|  |  |
| --- | --- |
| **密级状态：**绝密( ) 秘密( ) 内部资料( ) 公开( √ ) | |
|  |  |
| ISP IPC模块框架说明及接口规范 | |
| |  |  |  | | --- | --- | --- | | **文件状态：**  [√] 草稿  [ ]正在修改  [ ] 正式发布 | **文件标识：** | RK-XX-XX-XXX | | **当前版本：** | X.Y.Z | | **作 者：** | Author | | **完成日期：** | 2019-01-01 | | **审 核：** | Reviewer | | **审核日期：** | 201X-XX-XX |   （产品一部，三部） | |
|  |  |
| 福州瑞芯微电子股份有限公司  Fuzhou Rockchip Electronics Co.,Ltd. |  |

**目录**

**[1 框架概述： 2](#_Toc29422)**

[1.1 框架图： 2](#_Toc15127)

[1.2 框架说明： 2](#_Toc22785)

**[2 接口规范： 3](#_Toc27334)**

[2.1 应用层接口实现规范： 3](#_Toc18137)

[2.2 接口层规范： 4](#_Toc17063)

[2.3 协议层的规范： 4](#_Toc4763)

[2.4 uAPI层的规范： 5](#_Toc15959)

**[3 源码说明： 5](#_Toc30542)**

[3.1 Isp2-ipc源码结构说明： 5](#_Toc11032)

[3.2 CallFunIpc源码结构体说明： 6](#_Toc31260)

**免责声明**

本文档按“现状”提供，福州瑞芯微电子股份有限公司（“本公司”，下同）不对本文档的任何陈述、信息和内容的准确性、可靠性、完整性、适销性、特定目的性和非侵权性提供任何明示或暗示的声明或保证。本文档仅作为使用指导的参考。

由于产品版本升级或其他原因，本文档将可能在未经任何通知的情况下，不定期进行更新或修改。

**商标声明**

“Rockchip”、“瑞芯微”、“瑞芯”均为本公司的注册商标，归本公司所有。

本文档可能提及的其他所有注册商标或商标，由其各自拥有者所有。

**版权所有 © 2019福州瑞芯微电子股份有限公司**

超越合理使用范畴，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。

福州瑞芯微电子股份有限公司

Fuzhou Rockchip Electronics Co., Ltd.

地址： 福建省福州市铜盘路软件园A区18号

网址： [www.rock-chips.com](http://www.rock-chips.com)

客户服务电话： +86-4007-700-590

客户服务传真： +86-591-83951833

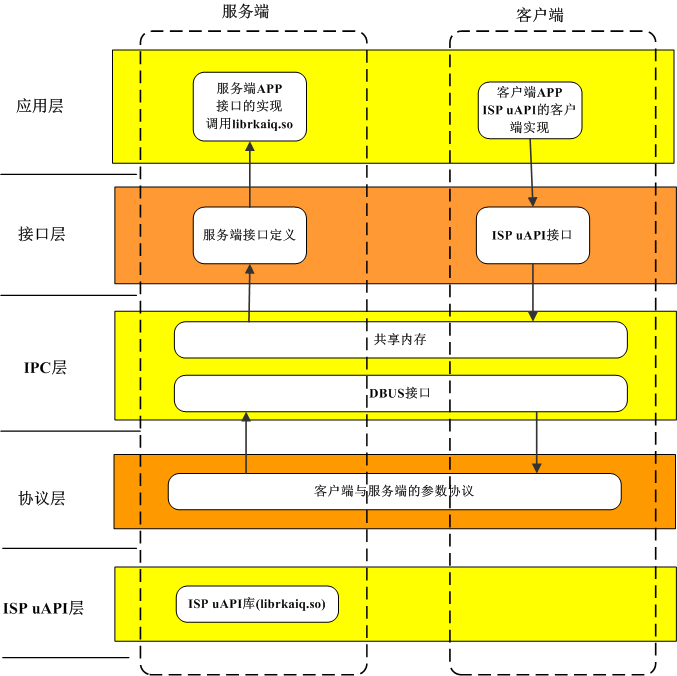
客户服务邮箱： fae@rock-chips.com

**修改记录**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本号 | 作者 | 修改日期 | 修改说明 | 备注 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# 框架概述：

## 框架图：



## 1.2 框架说明：

* 应用层：

服务端：

实现服务端的接口定义，主要基于uAPI接口定义的，在每个接口的末尾添加\_ipc

接口直接调用librkaiq.so的相关接口. 由于服务端的接口直接调用librkaiq.so

所以接口的名字不能跟uAPI接口一样。要不然编译会有问题。

客户端：

uAPI接口的客户端实现，如果客户原来不是用IPC模式，如果要切回IPC模式

只要依赖库改成libispclient.so. 调用的函数接口可以不要改动。实现无缝切换。

* 接口层：

服务端：

基于uAPI接口重新定义了一套接口。在原来的uAPI的接口的基础上，添加“\_ipc”

客户端：

直接采用uAPI接口定义。

* IPC层：
* 协议层：

基于uAPI接口参数，定义的结构体。

* ISP uAPI层：

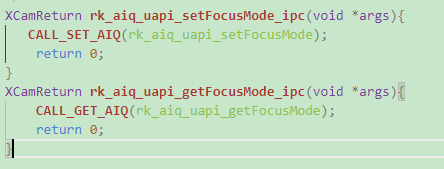
Librkaiq.so对外开放的接口。

# 接口规范：

## 2.1 应用层接口实现规范：

服务端：服务端主要实现interface目录下的接口定义。服务端根据set和get属性的方法，分别调用：CALL\_SET\_AIQ(uAPI接口名)和CALL\_SET\_AIQ(uAPI的接口名)。

比如：

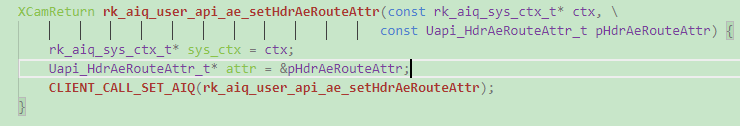


客户端：实现uAPI的.h文件接口的实现。服务端根据set和get属性的方法，

分别调用CLIENT\_CALL\_SET\_AIQ(uAPI接口名)和

CLIENT\_CALL\_SET\_AIQ(uAPI).如果参数名不是attr或者sys\_ctx.

