**文档名称：**

ZDYZ-Z(C)鸿蒙版支架控制器DevMng，LibDevMng，LibA7DriverApp，v4driver接口设计文档

**摘 要：**

本文档只适用于ZDYZ-Z(C)鸿蒙版支架控制系统（项目编号YX230301），用于描述软件程序接口，对于其他项目不具备指导和约束作用。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 项目编号 | YX230301 | 项目名称 | ZDYZ-Z(C)鸿蒙版支架控制系统项目 | | |
| 当前版本 | S0 | 取代版本 |  | | |
| 作 者 |  | 审 批 |  | 批 准 |  |
| 版本历史 | | | | | |
| 版本 | 作者 | 参与者 | 完成日期 | 备注 | |
| S0 | 郑志春 |  | 2024.03.15 | 初始建立 | |

**目 录**

[1 概述 3](#_Toc15189)

[1.1 目的 3](#_Toc17811)

[1.2 适用范围 3](#_Toc699)

[1.3 参考资料 3](#_Toc13639)

[1.4 定义 4](#_Toc9861)

[2 开发及运行环境 4](#_Toc9049)

[2.1 硬件平台 4](#_Toc14758)

[2.2 软件开发环境 4](#_Toc5393)

[2.3 运行环境 4](#_Toc6598)

[3 系统架构描述 5](#_Toc12538)

[3.1 功能概述 5](#_Toc31961)

[3.2 设计思路 6](#_Toc8313)

[3.2.1 Linux系统层 6](#_Toc11469)

[3.2.2 平台维护程序及平台通用服务 7](#_Toc15204)

[3.2.3 产品应用层 7](#_Toc26967)

[3.2.4 启动流程 9](#_Toc13572)

[3.3 设计约束 10](#_Toc26974)

[3.3.1 遵循标准 10](#_Toc32352)

[3.4 软件系统上下文定义 10](#_Toc156)

[3.4.1 硬件限制 11](#_Toc28624)

[3.4.2 技术限制 11](#_Toc21356)

[3.4.3 其它 12](#_Toc7199)

[4 功能实现 12](#_Toc4574)

[4.1 应用进程(逻辑与主界面)间通信方式 12](#_Toc458)

[4.1.1 共享内存 13](#_Toc15490)

[4.1.2 消息机制设计 16](#_Toc13078)

[4.2 设备管理程序与应用及驱动通信方式 19](#_Toc18578)

[4.2.1 总体设计思路 19](#_Toc5152)

[4.2.2 共享内存 22](#_Toc26209)

[4.2.3 消息机制 26](#_Toc1549)

[4.3 逻辑程序功能实现 28](#_Toc1074)

[4.3.1 功能概述 28](#_Toc20730)

[4.3.2 设计思路 29](#_Toc14571)

[4.3.3 启动流程 30](#_Toc27250)

[4.3.4 实现方案 32](#_Toc23870)

[4.4 界面显示（UI）程序概要设计 39](#_Toc25701)

[4.4.1 总体概述 39](#_Toc3903)

[4.4.2 设计思路 39](#_Toc28914)

[4.4.3 显示（View）描述 40](#_Toc31617)

[4.4.4 服务（Presenter）描述 43](#_Toc2447)

[4.4.5 数据模型（Model）描述 45](#_Toc24614)

[4.4.6 子线程描述 46](#_Toc31819)

[4.5 V4架间协议栈功能实现及驱动应用共享库提取 47](#_Toc8597)

[4.5.1 V4架间协议栈功能实现 47](#_Toc30778)

[4.5.2 驱动共享库提取 49](#_Toc17838)

[4.6 驱动程序(除V4Driver)功能实现 53](#_Toc14939)

[4.6.1 驱动程序总体描述 53](#_Toc4521)

[4.6.2 设备间驱动 54](#_Toc18027)

[4.6.3 Usart驱动 57](#_Toc10260)

[4.6.4 模拟量驱动 61](#_Toc13181)

[4.6.5 I/O驱动 62](#_Toc12162)

# 概述

## 目的

本文档主要描述ZDYZ-Z(C)鸿蒙版支架控制系统(简称A7版控制器，下同)程序中关于DevMng，LibDevMng，LibA7DriverApp，v4driver程序的接口使用和流程概述。

