5 错误码

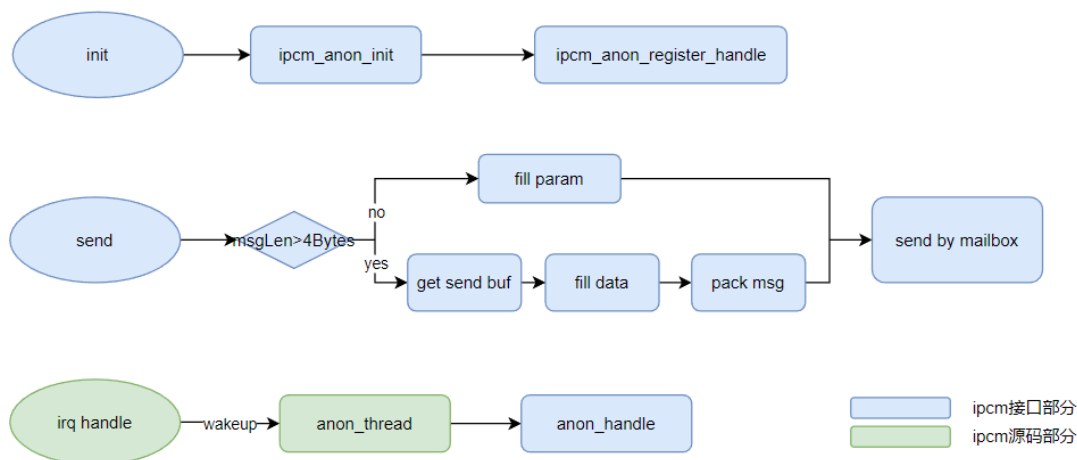
错误代码	宏定义	描述
-200	-EQFULL	msg 队列消息未及时处理，msg 队列已满
-201	-EQEMPTY	msg 队列中不存在 msg，msg 队列为空
-202	-EMLOCK	mailbox 上锁失败
-203	-EMBUSY	mailbox 通道已全部使用
-204	-EQNULL	msg 队列未初始化

3 IPCM_ANON

3.1 功能概述

为 app 提供双核通信的 api 接口。

3.2 设计概述



- ipcm anon 初始化只需要调用 ipcm_anon_init 和调用 ipcm_anon_register_handle 注册 handle 即可
- 如果 msg 长度不超过 4byte，例如只传一下 command，则可以直接填充 msg 结构体的 param 域，然后通过 mailbox 发送到另一个 RISC-V
- 如果 msg 长度超过 4byte，则需要从 share memory 中申请一块 pool，然后填充 pool 内容和调用 ipcm_data_packed 进行 msg 打包，最后再通过 mailbox 发送到另一个 RISC-V

注意： msg size 有长度限制

1. 最大不能超过 4095
2. 最大不能超过 ipcm_port_common_init 时配置的最大 pool size

- 当 RISC-V 收到 anon msg 时, anon_thread 会被唤醒, 并调用 anon 初始化时注册的 handle, 在 handle 中可以通过 port_id 和 msg_id 区分不同的 anon msg

3.3 API 参考

3.3.1 ipcm_anon_init

【描述】

ipcm anon 初始化。

【语法】

```
s32 ipcm_anon_init(void);
```

【参数】

无

【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: ipcm_common.h, ipcm_anonymous.h

【注意】

无

【举例】

无

【相关主题】

无

3.3.2 ipcm_anon_uninit

【描述】

ipcm anon 去初始化。

【语法】

```
s32 ipcm_anon_uninit(void);
```

【参数】

无

【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: ipcm_common.h, ipcm_anonymous.h

【注意】

无

【举例】

无

【相关主题】

无

3.3.3 ipcm_anon_register_handle

【描述】

注册 ipcm anon handler。

【语法】

```
s32 ipcm_anon_register_handle(ANON_MSGPROC_FN handler, void *data);
```

【参数】

参数名称	描述	输入/输出
handler	anon handler	输入
data	handler 应用数据	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: ipcm_common.h, ipcm_anonymous.h

【注意】

无

【举例】

无

【相关主题】

无

3.3.4 ipcm_anon_send_msg

【描述】

通过 mailbox 发送 msg，该接口会检查 msg port id 是否是属于 anon 的 port id。

【语法】

```
s32 ipcm_anon_send_msg(MsgData *msg);
```

【参数】

参数名称	描述	输入/输出
msg	待发送的 msg 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: ipcm_common.h, ipcm_anonymous.h

【注意】

无

【举例】

无

【相关主题】

无

3.3.5 ipcm_anon_send_param

【描述】

当 msg 不超过 4byte 时，可以通过该接口发送 4byte 的 param。

【语法】

```
s32 ipcm_anon_send_param(u8 port_id, u8 msg_id, u32 param);;
```

【参数】

参数名称	描述	输入/输出
port_id	port id, 需要属于 anon 的范围	输入
msg_id	msg id, 范围是 [0,127]	输入
param	param 内容	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: ipcm_common.h, ipcm_anonymous.h

【注意】

无

【举例】

无

【相关主题】

无

3.3.6 ipcm_anon_get_user_addr

【描述】

返回对应物理地址的虚拟地址, 物理地址的范围必须是 alios 管理的地址。

在 linux 侧, 将从 alios 拿到的物理地址转成虚拟地址;

在 alios 侧, 因为 alios 没有开 mmc, 所以直接返回 alios 的物理地址。

【语法】

```
void *ipcm_anon_get_user_addr(u32 paddr);
```

【参数】

参数名称	描述	输入/输出
paddr	待转换的物理地址	输入

【返回值】

返回值	描述
NULL	失败, 物理地址不在 alios 管理的范围
其他	成功, 返回对应物理地址的虚拟地址

【需求】

- 头文件: ipcm_common.h, ipcm_anonymous.h

【注意】

无

【举例】

无

【相关主题】

无

3.4 数据类型

3.4.1 ipcm_anon_msg_t

【说明】

定义 ipcm anon 消息结构体，注册 anon handler 后，如果收到 anon 消息，handler 会返回改结构体的指针。

【定义】

```
typedef struct _ipcm_anon_msg_t {  
    u8 port_id;  
    u8 msg_id : 7;  
    u8 data_type : 1;  
    void *data;  
    u32 size;  
} ipcm_anon_msg_t;
```

【成员】

成员名称	描述
port_id	port 号
msg_id	msg 号
data_type	msg 数据类型 0: param 存的是 share memory pool 的 pos 和 size 1: param 存的是 msg 的内容，msg 内容不超过 4bytes
data	根据 data_type 不同，返回值不同 0: 返回 msg 的地址 1: 返回实际参数
size	根据 data_type 不同，返回值不同 0: 返回 msg 的实际长度 1: 固定为 4

【注意事项】

无

【相关数据类型及接口】

无

3.4.2 ANON_MSGPROC_FN

【说明】

定义 ipcm anon 的 handler

【定义】

```
typedef s32 (*ANON_MSGPROC_FN)(void *priv, ipcm_anon_msg_t *data);
```

【成员】

成员名称	描述
priv	用户数据指针；注册 handle 时传入的数据指针
data	接收到的 msg 指针

【注意事项】

无

【相关数据类型及接口】

无

3.5 错误码