SHRD100-配置管理-设计说明

作者：滑国青

历史：2023.09.12 v0.1 创建

2023.09.18 v0.2 滑国青 （1）软件配置文件增加PbNetProto，组网通讯协议支持除了可以支持alink协议，还可以支持与无人机遥控器对接。

[前言 1](#_Toc145942169)

[接口说明 2](#_Toc145942170)

[头文件 2](#_Toc145942171)

[获取硬件的配置对象 ProductHwCfgGet() 2](#_Toc145942172)

[获取软件版本特性的配置对象ProductSwCfgGet() 2](#_Toc145942173)

[获取调试参数的配置对象 ProductDebugCfgGet() 3](#_Toc145942174)

[Json转pb对象 json\_string\_to\_message() 3](#_Toc145942175)

[pb对象转json message\_to\_json\_string() 3](#_Toc145942176)

[实现方案 4](#_Toc145942177)

[配置文件集合位置 4](#_Toc145942178)

[打包配置到rootfs 5](#_Toc145942179)

[编译pb描述文件 5](#_Toc145942180)

[编译linux\_app 5](#_Toc145942181)

[打包配置到rootfs 5](#_Toc145942182)

[系统启动时识别软硬件版本信息S01cfg\_check.sh 6](#_Toc145942183)

[linux\_app启动时读入配置 6](#_Toc145942184)

[调试时临时更改配置文件 7](#_Toc145942185)

[Linux\_app配置获取方法 7](#_Toc145942186)

[获取硬件配置对象 7](#_Toc145942187)

[获取软件配置对象 8](#_Toc145942188)

[获取调试配置对象 8](#_Toc145942189)

[配置说明 8](#_Toc145942190)

[增加组网通讯协议NET\_PROTO\_TYPE\_TRACER\_UAVCONTROLLER 8](#_Toc145942191)

[ProductSwCfg.proto 8](#_Toc145942192)

[支持NET\_PROTO\_TYPE\_TRACER\_UAVCONTROLLER的版本号定义为SHRD103L 10](#_Toc145942193)

[协议接收处理函数 10](#_Toc145942194)

[协议发送处理 11](#_Toc145942195)

[问题 12](#_Toc145942196)

[Protoc工具版本不对，导致编译失败 12](#_Toc145942197)

[解决办法： 12](#_Toc145942198)

[结束 12](#_Toc145942199)

# 前言

本配置管理设计要解决几个问题

1. 提供基本的json与pb的互转机制。配置信息以json或pb序列化文件方式提供。
2. 同一固件源码，适配不同硬件版本的能力。(比如tracer T2,T3硬件单板)
3. 同一固件源码，适配产品特性多系列的能力（比如tracer-s, tracer-p, tracer-uav）

（4） 提供基础的用户配置保存和读取能力。

# 接口说明

## 头文件

#include "json\_pb\_converter.h"

include\_directories(${CMAKE\_SOURCE\_DIR}/src/eap include/os\_adapter)

include\_directories(${CMAKE\_SOURCE\_DIR}/src/eap/alink\_pb)

## 获取硬件的配置对象 ProductHwCfgGet()

硬件配置对象里，定义所有gpio,串口等相关的配置。通过功能逻辑名字key,可以找到对应的物理参数。

/\*\*

\* @brief get ProductHwCfg Obj

\*

\* @param [in] filename file

\*

\* @return ProductHwCfg&

\*

\* @see linux\_app/src/eap/alink\_pb/ProductHwCfg.proto

\* @note

\* @warning

\*/

ProductHwCfg& ProductHwCfgGet();

## 获取软件版本特性的配置对象ProductSwCfgGet()

在硬件相同的情况下，通过软件版本特性参数，可以建构不同用途的版本，比如droneid版本，频谱版本，机载版本等。

/\*\*

\* @brief get ProductSwCfg Obj

\*

\* @param [in] filename file

\*

\* @return ProductSwCfg&

\*

\* @see linux\_app/src/eap/alink\_pb/ProductSwCfg.proto

\* @note

\* @warning

\*/

ProductSwCfg& ProductSwCfgGet();

## 获取调试参数的配置对象 ProductDebugCfgGet()

获取调试参数的配置对象。

定义了各种调试用途的标记文件或参数，方便调试。

/\*\*

\* @brief get ProductDebugCfg Obj

\*

\* @param [in] filename file

\*

\* @return ProductDebugCfg&

\*

\* @see linux\_app/src/eap/alink\_pb/ProductDebugCfg.proto

\* @note

\* @warning

\*/

ProductDebugCfg& ProductDebugCfgGet();