## 适用范围

本文档只适用于A7版支架控制器项目，用于指导其软件程序设计，对于其他项目不具备指导和约束作用；本文档仅限于项目组成员及相关领导查阅。

## 参考资料

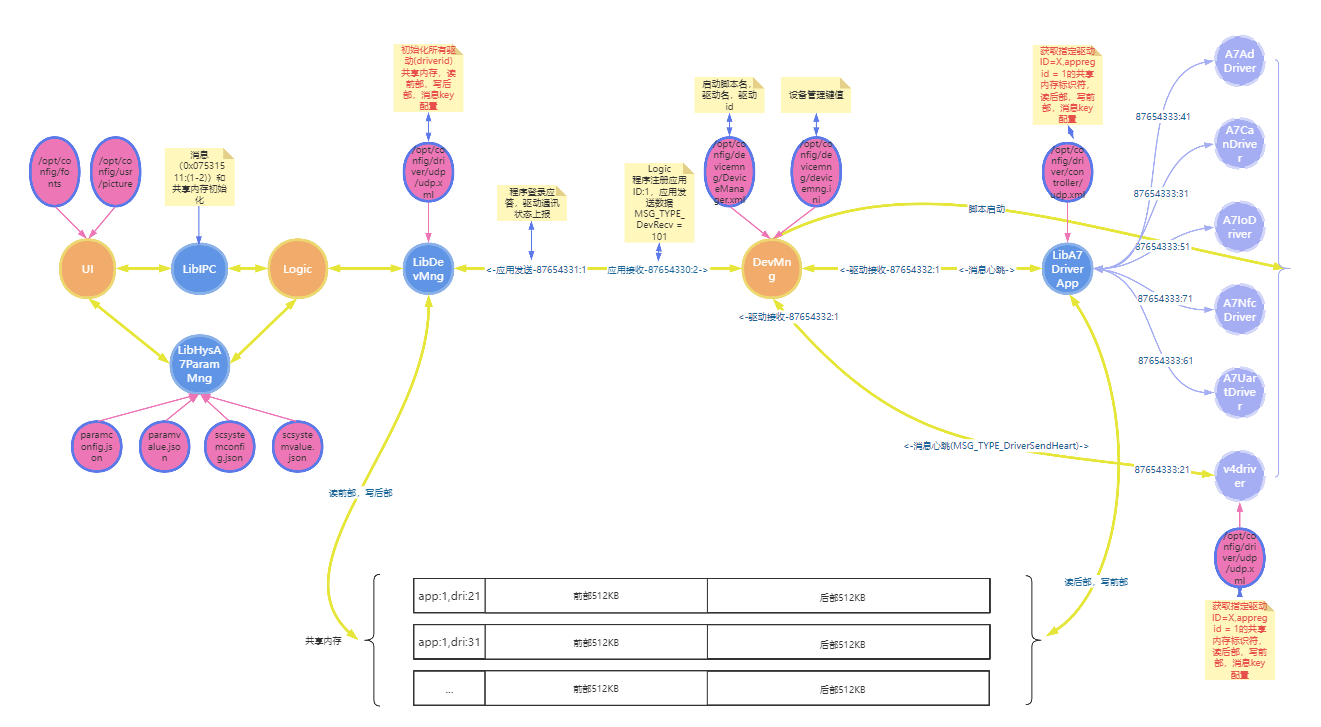
《ZDYZ-Z(C)鸿蒙版支架控制系统软件概要设计S2》

# 系统架构描述

### **启动流程**

各进程间的启动顺序，以及各个进程的软件流程图，详细描述如下：





各进程间通讯关系详细描述如下：

以下章节，将对各个模块间的通信及交互方式、各个功能模块的具体实现，进行详细描述。