参数名需要做下转换，比如：



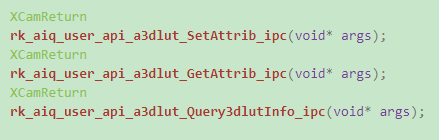
2.2 接口层规范：

服务端：

定义规范：uAPI接口名+\_ipc+(void \*args)

void\* args:共享内存结构体指针。

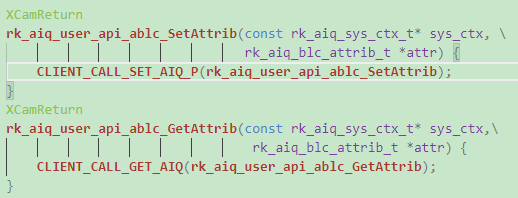
比如：



客户端：

直接采用uAPI接口进行实现，不要进行任何修改。

比如：



2.3 协议层的规范：

协议层主要根据接口的参数进行定义的。定义协议的时候，不能出现指针（除rk\_aiq\_sys\_ctx\_t\* sys\_ctx）。如果有返回值需要添加返回值定义。

定义规则：

Typedef struct uAPI接口名 {

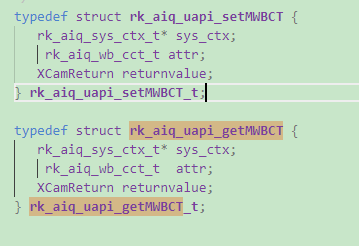
rk\_aiq\_sys\_ctx\_t\* sys\_ctx；

参数2；

xCamReturn returnvalue;

}

比如：



注意：每个接口定义一个协议文件。

2.4 uAPI层的规范：

uAPI层提供出来的参数结构体不能是指针。如果是指针共享内存拷贝会存在问题。。

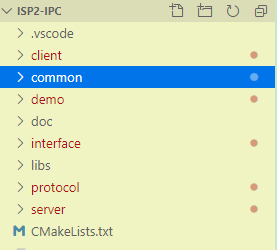
# 源码说明：

源码主要由两部分组成，isp2-ipc和CallFunIpc.

Isp2-ipc主要是rkaiq的IPC实现。

CallFunIpc主要是共享内存和DBUS的接口封装。

3.1 Isp2-ipc源码结构说明：



Client: 客户端uAPI接口实现，编译将生成libispclient.so

Common:存放公共文件。

Demo：客户端的demo实现，主要原来libispclient.so

Doc: 存放相关文档

interface：服务端的接口定义。

Libs:依赖的库文件，比如librkaiq.so

Protocol:存放协议文件，每个接口一个协议文件。

Server:服务端的接口实现。

源码地址： ssh://10.10.10.29:29418/linux/external/isp2-ipc

3.2 CallFunIpc源码结构体说明：

3.2.1、文件说明：

demo:提供客户端和服务端的demo代码。

call\_fun\_ipc.c：供客户端和服务端调用的API。

dbus.c：实现dbus通信的API，主要供用call\_fun\_ipc.c使用。

shared\_memory.c：实现进程间内存共享API，主要供用call\_fun\_ipc.c使用。

3.2.2、流程说明：

客户端：初始化时先调用call\_fun\_ipc\_client\_init(DBUS\_NAME, DBUS\_IF, DBUS\_PATH, SHARE\_PATH, 1);，然后才可以调用call\_fun\_ipc\_call((char \*)\_\_func\_\_, &para, sizeof(struct Examples\_s), 1);，在应用关闭时一定要调用call\_fun\_ipc\_client\_deinit();。

服务端：初始化时先调用call\_fun\_ipc\_server\_init(map, sizeof(map) / sizeof(struct FunMap), DBUS\_NAME, DBUS\_IF, DBUS\_PATH, 0);，在应用关闭时一定要调用call\_fun\_ipc\_server\_deinit();。

DBUS：demo里面有一个dbus的配置文件，应用可根据自己需求修改dbus的名字，名字要与服务器和客户端初始化时一致。

3.2.3、API说明：

void call\_fun\_ipc\_server\_init(struct FunMap \*map, int num, char \*dbus\_name, char \*dbus\_if, char \*dbus\_path, int needloop);和void call\_fun\_ipc\_client\_init(char \*dbus\_name, char \*dbus\_if, char \*dbus\_path, char \*share\_path, int needloop);，如果你的应用已经有运行g\_main\_loop\_run，needloop = 0，如果没有就设1；

3.2.4、结构体说明：

struct FunMap{

char \*fun\_name; 客户端的函数名字。

int (\*fun)(void \*); 服务端对应的函数指针。

};

# 4.后续还需解决的问题：

* 第三方算法库的加载接口需要重新定义。
* 部分算法库文件还存在指针，需要做重新定义。