## Json转pb对象 json\_string\_to\_message()

json配置文本字符串格式 转 pb message对象

// C function declaration for converting JSON string to Protobuf message

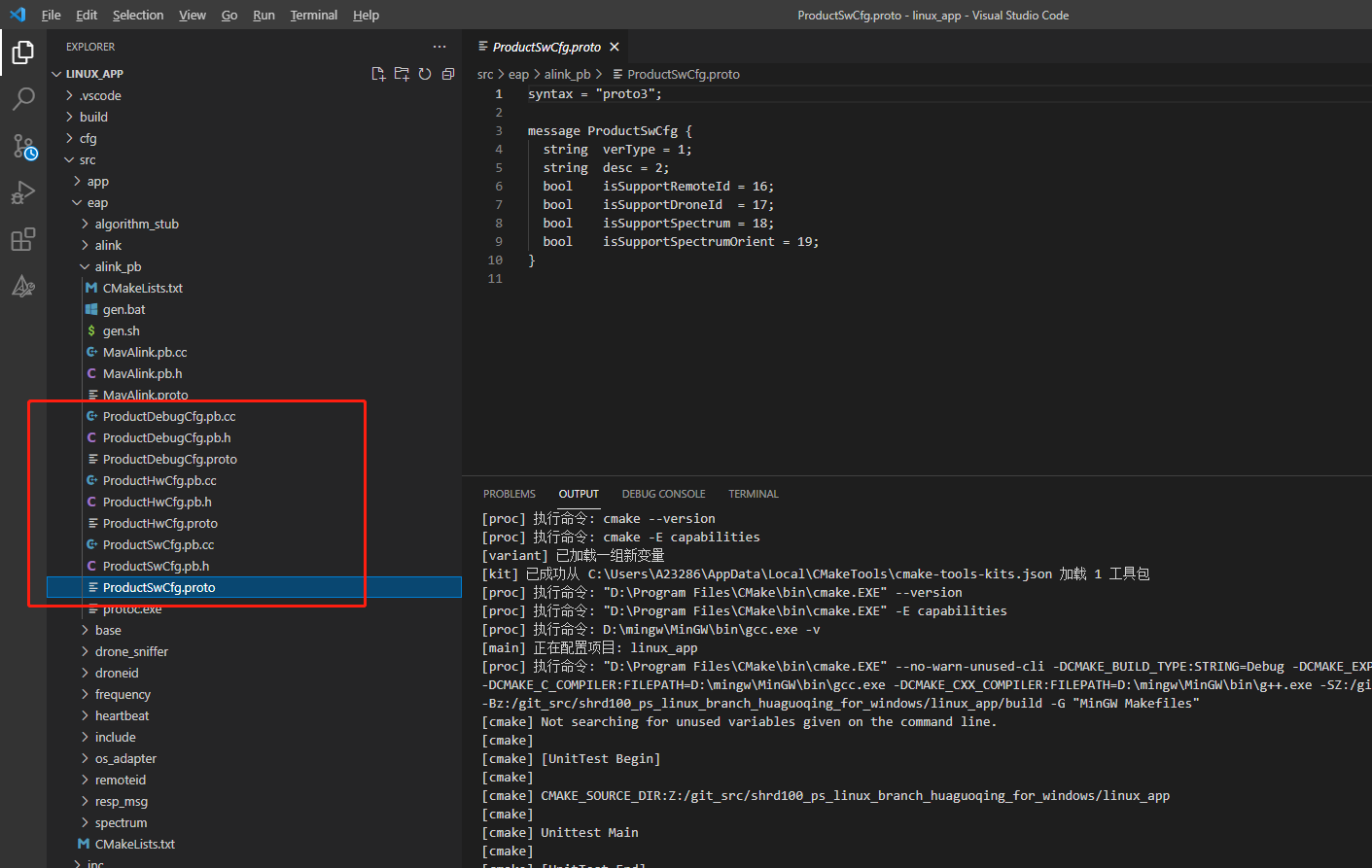
int json\_string\_to\_message(const char\* json\_string, void\* message);

## pb对象转json message\_to\_json\_string()

pb message对象 转json配置文本字符串格式

// C function declaration for converting Protobuf message to JSON string

int message\_to\_json\_string(const void\* message, char\*\* json\_string);



# 实现方案

## 配置文件集合位置

当前版本配置文件情况如下：

linux\_app/cfg

├── product\_hw\_cfg.json

├── product\_hw\_cfg.json.T2

├── product\_hw\_cfg.json.T3

├── product\_sw\_cfg.json

├── product\_sw\_cfg.json.SHRD101L

└── product\_sw\_cfg.json.SHRD102L

## 打包配置到rootfs

### 编译pb描述文件

(1)设置交叉编译工具链，以提供相应版本的pb编译工具protoc

source /opt/tracer/petalinux/2022.2/environment-setup-cortexa72-cortexa53-xilinx-linux