## 设备管理程序DevMng与应用及驱动通信方式

### 设计思路

设备管理DevMng与驱动程序，在系统架构中位置如下：

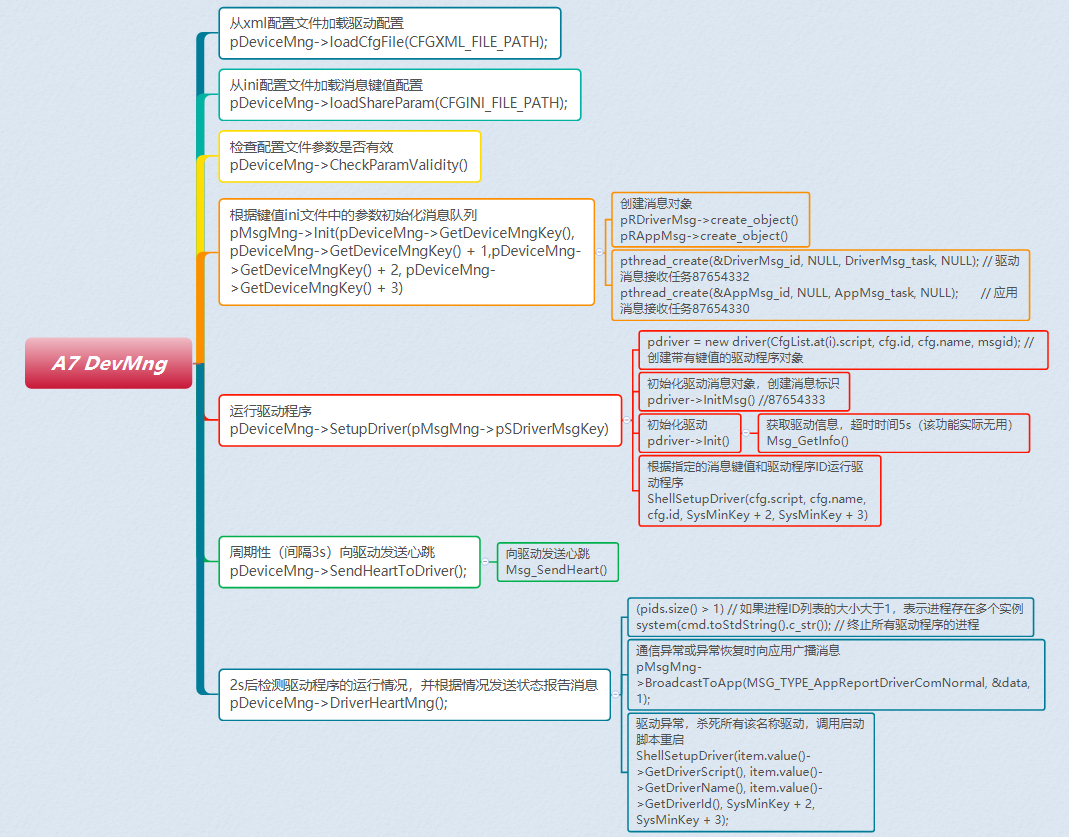


系统设计时采用“设备管理”思想，即：将所有驱动和设备上的输入、输出点统一排序，由设备管理进行管控，对应用层提供统一的接口，包括但不限于数据采集、状态上报、输出控制等操作，可以让应用层无需关心驱动间数据格式的差异。同时，当驱动发生紧急事件或异常时，以消息的形式上报给应用程序。总体设计思路如下：



设备管理程序DevMng其设计思路，见下图所示:

 设备管理程序DevMng程序运行流程，见下图所示:



功能如下：

1. 设备管理程序初始化时，根据自身配置文件(/opt/config/devicemng/DeviceManager.xml)及启动脚本文件(/opt/config/devicemng/\*\*\*\_dykss\_start.sh)，启动相应驱动程序；每一个驱动启动时，传递驱动号、驱动名称、启动脚本路径、消息接收键值(固定为87654333)等信息；
2. 所有的驱动启动后，设备管理程序以消息的形式每3s向驱动发送心跳，应答消息在LibA7DriverApp库程序中自动完成。设备管理程序负责监视所有的驱动，如驱动异常退出或同一驱动存在多个进程号，将杀死并重新启动该驱动程序，同时将异常驱动号通过LibDevMng库程序接口上报给逻辑程序。
3. 应用程序(目前仅逻辑程序使用)启动时，通过调用设备管理库程序LibDevMng中的(int registeappid(int regappid, int forcereg = 0))初始化函数，注册自身应用编号(逻辑为1)、调用DeviceMngApi::setCallBack\_R\_Shm设置共享内存数据接收回调函数、调用DeviceMngApi::setCallBack\_R\_Msg设置消息接收的回调函数，间接与驱动程序进行通信，通过读取或写入共享内存、发送或接收消息形式，实现与驱动设备的交互。同时当应用程序退出时也会调用设备管理库LibDevMng中logout()方法注销自身应用。
4. 驱动向应用发送共享内存数据时，驱动程序接收到设备上报的数据，转化为统一的数据格式，放入对应的共享内存状态区，并向该应用的设备管理库程序LibDevMng发送信号量(LibDevMng中函数为fore\_sem.sem\_v()，V-释放资源)。后者接收到信号量后(LibDevMng中函数为fore\_sem.sem\_p()，P-申请资源)，读取共享内存，并调用应用程序设置的接收共享内存数据回调函数(DeviceMngApi::setCallBack\_R\_Shm)，从而使应用程序能够读取到该数据。
5. 应用向驱动发送共享内存数据时，应用程序通过调用设备管理库程序LibDevMng提供的数据下发函数(DeviceMngApi::writeshmp)，将下发数据写入到共享内存控制区。设备管理库程序LibDevMng同时发送信号量(back\_sem.sem\_v()，V-释放资源)。驱动程序接收到信号量后(back\_sem.sem\_p()，P-申请资源)，读取共享内存获取数据，并经协议转换后，发送给特定设备。
6. 应用向驱动发送消息时，应用程序通过调用设备管理库程序LibDevMng提供的消息下发函数(DeviceMngApi::writemsgp)发送消息，经DevMng将类型为MSG\_TYPE\_DevRecv的消息转发给LibA7DriverApp。驱动程序收到应用程序发送的消息(由Ydshrdlib::setMsgCallBack设置的消息接收回调函数处理数据)，并完成相应的功能。
7. 驱动向应用发送消息时，驱动程序调用Ydshrdlib::sendMsgDataFunc函数发送类型为MSG\_TYPE\_DevAutoReport的消息，经DevMng转发后向设备管理库程序LibDevMng发送消息。设备管理库程序LibDevMng接收后，调用接收消息的处理方法(DeviceMngApi::setCallBack\_R\_Msg)，从而通知应用程序。
8. 设备管理程序会监测驱动程序运行状态，设备管理程序将驱动程序启动后会每隔3s给驱动发送心跳信息，若等待2s仍未应答且判断驱动进程号为空时则将驱动程序进行重启。

### 共享内存

#### 实现方案

应用通过调用DeviceMngApi::registeappid注册函数，获取n(n为应用个数,下同)个共享内存控制区key、n个共享内存状态区key、n个发送信号量key、n个接收信号量key、n个发送消息key、n个接收消息key，驱动程序中，根据不同应用程序，提供一份共享内存。(本项目中，相关的应用只有逻辑一个)



每一份共享内存，包括状态区（输入、输出状态或数值）和下发控制区，其结构如下图：



* 状态区（驱动上报的状态信息）

每一个驱动都拥有一个状态数据区，依据设定的点的数量创建共享内存区，且大小不固定，可以配置（目前配置为512KB）。驱动程序上传数据时，读取状态区中的写指针，并把数据填入写指针指向的位置；同时将写指针指向下一个区域，并发送信号量。应用程序通过设备管理库程序LibDevMng接收到信号量，根据驱动编号读取特定驱动的共享内存,通过读取每一包数据的写入地址以及数据长度，确定每一包数据的起始和结束，并以“设备号+子设备号+点号+数据类型+数据”的方式解析数据，并向上提供固定的访问接口。

* 输出控制区（逻辑控制输入）

每一个驱动都拥有一个输出控制区，且大小不固定，可以配置（目前配置为512KB）。应用程序下发数据时，读取输出控制区中的写指针，并把下发的数据填入写指针指向的位置；同时将写指针指向下一个区域，并发送信号量。驱动程序接收到信号量后，根据读指针指向的位置取出此长度的数据，解析下发控制的设备号、子设备号、点号、数据，进行数据处理，或进行控制指令的下发。

#### LibDevMng外部接口

1. 获取设备管理库指针

/\*\*

\* @brief DeviceMngApi::getInston 获取DeviceMngApi类的单例

\* @return DeviceMngApi类的单例

\*/

static DeviceMngApi \*getInston();

1. 主逻辑进程或其他程序向数据通信服务进行注册

/\*\*

\* @brief DeviceMngApi::registeappid 主逻辑进程或其他程序向数据通信服务进行注册

\* @param regappid 应用程序的编号

\* @param forcereg 是否首次注册注册，forcereg 设置为 1，会收到DevMng返回的MSG\_TYPE\_AppLogIn消息应答

\* @return 成功返回 0；失败返回 -1

\*/

int DeviceMngApi::registeappid(int regappid, int forcereg)

1. 主逻辑进程或其他程序接收共享内存数据回调函数

/\*\*

\* @brief DeviceMngApi::setCallBack\_R\_Shm 设置共享内存接收回调函数

\* @param driverid 驱动ID

\* @param int (\*\_readshm)(uchar, uchar, ushort, ushort, uchar \*, int) 回调函数

\* @return 成功返回0，失败返回-1

\*/

int setCallBack\_R\_Shm(uchar driverid, int (\*readshm)(uchar devtype1, uchar devtype2, ushort numb, ushort frametype, uchar \*data, int len));

1. 主逻辑进程或其他程序接收消息回调函数

/\*\*

\* @brief DeviceMngApi::setCallBack\_R\_Msg 设置消息接收回调函数

\* @param driverid 驱动ID

\* @param int (\*\_msgcallback)(uint8\_t, uint8\_t, ushort, uchar \*, int) 回调函数

\* @return 成功返回0，失败返回-1

\*/

int setCallBack\_R\_Msg(uchar driverid, int (\*\_msgcallback)(uint8\_t devtype1, uint8\_t devtype2, ushort numb, uchar \*data, int len));

1. 主逻辑进程或其他程序下发控制指令函数

/\*\*

\* @brief DeviceMngApi::writeshmp 向共享内存写入数据，主逻辑进程或其他程序下发控制指令

\* @param driverid 驱动ID(程序编号)

\* @param devtype1 设备号

\* @param devtype2 子设备号

\* @param numb 点号

\* @param frametype 功能码

\* @param data 数据

\* @param length 数据长度

\* @return 成功返回0，失败返回-1

\*/

int writeshmp(uchar driverid, uchar devtype1, uchar devtype2, uchar numb, u16 frametype, u\_int8\_t \*data, int length);

1. 主逻辑进程或其他程序下发消息函数

/\*\*

\* @brief DeviceMngApi::writemsgp 向驱动发送消息数据

\* @param driverid 驱动ID(程序编号)

\* @param devtype1 设备号

\* @param devtype2 子设备号

\* @param numb 点号

\* @param data 数据

\* @param length 数据长度

\* @return 成功返回0，失败返回-1

\*/

int writemsgp(uchar driverid, uchar devtype1, uchar devtype2, uchar numb, u\_int8\_t \*data, int length);

### 消息机制

除共享内存通信方式外，设备管理程序DevMng其他非周期性的交互采用消息通信方式。包括驱动状态上报、文件传输及结果反馈等等。

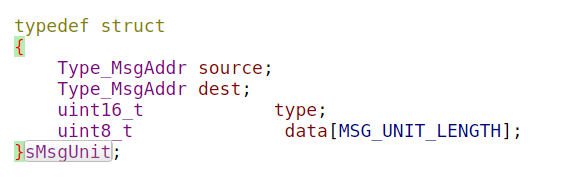
#### 数据结构

其消息结构如下图所示：



其中，消息源和目的：对应用程序为进程id号；对驱动程序为驱动号+设备号+子设备号+点号，type对应消息的类型(消息的功能码)。

消息的数据结构如下图所示:



消息类型(消息的功能码)的枚举定义如下:



具体数据解析中，会对上述data[MSG\_UINT\_LENGTH]数据进行具体划分，结构体如下:

typedef

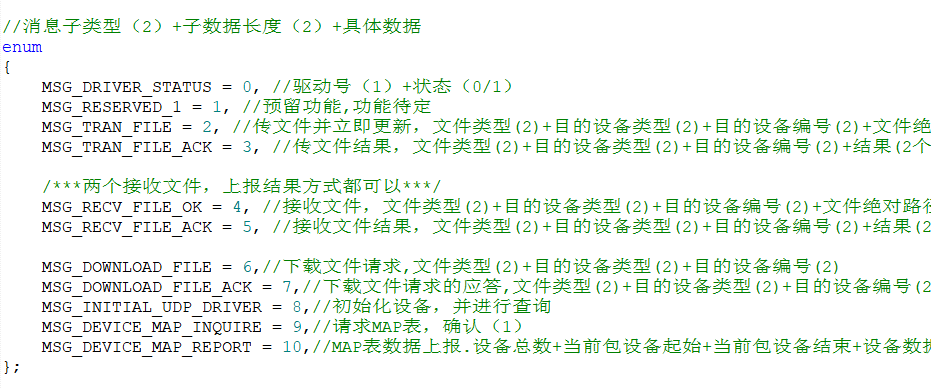
{

uint16\_t dataType;

uint8\_t data[MSG\_UINT\_LENGTH-0x02];