(2)重新编译proto文件

cd linux\_app/src/eap/alink\_pb

./gen.sh

### 编译linux\_app

省略

请参考linux\_app/README

或调用app\_build.sh

### 打包配置到rootfs

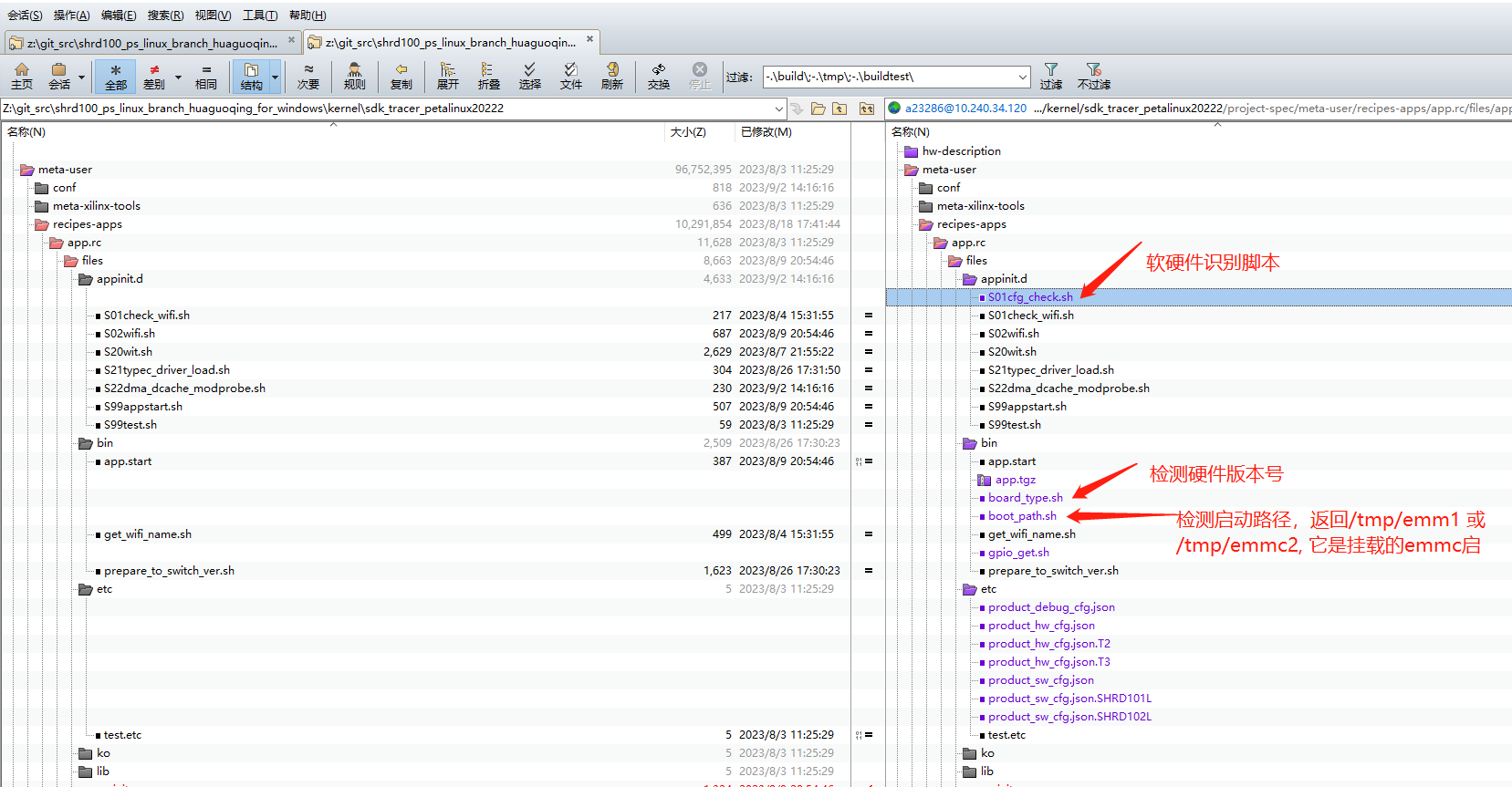
cd linux\_app

./app\_pack.sh

此脚本，会自动将linux\_app/cfg/\* 拷到kernel/sdk\_tracer\_petalinux20222/project-spec/meta-user/recipes-apps/app.rc/files/etc

在rootfs里，配置文件在/etc目录下

## 系统启动时识别软硬件版本信息S01cfg\_check.sh



S01cfg\_check.sh执行后，

/etc/product\_hw\_cfg.json便是正确的硬件配置文件

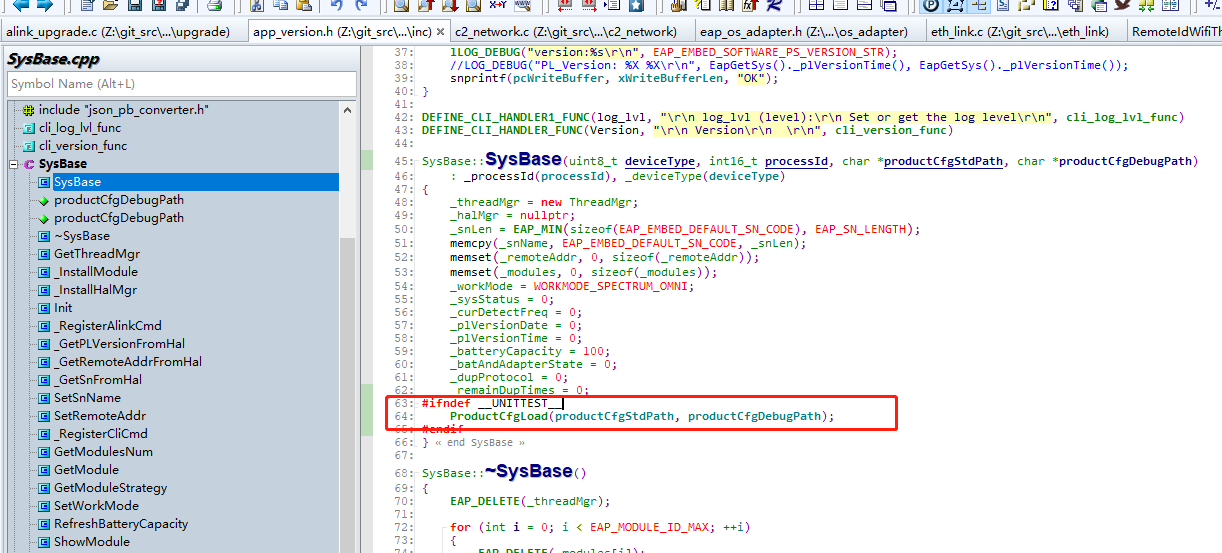
它是根据board\_type.sh返回的硬件版本字符串，找到对应的/etc/product\_hw\_cfg.json.XX ，然后拷贝为/etc/product\_hw\_cfg.json

/etc/product\_sw\_cfg.json便是正确的软件配置文件。

它是根据版本version的首段字符串，找相应后缀的/etc/product\_sw\_cfg.json.SHRDXXX ，然后拷贝为/etc/product\_sw\_cfg.json

## linux\_app启动时读入配置

请参考：ProductCfgLoad()



## 调试时临时更改配置文件

为方便调试时，临时更改配置, 拷贝1份配置文件，放在/tmp/emmc\_fat/目录下，并修改内容。

如软件配置：

cp /etc/ProductSwCfg.json /tmp/emmc\_fat/ ProductSwCfg.json

linux\_app在启动时，优先在/tmp/emmc\_fat下查找配置文件，然后才是/etc/下查找。

## Linux\_app配置获取方法

请参考此文件中的project\_cfg\_dump()

学习使用：linux\_app\src\eap\base\cfg\json\_pb\_converter.cpp

基本使用方法，就是每个配置对象，针对每个字段，如foo，都提供了foo(); set\_foo()函数接口。

所以，获取到配置对象后，直接通过字段相关的函数读写数据。

### 获取硬件配置对象

ProductHwCfg &hw = ProductHwCfgGet();

通过调用配置字段的函数，获取值

printf("producthwtype:%s\n",hw.producthwtype().c\_str());

### 获取软件配置对象

ProductSwCfg &sw = ProductSwCfgGet();

printf("vertype:%s,issupportdroneid:%d,issupportremoteid:%d,issupportspectrum:%d,issupportspectrumorient:%d\n",

sw.vertype().c\_str(),sw.issupportdroneid(),sw.issupportremoteid(),

sw.issupportspectrum(),sw.issupportspectrumorient());

### 获取调试配置对象

ProductDebugCfg &debug = ProductDebugCfgGet();

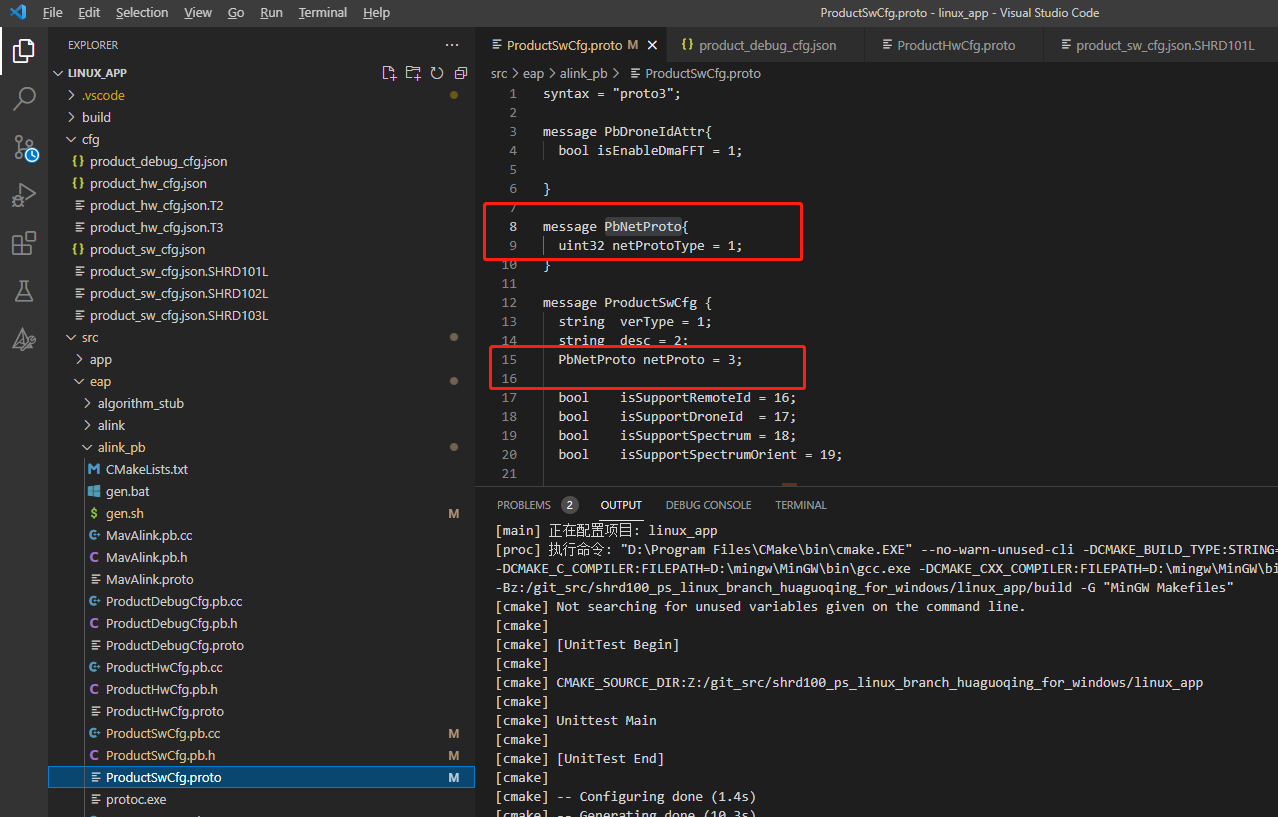
printf("flag\_debug\_disable\_buzz\_motor:%d\n",debug.flag\_debug\_disable\_buzz\_motor());

# 配置说明

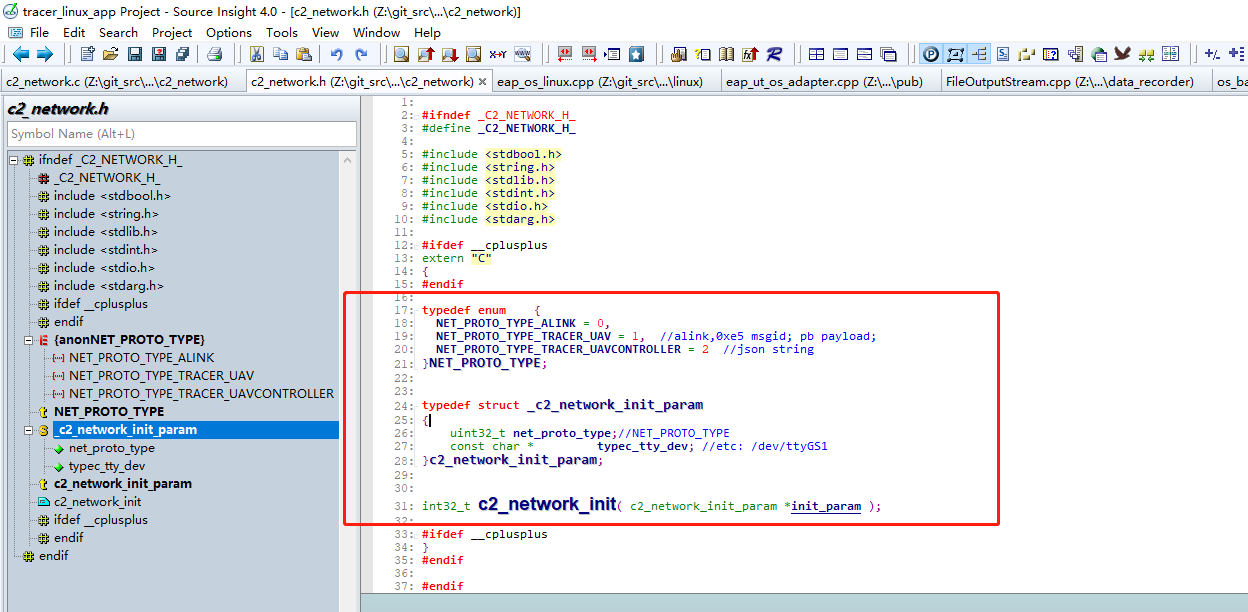
## 增加组网通讯协议NET\_PROTO\_TYPE\_TRACER\_UAVCONTROLLER

### ProductSwCfg.proto

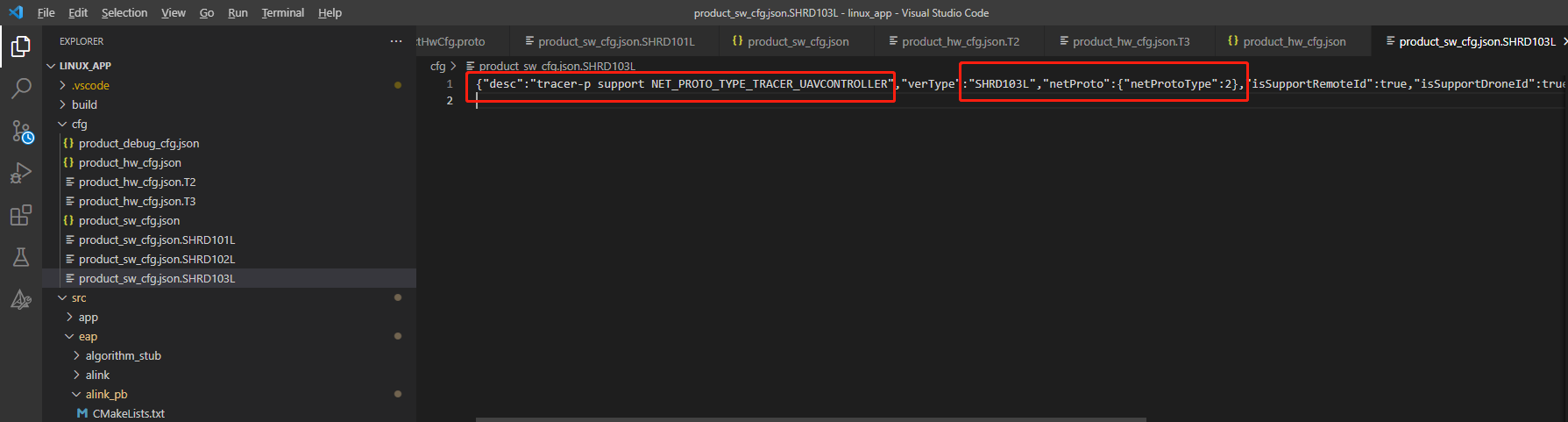
增加pbNetProto消息定义

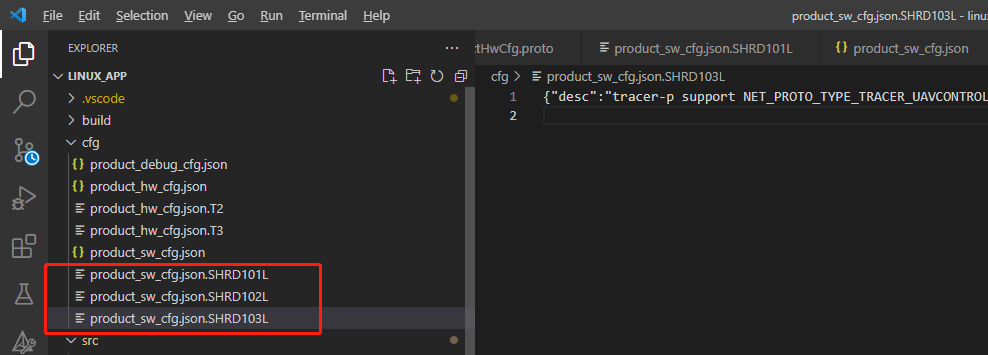


netProtoType取值枚举为下面NET\_PROTO\_TYPER的枚举定义



### 支持NET\_PROTO\_TYPE\_TRACER\_UAVCONTROLLER的版本号定义为SHRD103L

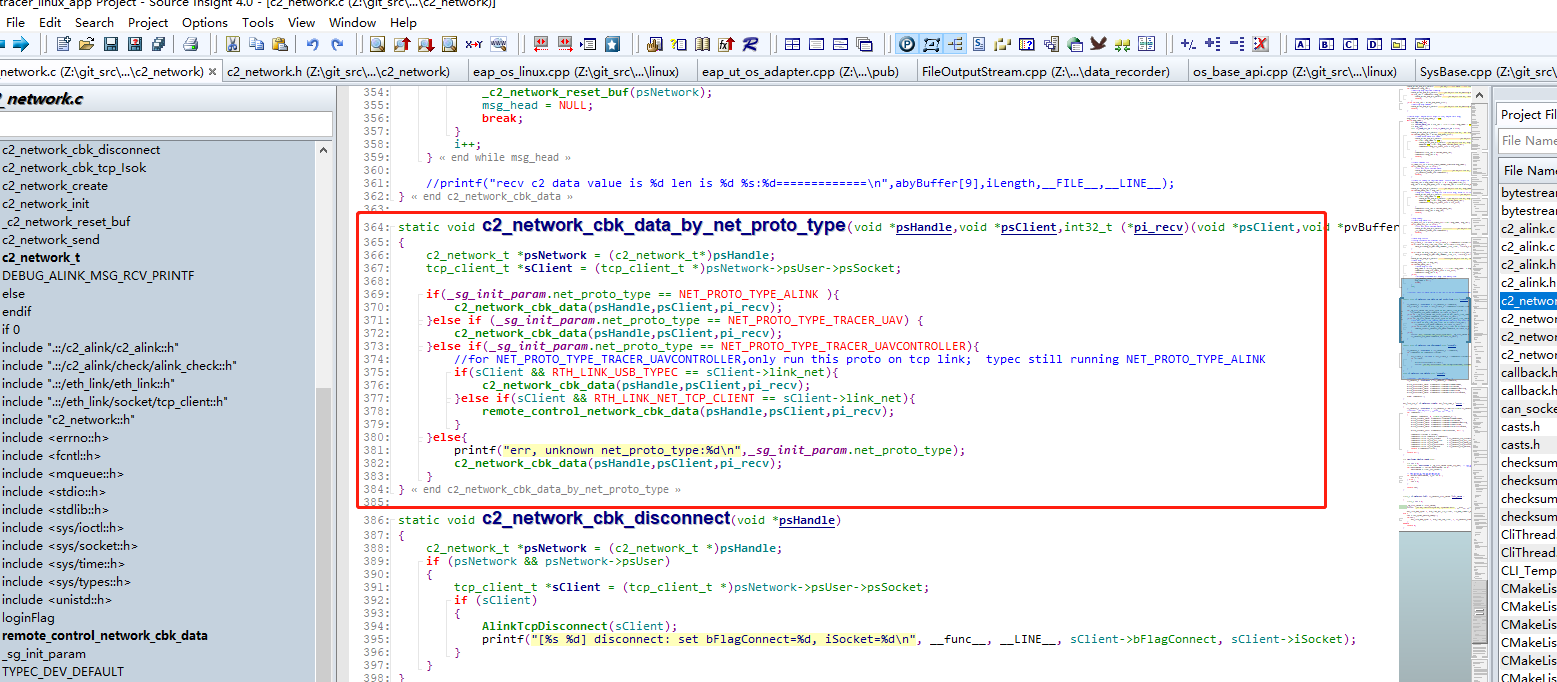




### 协议接收处理函数

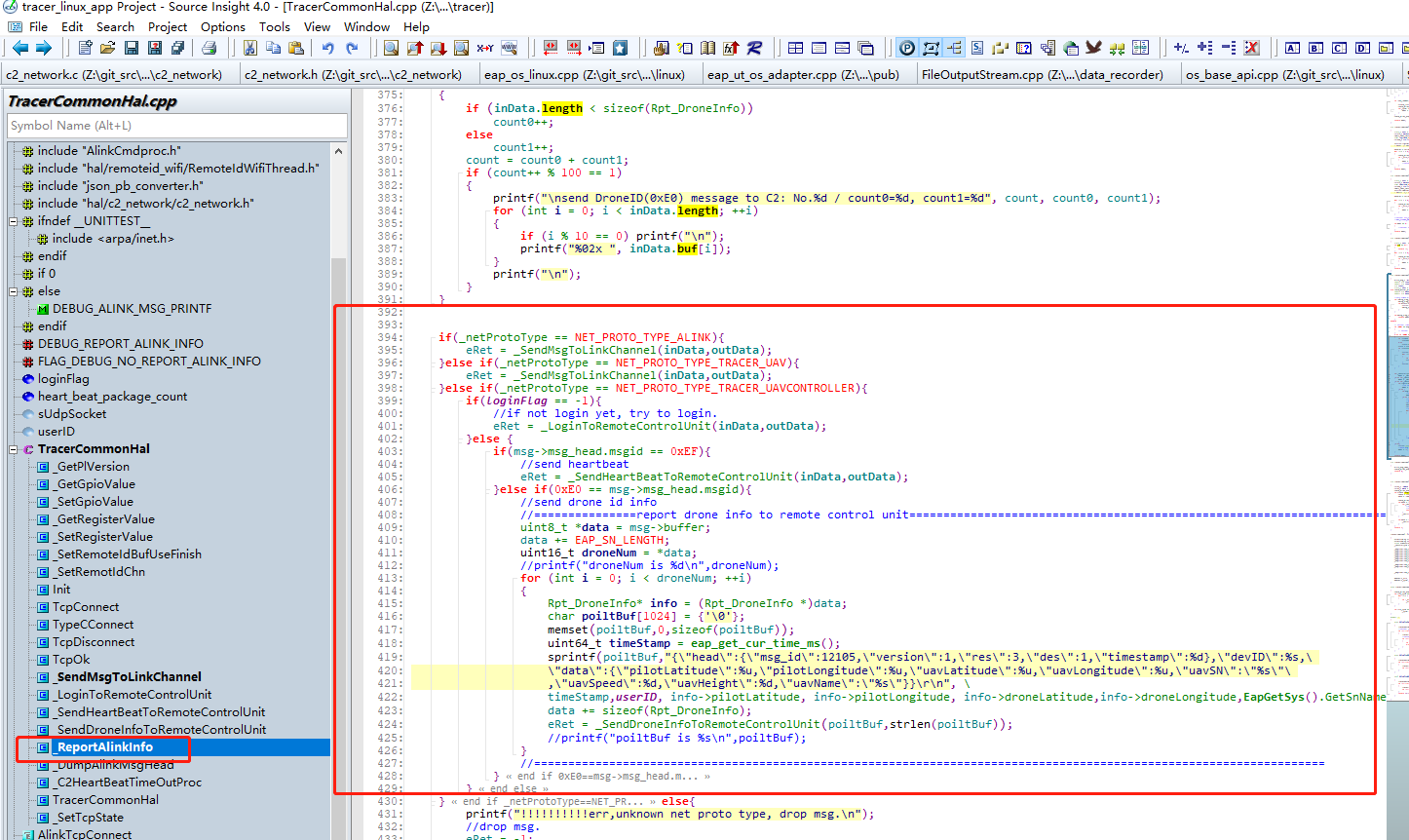
c2\_network\_cbk\_data\_by\_net\_proto\_type

根据不同的协议，作不同的消息收包处理。



### 协议发送处理

内部仍以alink协议为主，在上报数据时，将alink协议，根据当前组网通讯协议，进行消息格式转换。



# 问题

## Protoc工具版本不对，导致编译失败

In file included from /home/a23286/git\_src/shrd100\_ps\_linux\_branch\_hua/linux\_app/src/eap/alink\_pb/MavAlink.pb.cc:4:

/home/a23286/git\_src/shrd100\_ps\_linux\_branch\_hua/linux\_app/src/eap/alink\_pb/MavAlink.pb.h:13:2: error: #error "This file was generated by a newer version of protoc which is"

13 | #error "This file was generated by a newer version of protoc which is"

### 解决办法：

使用petalinux交叉工具链提供的protoc. 这样protobuf库的版本和protoc工具的版本才能一致。

不要直接使用 开发主机x86\_64上安装的版本。

Source /opt/tracer/petalinux/2022.2/environment-setup-cortexa72-cortexa53-xilinx-linux

a23286@build$which protoc

/opt/tracer/petalinux/2022.2/sysroots/x86\_64-petalinux-linux/usr/bin/protoc

# 结束