}sMsgDataType;

其中，消息子类型dataType(数据的功能码)的枚举定义如下图所示:



## 驱动共享库LibA7DriverApp

### 驱动共享库

上述软件架构中的应用层负责和逻辑进行通信和xml配置文件解析，通讯方式分为共享内存和消息。每个共享内存分为控制区和状态区，控制区为逻辑提供控制输入，状态区为控制器上报的状态信息；消息为点对多的方式，驱动程序会根据数据内容分别给每个逻辑发送消息，也会收到来自不同逻辑的消息。

#### 功能描述

* LibA7DriverApp驱动共享库程序（以下简称共享库）对驱动程序的应用层进行抽象，能够供所有驱动程序进行使用。
* 共享库为驱动程序提供接口，包括注册驱动号、消息数据收发、共享内存收发。
* 共享库负责解析/opt/config/driver/controller/udp.xml配置文件，为共享内存和信号量提供key，提供共享内存大小。
* 共享库通过共享内存和消息的方式同设备管理程序DevMng、逻辑进行通讯。
* 维护与设备管理程序DevMng的心跳。

#### 设计思路

* 共享库程序对消息和共享内存结构进行封装，对外提供统一接口函数。驱动通过注册回调函数接收消息和共享内存数据，通过调用接口函数发送消息和共享内存数据。
* 库程序内部通过消息维持和设备管理程序的心跳，设备管理程序DevMng能在驱动程序掉线时对程序进行重启。
* 根据配置文件创建一块共享内存，内存分为发送区和接收区。发送区和接收区的大小通过配置文件指定，且内存区中数据存储方式为不定长存储。
* 共享库程序被驱动程序调用，通过注册自身驱动号，接收消息key，发送消息key，接收消息类型，发送消息类型和读取xml文件完成初始化。之后会进行共享内存的创建，开始和设备管理程序DevMng进行通讯，进而和逻辑进行通讯。接收到共享内存和消息数据时调用回调函数交给具体解析函数进行处理，驱动程序调用接口发送消息和共享内存数据。

#### 数据结构

* xml文件结构描述

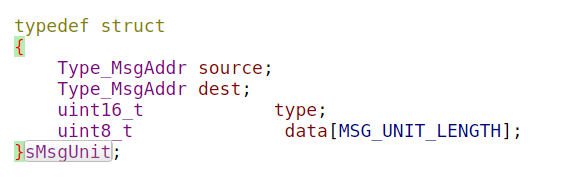


|  |  |  |  |
| --- | --- | --- | --- |
| devtype | 预留，与appregid相同 | appregid | 应用号 |
| shmkey | 共享内存key | todripsem | 给驱动的信号量 |
| tologicsem | 给逻辑的信号量 | todripmsg | 预留 |
| tologicmsg | 预留 | shmsize | 共享内存大小（发送区和接收区） |
| forepart | 发送区内存大小 | driverid | 驱动号 |

xml文件配置了某个逻辑对应的n个驱动和某个驱动对应的n个逻辑，共享库根据驱动号创建与之匹配逻辑的共享内存块，并根据分配共享内存大小，发送区和接收区的大小，指定了共享内存和消息的key值。

* 消息队列设计





逻辑程序和驱动程序不能直接通讯，需要设备管理程序进行转发，且使用统一的数据结构。设备管理程序DevMng与驱动的消息维护由两个消息key进行收发，设备管理向驱动发送消息时候类型为对应的驱动号；驱动给设备管理程序DevMng发送消息类型为固定值。

共享库程序为消息发送提供统一发送函数，在消息接收中根据appid和类型增加注册函数。在接收到消息数据时将数据进行解析并且调用与appid、类型都匹配的回调函数。

* 共享内存设计



逻辑程序和驱动程序采用直接通讯的方式。

库程序会根据驱动号对应几个逻辑程序创建几块共享内存，每块共享内存分为发送区和接收区，发送区和接收区数据结构一样，唯一区别在于发送区是驱动发送给逻辑的，接收区是驱动接收逻辑发送的数据。发送区分为索引区和数据区，索引区记录n组索引，每个索引记录读取的数据区地址和数据区长度。



共享内存 + 信号量为数据同步机制，在逻辑和驱动启动时候都会对内存块初始化。

库程序为发送共享内存数据提供统一发送接口，在接收中数据后会转化成设备+子设备+点+类型+数据的方式交给回调函数进行解析。

#### LibA7DriverApp外部接口

/\*\*

\* @brief Ydshrdlib::getInston 获取Ydshrdlib单例实例

\* @return Ydshrdlib单例实例

\*/

static Ydshrdlib \*getInston();

/\*\*

\* @brief Ydshrdlib::setShmCallBack 设置共享内存后部接收回调函数

\* @param appid 应用ID

\* @param type 功能码

\* @param pfunc 回调函数

\* @return 成功返回0，失败返回-1

\*/

int setShmCallBack(u8 appid, u16 type, CallBackFunc pfunc);

/\*\*

\* @brief Ydshrdlib::sendShmDataFunc 向共享内存前部写数据

\* @param appid 应用ID

\* @param type 功能码

\* @param device 设备号

\* @param childdevice 子设备号

\* @param point 点号

\* @param data 数据

\* @param len 数据长度

\* @return 成功返回0，失败返回-1

\*/

int sendShmDataFunc(u8 appid, u16 type, u8 device, u8 childdevice, u8 point, u8 \*data, int len);

/\*\*

\* @brief Ydshrdlib::setMsgCallBack 设置消息接收回调函数

\* @param appid 应用ID

\* @param type 功能码

\* @param pfunc 回调函数

\* @return 成功返回0，失败返回-1

\*/

int setMsgCallBack(u8 appid, u16 type, CallBackFunc pfunc);

/\*\*

\* @brief Ydshrdlib::sendMsgDataFunc 向应用发送消息

\* @param appid 应用ID

\* @param type 功能码

\* @param device 设备号

\* @param childdevice 子设备号

\* @param point 点号

\* @param data 数据

\* @param len 数据长度

\* @return 成功返回0，失败返回-1

\*/

int sendMsgDataFunc(u8 appid, u16 type, u8 device, u8 childdevice, u8 point, u8 \*data, int len);

/\*\*

\* @brief Ydshrdlib::InitLibParam 初始化Lib库参数

\* @param driverid 驱动ID(程序编号)

\* @param recvmsgkey 接收消息key

\* @param sendmsgkey 发送消息key

\* @param sendmsgtype 发送消息类型

\* @return 成功初始化返回0，失败返回-1

\*/

int InitLibParam(u8 driverid, int recvmsgkey = 87654333, int sendmsgkey = 87654332, u8 sendmsgtype = 1);

## V4Driver功能实现

#### 功能描述

* V4Driver驱动程序使用TCP/IP运输层的UDP协议传递架间链路数据通讯。
* 维护与设备管理程序DevMng的心跳。
* 通过共享内存和消息的方式同设备管理程序DevMng、逻辑进行通讯。
* 使用RSYNC服务文件传输协议实现文件更新。
* 使用802.1Q VLAN帧实现沿线配址功能。

#### 程序运行流程

DevMng程序通过/opt/config/devicemng/udp\_dykss\_start.sh文件启动V4Driver程序，并在启动时将驱动发送消息键值(87654332)，驱动发送消息类型(1)，驱动接收消息键值(87654333)，驱动ID号(21)，网卡名称传递给V4Driver程序

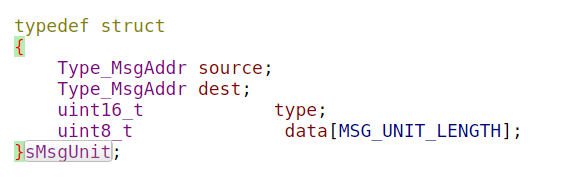
#### 数据结构

其消息结构如下图所示：



其中，消息源和目的：对应用程序为进程id号；对驱动程序为驱动号+设备号+子设备号+点号，type对应消息的类型(消息的功能码)。

消息的数据结构如下图所示:



消息类型(消息的功能码)的枚举定义如下:

