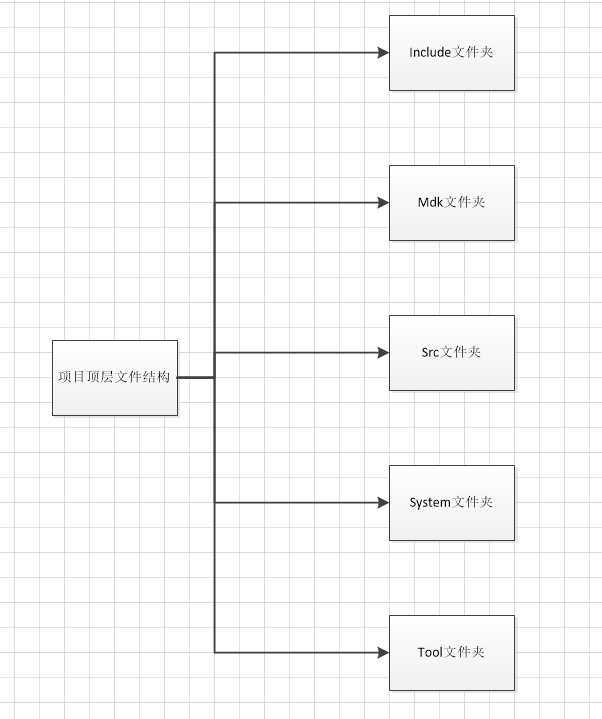
交换机程序结构

吉富逵

2020/04/02

# 概述

交换机的程序结构分为include、mdk、src、systems、tools五个部分。



项目顶层文件结构

include部分为整个项目的本地头文件。

mdk部分为项目中PHY的驱动程序。

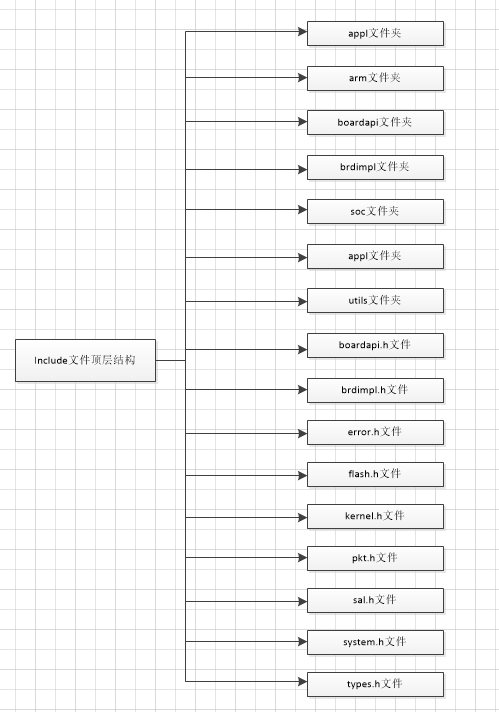
src部分为整个项目中上层功能的实现。

system部分为项目中BSP部分的实现根据不同的芯片进行实现。

tools部分为整个项目中的工具包括将引导文件和固件程序文件合并成一个文件，将HTML文件转换为C程序文件。

# include部分

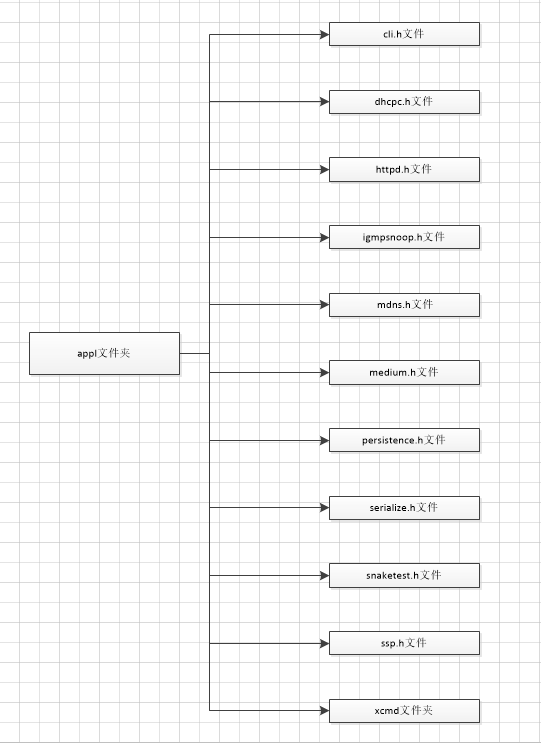
include文件部分包含真个项目中主要的头文件。



include文件结构

## appl文件夹

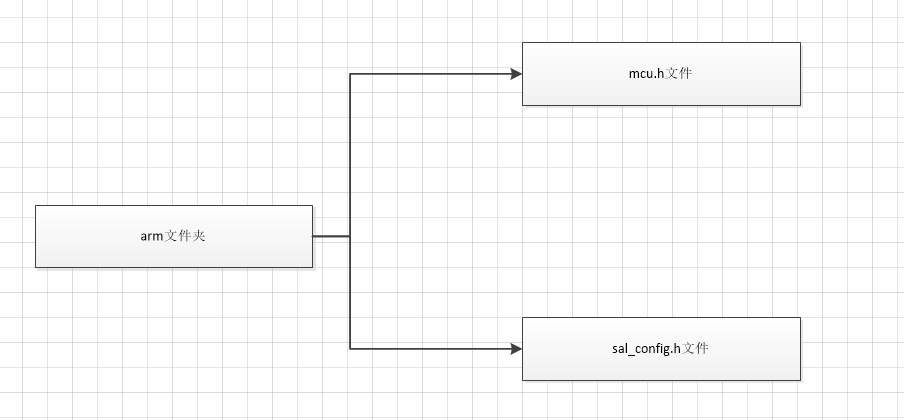
appl文件夹下面主要是上层应用程序的头文件。



appl文件夹结构

## arm文件夹

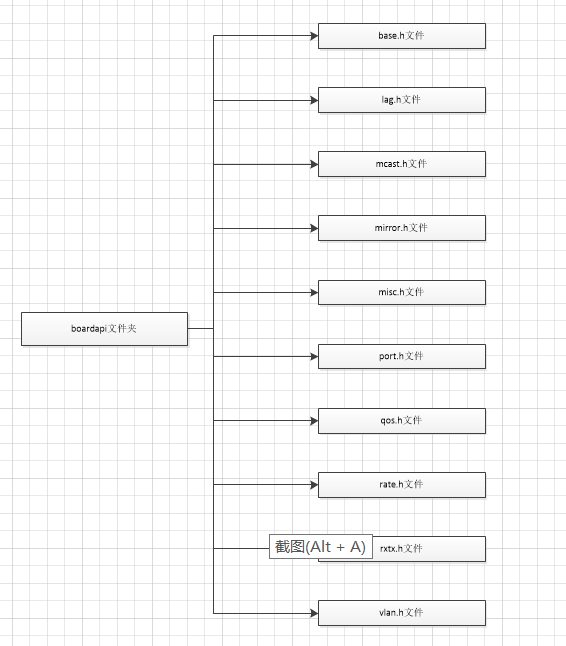
arm文件夹下的是



arm文件夹结构

## boardapi文件夹

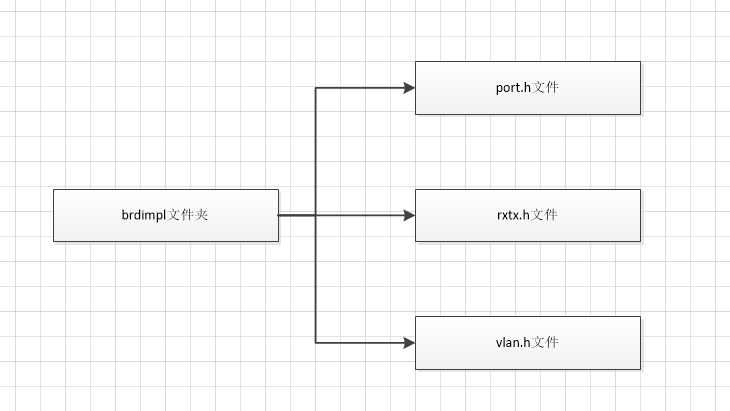
boardapi文件夹



boardapi文件夹结构

## brdimpl文件夹

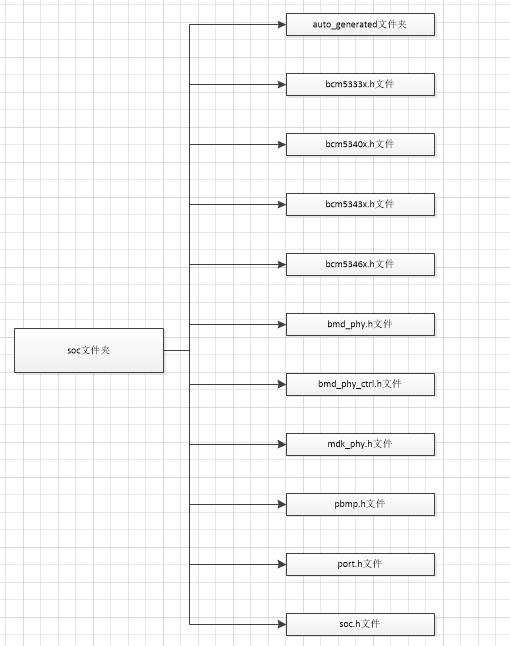
brdimpl文件夹



brdimpl文件夹

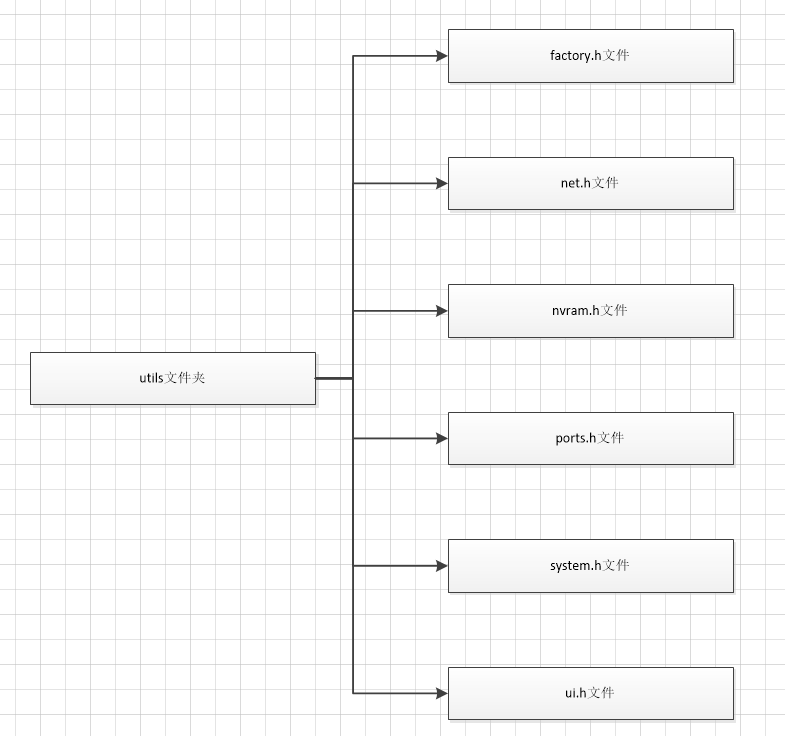
## soc文件夹

soc文件夹



## utils文件夹

utils文件夹



## 2.7 boardapi.h文件

此文件作为引入设备功能头文件的总的头文件。

## 2.8 brdimpl.h文件

此文件作为引入板卡功能头文件的总的头文件。

## 2.9 error.h文件

此文件定义设备错误类型。

## 2.10 flash.h文件

此文件定义操作flash的功能和对应的flash数据结构。

## 2.11 kernel.h文件

此文件声明内核任务相关的函数和任务相关的数据结构。

## 2.12 pkt.h文件

此文件声明数据包相关的数据结构和宏。

## 2.13 sal.h文件

此文件声明系统相关的函数。

## 2.14 system.h文件

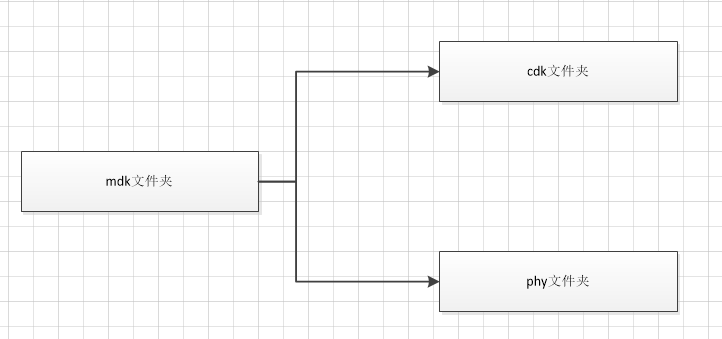
此文件是引入系统相关的头文件的总的头文件。

## 2.15 types.h文件

此文件定义文件的数据类型别名和一些常见的宏。

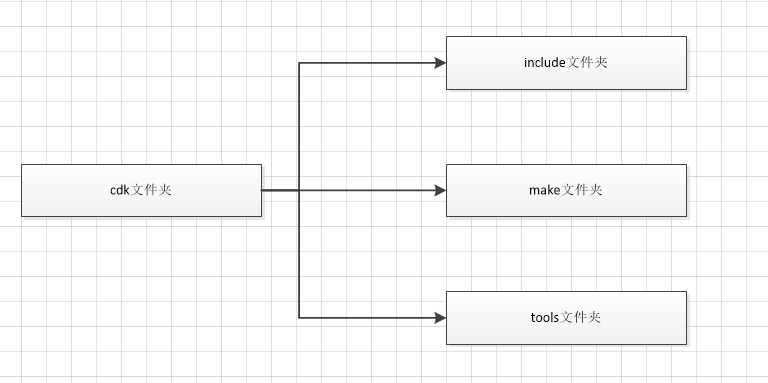
# mdk部分

mdk部分的主要文件为phy部分，这个部分是用于控制PHY芯片,实现PHY功能。



## 3.1cdk文件夹

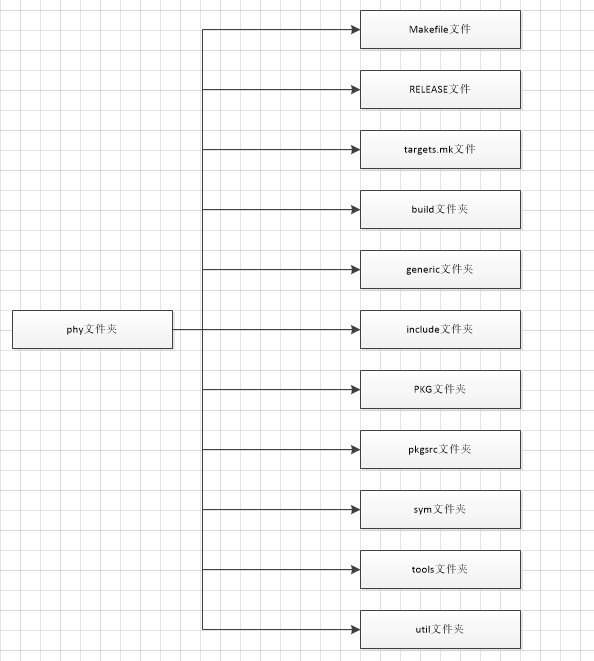
cdk文件夹



cdk文件夹结构

## 3.2phy文件夹

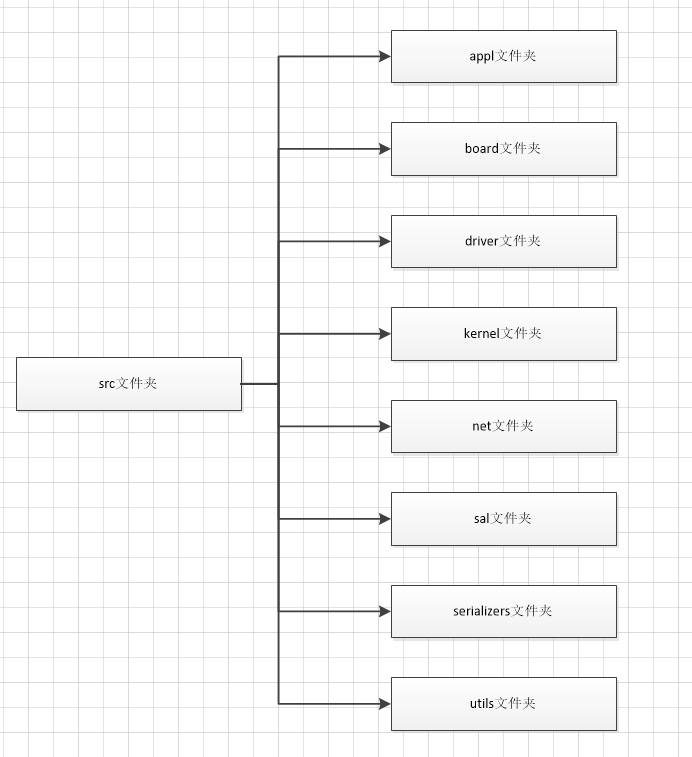
phy文件夹是关于phy的相关驱动。



phy文件夹结构

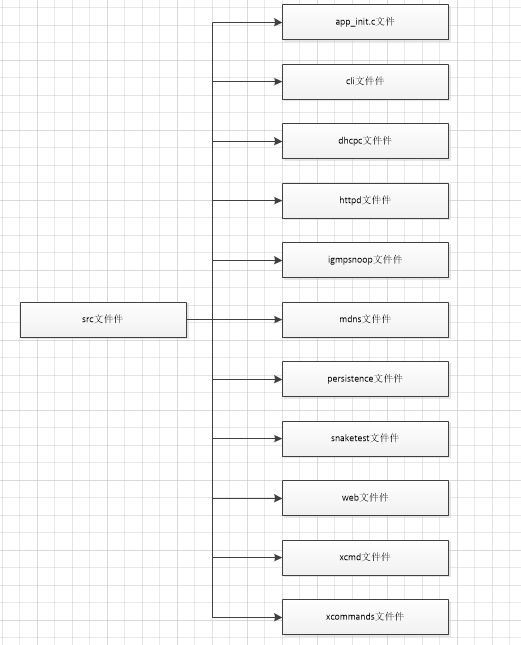
# src部分

src部分是程序中上层应用的实现。



## 4.1appl文件夹

appl文件夹用于

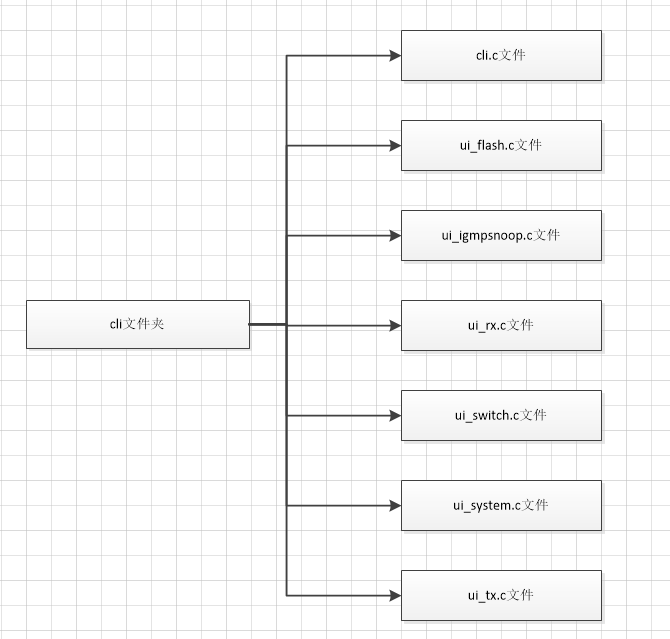


appl文件夹结构图

### 4.1.1app\_init.c文件

### 4.1.2cli文件夹

cli文件夹是对于人机交互的实现。



cli文件夹结构

#### 4.1.2.1cli.c文件

cli.c文件是整个人机交互实现的顶层文件。包含了任务队列的初始化，任务队列的添加，任务队列的删除，添加基本任务，以及对用户输入的获取。

#### 4.1.2.2ui\_flash.c文件

用户对于flash的操作的实现。

#### 4.1.2.3ui\_igmpsnoop.c文件

#### 4.1.2.4ui\_rx.c文件

#### 4.1.2.5ui\_switch.c文件

#### 4.1.2.6ui\_system.c文件

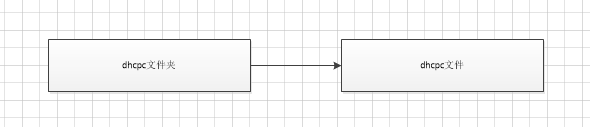
ui\_system.c文件是对系统信息的获取和设置。包括：

1. 获取系统tick值和每tick消耗的微秒数。
2. 获取内存数据
3. 设置内存数据
4. 访问控制
5. flash固件程序信息
6. 复位button

#### 4.1.2.7ui\_tx.c文件

### 4.1.3dhcpc文件夹

dhcpc文件夹

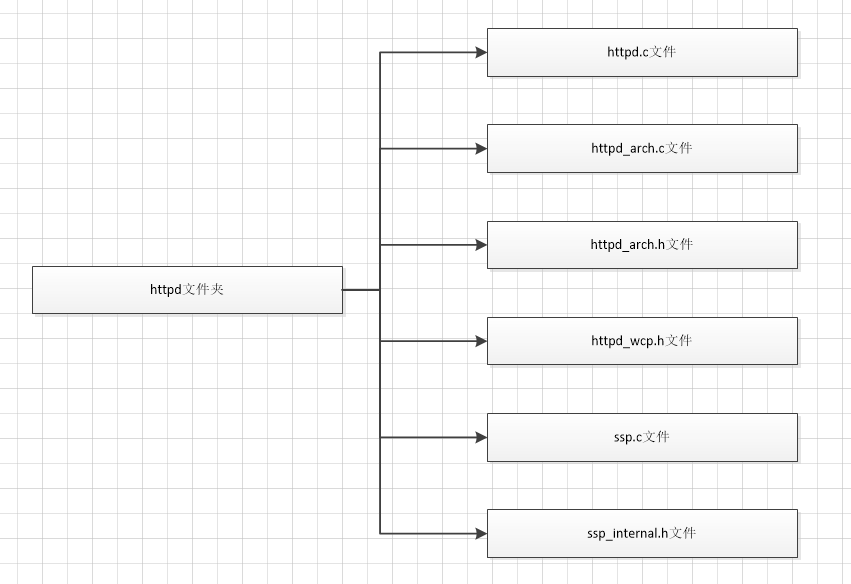


dhcpc文件夹

#### 4.1.3.1dhcpc文件

### 4.1.4httpd文件夹

httpd文件夹



httpd文件夹

#### 4.1.4.1httpd.c文件

#### 4.1.4.2httpd\_arch.c文件

#### 4.1.4.3httpd\_arch.h文件

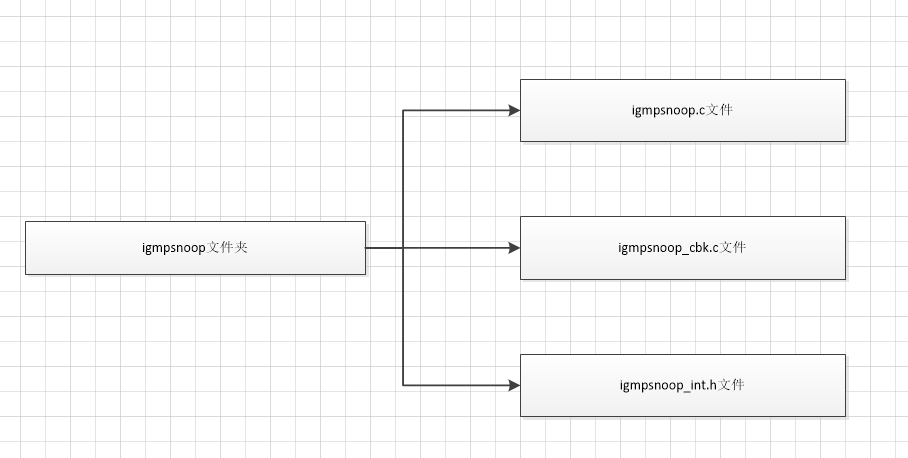
#### 4.1.4.4httpd\_wcp.h文件

#### 4.1.4.5ssp.c文件

#### 4.1.4.6ssp\_internal.h文件

### 4.1.5igmpsnoop文件夹

igmpsnoop文件夹主要是关于igmp版本2的支持。



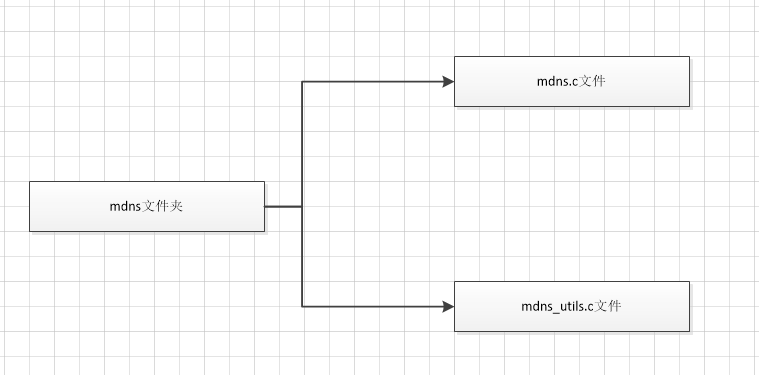
#### 4.1.5.1igmpsnoop.c文件

#### 4.1.5.2igmpsnoop\_cbk.c文件

#### 4.1.5.3igmpsnoop\_int.h文件

### 4.1.6mdns文件夹

mdns文件



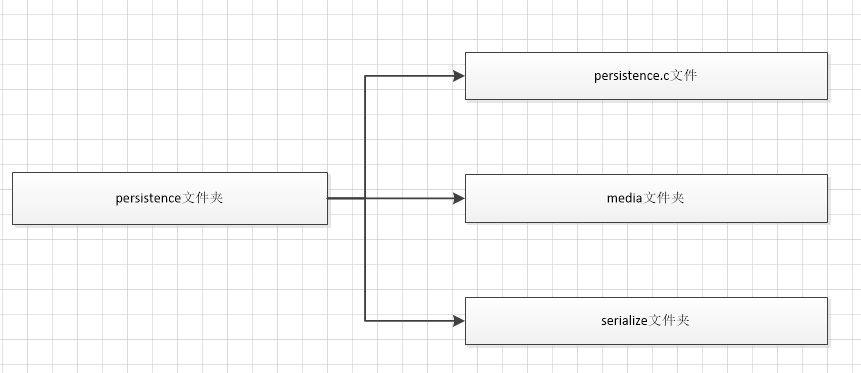
mdns文件结构

#### 4.1.6.1mdns.c文件

#### 4.1.6.2mdns\_utils.c文件

### 4.1.7persistence文件夹

persistence文件夹



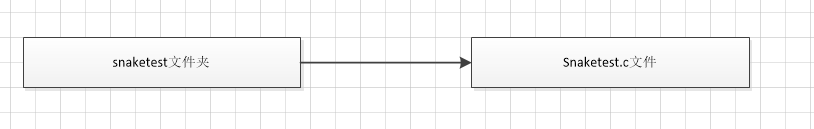
#### 4.1.7.1persistence.c文件

#### 4.1.7.2media文件夹

#### 4.1.7.3serialize文件夹

### 4.1.8snaketest文件夹

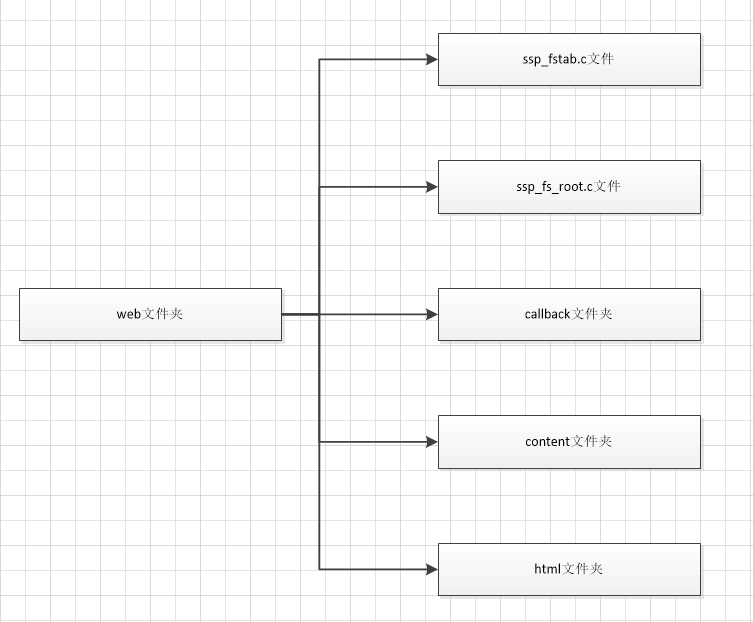
snaketest文件夹



#### 4.1.8.1snaketest.c文件

### 4.1.9web文件夹

web文件夹是



web文件夹

#### 4.1.9.1ssp\_fstab.c文件

#### 4.1.9.2ssp\_fs\_root.c文件

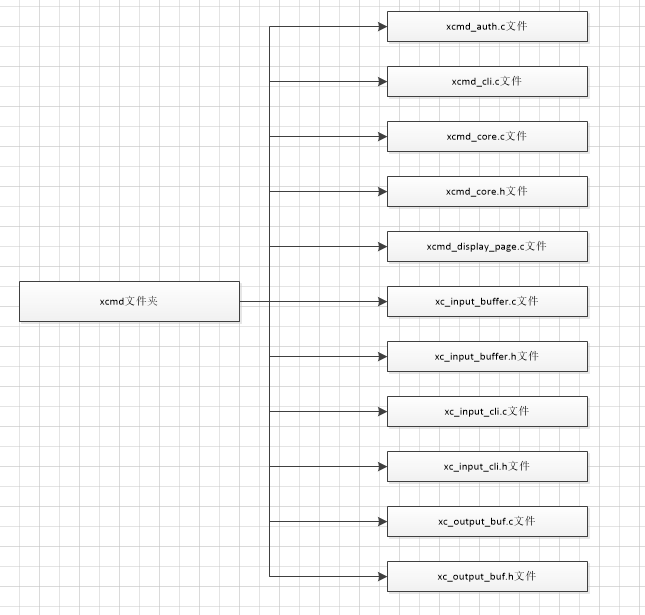
#### 4.1.9.3callback文件夹

#### 4.1.9.4content文件夹

#### 4.1.9.5html文件夹

### 4.1.10xcmd文件夹

xcmd文件



#### 4.1.10.1xcmd\_auth.c文件

#### 4.1.10.2xcmd\_cli.c文件

#### 4.1.10.3xcmd\_core.c文件

#### 4.1.10.4xcmd\_core.h文件

#### 4.1.10.5xcmd\_display\_page.c文件

#### 4.1.10.6xc\_input\_buffer.c文件

#### 4.1.10.7xc\_input\_buffer.h文件

#### 4.1.10.8xc\_input\_cli.c文件

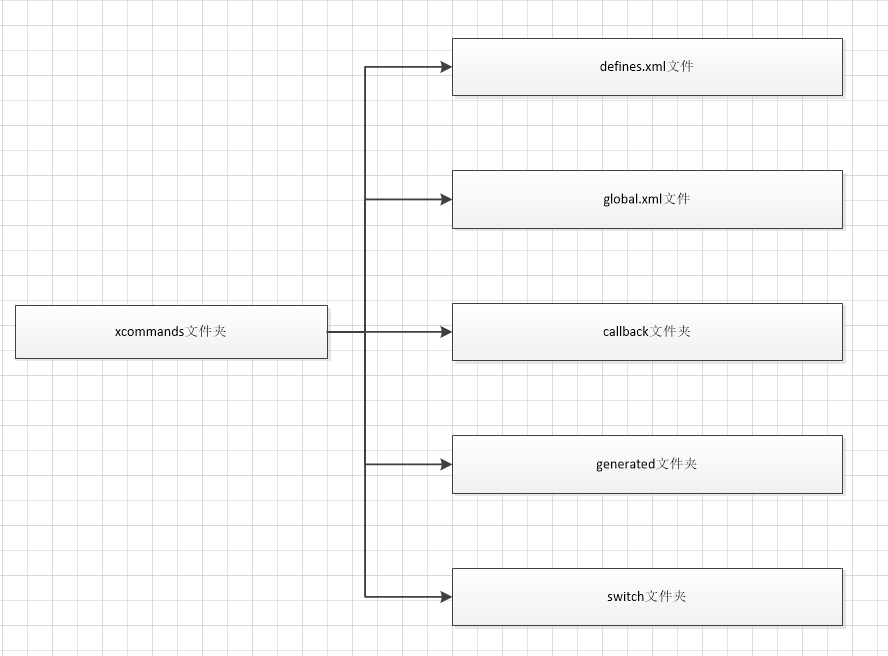
#### 4.1.10.9xc\_input\_cli.h文件

#### 4.1.10.10xc\_output\_buf.c文件

#### 4.1.10.11xc\_output\_buf.h文件

### 4.1.11xcommands文件夹

xcommands文件夹



#### 4.1.11.1defines.xml文件

#### 4.1.11.2global.xml文件

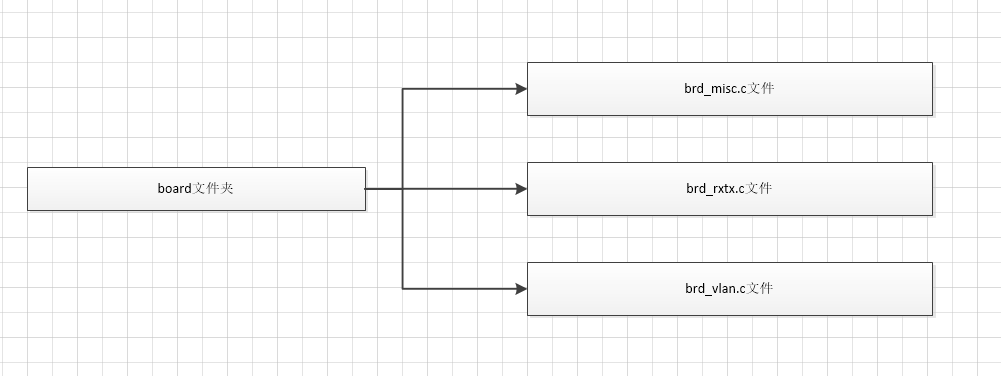
#### 4.1.11.3callback文件夹

#### 4.1.11.4generated文件夹

#### 4.1.11.5switch文件夹

## 4.2board文件夹

board文件夹



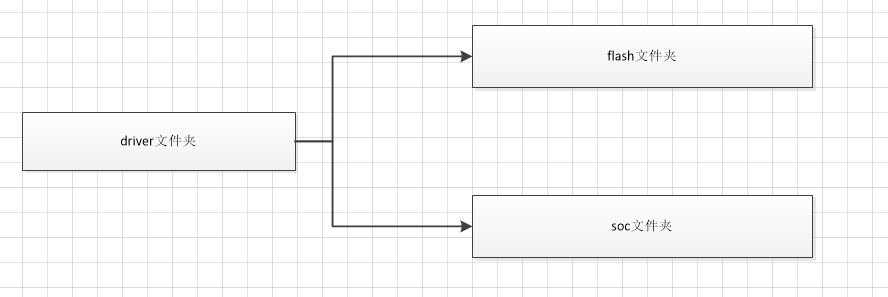
### 4.2.1brd\_misc.c文件

### 4.2.2brd\_rxtx.c文件

### 4.2.3brd\_vlan.c文件

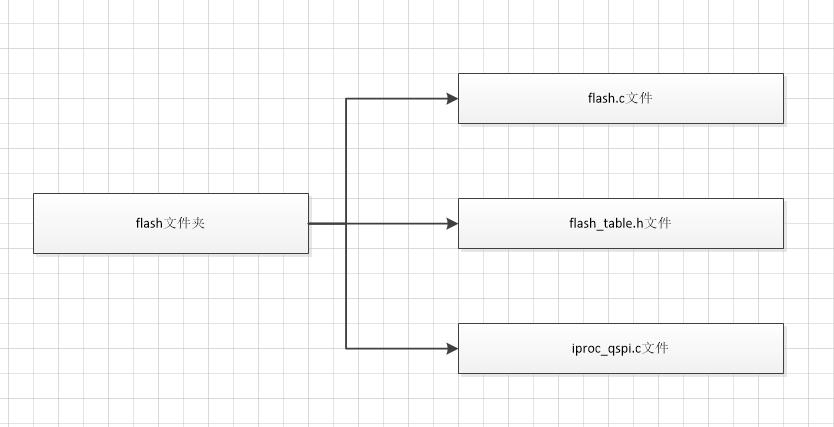
## 4.3driver文件夹

driver文件。



### 4.3.1flash文件夹

flash文件夹。



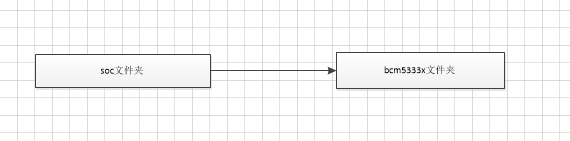
#### 4.3.1.1flash.c文件

#### 4.3.1.2flash\_table.h文件

#### 4.3.1.3iproc\_qspi.c文件

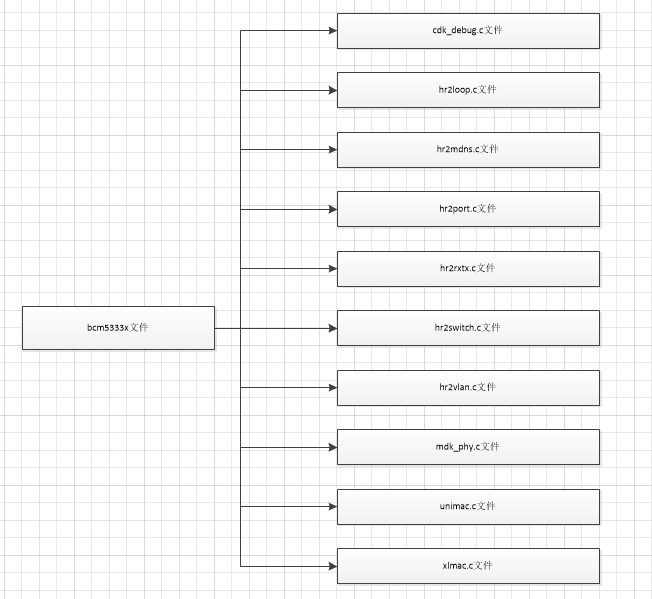
### 4.3.2soc文件夹

soc文件夹



#### 4.3.2.1bcm5333x文件夹

bcm5333x文件夹



##### 4.3.2.1.1cdk\_debug.c文件

##### 4.3.2.1.2hr2loop.c文件

#### 4.3.2.1.3hr2mdns.c文件

##### 4.3.2.1.4hr2port.c文件

##### 4.3.2.1.5hr2rxtx.c文件

设备级的发送接收数据包的实现。

##### 4.3.2.1.6hr2switch.c文件

##### 4.3.2.1.7hr2vlan.c文件

此文件是关于VLAN的底层实现。

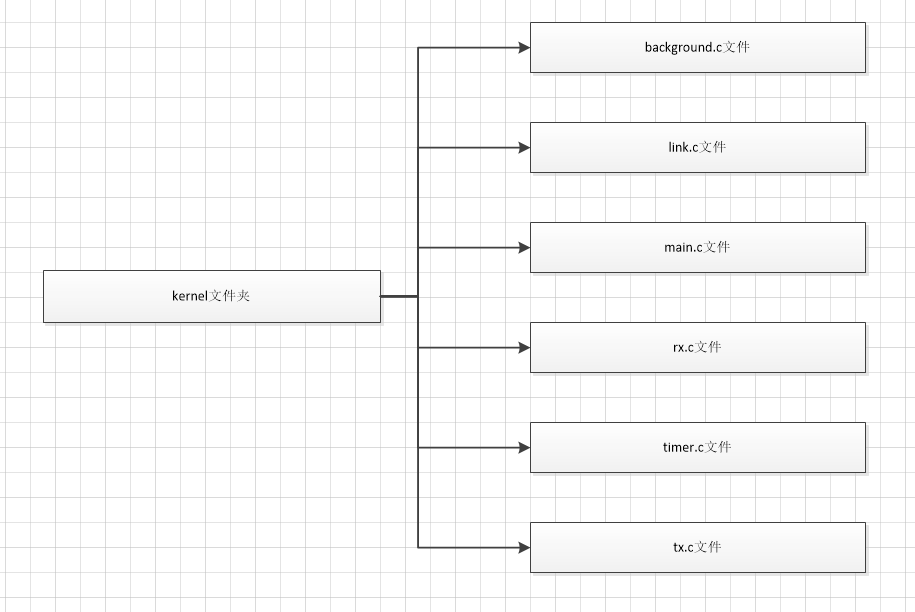
##### 4.3.2.1.8mdk\_phy.c文件

##### 4.3.2.1.9unimac.c文件

##### 4.3.2.1.10xlmac.c文件

## 4.4kernel文件夹

kernel文件夹

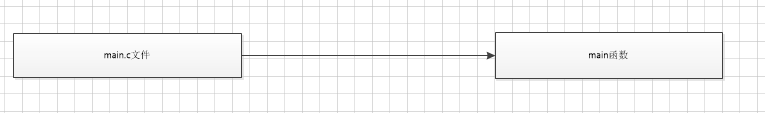


### 4.4.1background.c文件

### 4.4.2link.c文件

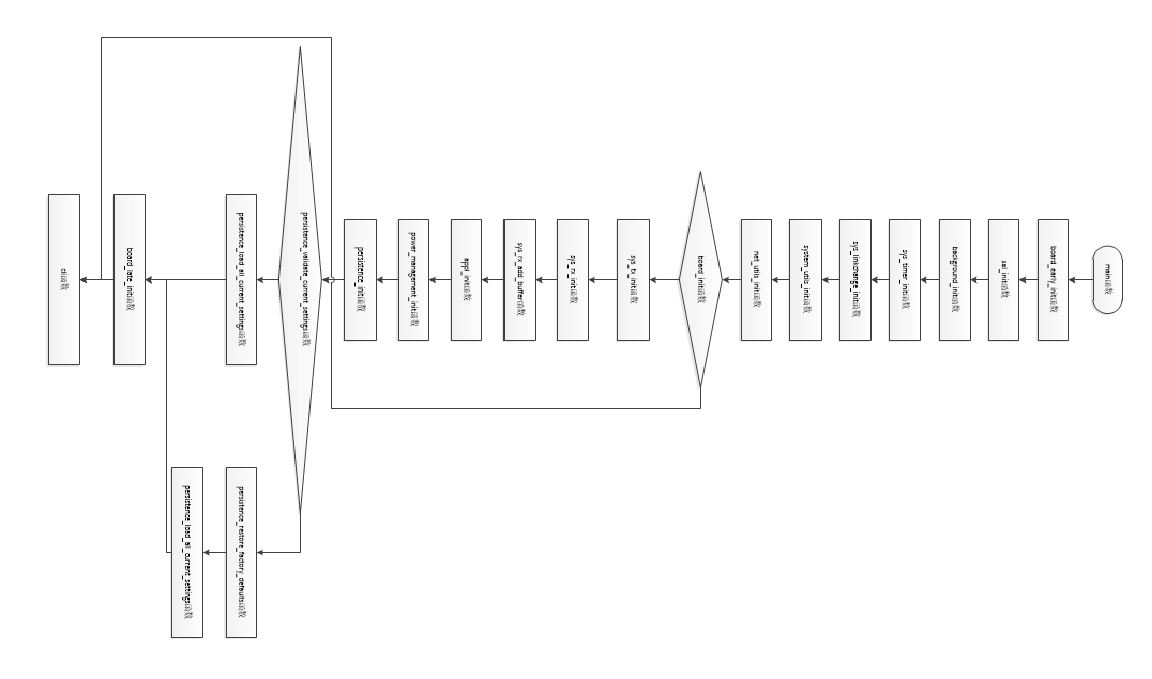
### 4.4.3main.c文件

main.c文件定义了main函数，main函数作为整个项目的起始函数。



#### 4.4.3.1main函数

main函数最为整个项目的起始函数。



main函数的流程图

##### 4.4.3.1.1board\_early\_init函数

board\_early\_init函数在BCM5333x\systems\bcm95333x\src\board\_init.c文件中。[[1]](#footnote-1)

##### 4.4.3.1.2sal\_init函数

sal\_init函数

##### 4.4.3.1.3background\_init函数

background\_init函数

##### 4.4.3.1.4sys\_timer\_init函数

sys\_timer\_init函数

##### 4.4.3.1.5sys\_linkchange\_init函数

sys\_linkchange\_init函数

##### 4.4.3.1.6system\_utils\_init函数

system\_utils\_init函数

##### 4.4.3.1.7net\_utils\_init函数

net\_utils\_init函数

##### 4.4.3.1.8board\_init函数

board\_init函数

##### 4.4.3.1.9sys\_tx\_init函数

sys\_tx\_init函数

##### 4.4.3.1.10sys\_rx\_init函数

sys\_rx\_init函数

##### 4.4.3.1.11sys\_rx\_add\_buffer函数

sys\_rx\_add\_buffer函数

##### 4.4.3.1.12appl\_init函数

appl\_init函数

##### 4.4.3.1.13power\_management\_init函数

power\_management\_init函数

##### 4.4.3.1.14persistence\_init函数

persistence\_init函数

##### 4.4.3.1.15persistence\_validate\_current\_settings函数

persistence\_validate\_current\_settings函数

##### 4.4.3.1.16persistence\_load\_all\_current\_settings函数

persistence\_load\_all\_current\_settings函数

##### 4.4.3.1.17persistence\_restore\_factory\_defaults函数

persistence\_restore\_factory\_defaults函数

##### 4.4.3.1.18persistence\_save\_all\_current\_settings函数

persistence\_save\_all\_current\_settings函数

##### 4.4.3.1.19board\_late\_init函数

board\_late\_init函数

##### 4.4.3.1.20cli函数

cli函数

### 4.4.4rx.c文件

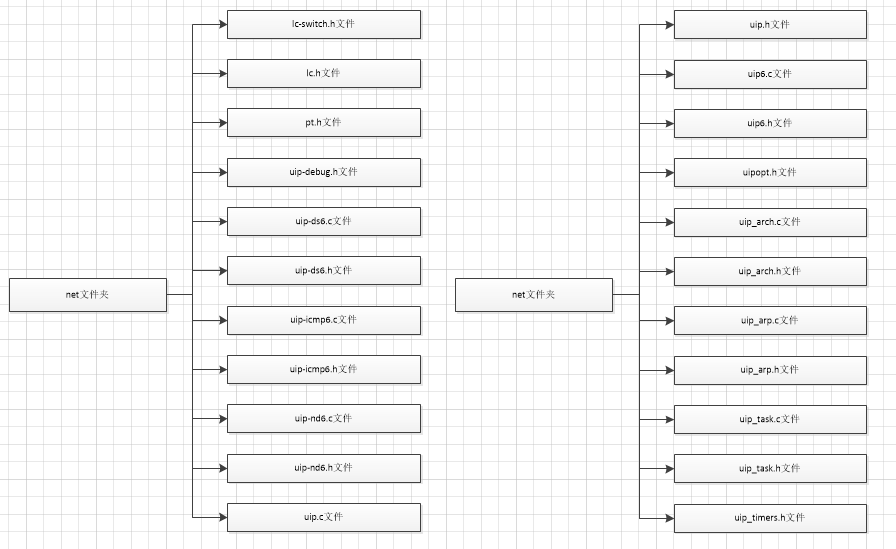
### 4.4.5timer.c文件

### 4.4.6tx.c文件

此文件是关于发送数据的顶层的实现。

## 4.5net文件夹

net文件夹



### 4.5.1lc-switch.h文件

### 4.5.2lc.h文件

### 4.5.3pt.h文件

### 4.5.4uip-debug.h文件

### 4.5.5uip-ds6.c文件

### 4.5.6uip-ds6.h文件

### 4.5.7uip-icmp6.c文件

### 4.5.8uip-icmp6.h文件

### 4.5.9uip-nd6.c文件

### 4.5.10uip-nd6.h文件

### 4.5.11uip.c文件

此文件是微型的TCP\IP协议栈的实现。

### 4.5.12uip.h文件

### 4.5.13uip6.c文件

### 4.5.14uip6.h文件

### 4.5.15uipopt.h文件

### 4.5.16uip\_arch.c文件

此文件主要是冗余值的计算。

### 4.5.17uip\_arch.h文件

### 4.5.18uip\_arp.c文件

此文件主要是用于对ARP协议的处理

### 4.5.19uip\_arp.h文件

### 4.5.20uip\_task.c文件

### 4.5.21uip\_task.h文件

### 4.5.22uip\_timers.h文件

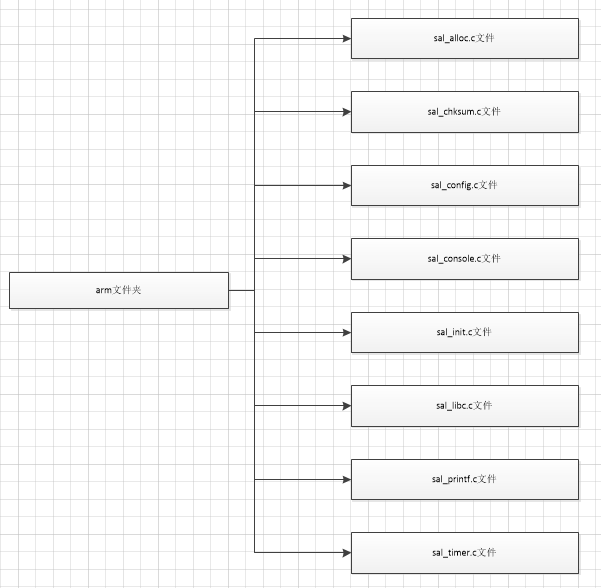
## 4.6sal文件夹

sal文件夹



### 4.6.1arm文件夹

arm文件夹



#### 4.6.1.1sal\_alloc.c文件

#### 4.6.1.2sal\_chksum.c文件

此文件主要是用于计算IP数据包的冗余值。

#### 4.6.1.3sal\_config.c文件

#### 4.6.1.4sal\_console.c文件

#### 4.6.1.5sal\_init.c文件

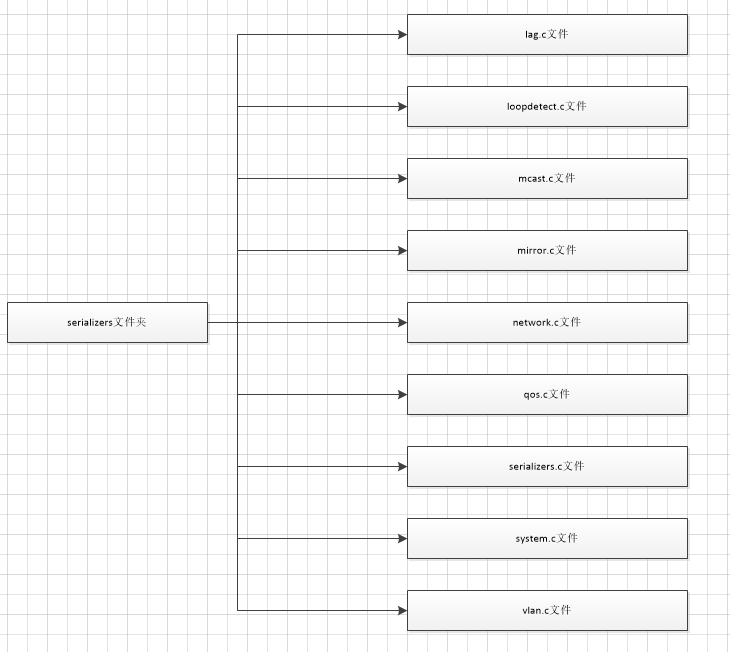
#### 4.6.1.6sal\_libc.c文件

#### 4.6.1.7sal\_printf.c文件

#### 4.6.1.8sal\_timer.c文件

## 4.7serializers文件夹

serializers文件夹



### 4.7.1lag.c文件

### 4.7.2loopdetect.c文件

### 4.7.3mcast.c文件

### 4.7.4mirror.c文件

### 4.7.5network.c文件

### 4.7.6qos.c文件

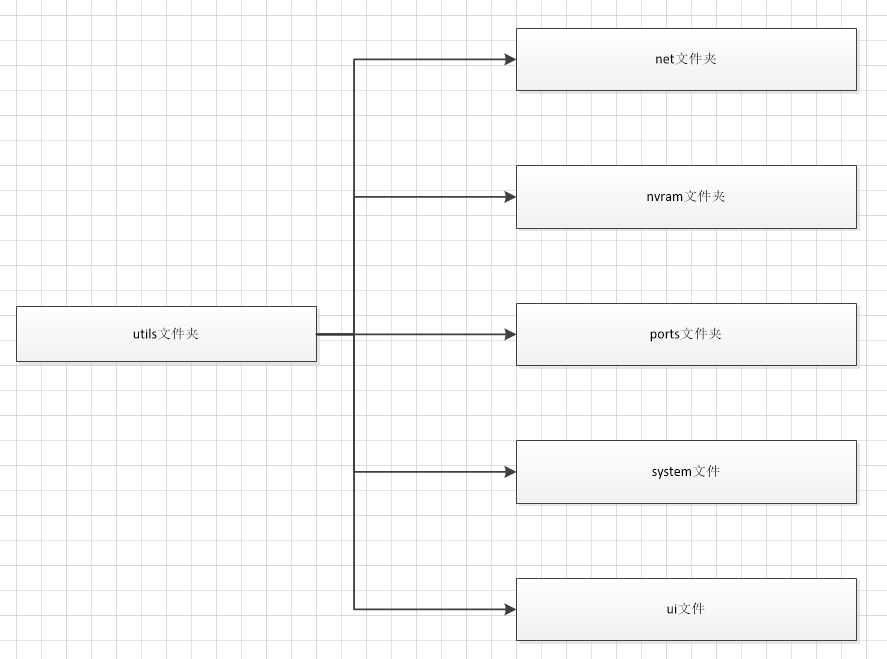
### 4.7.7serializers.c文件

### 4.7.8system.c文件

### 4.7.9vlan.c文件

## 4.8utils文件夹

utils文件



### 4.8.1net文件夹

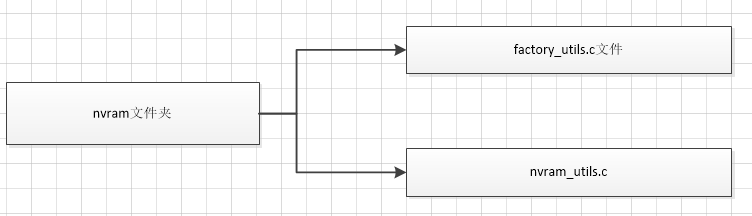
net文件夹



#### 4.8.1.1net\_utils.c文件

### 4.8.2nvram文件夹

nvram文件件

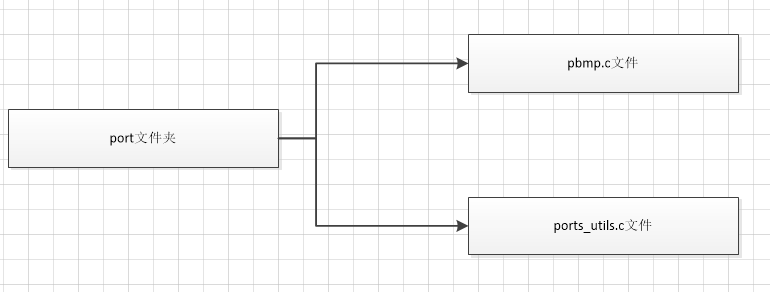


#### 4.8.2.1factory\_utils.c文件

#### 4.8.2.2nvram\_utils.c文件

### 4.8.3ports文件夹

port文件夹



#### 4.8.3.1pbmp.c文件

#### 4.8.3.2ports\_utils.c文件

### 4.8.4system文件夹

system文件夹



#### 4.8.4.1system\_utils.c文件

### 4.8.5ui文件

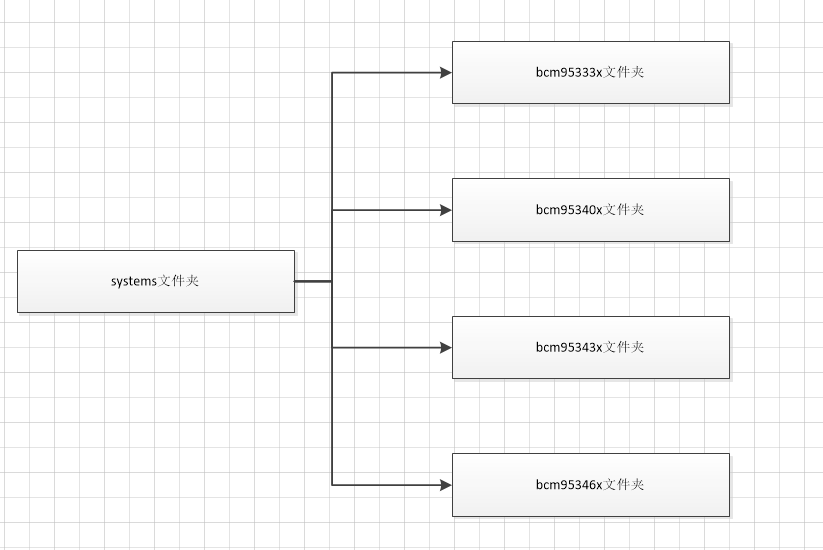
ui文件



#### 4.8.5.1ui\_utils.c文件

# system部分

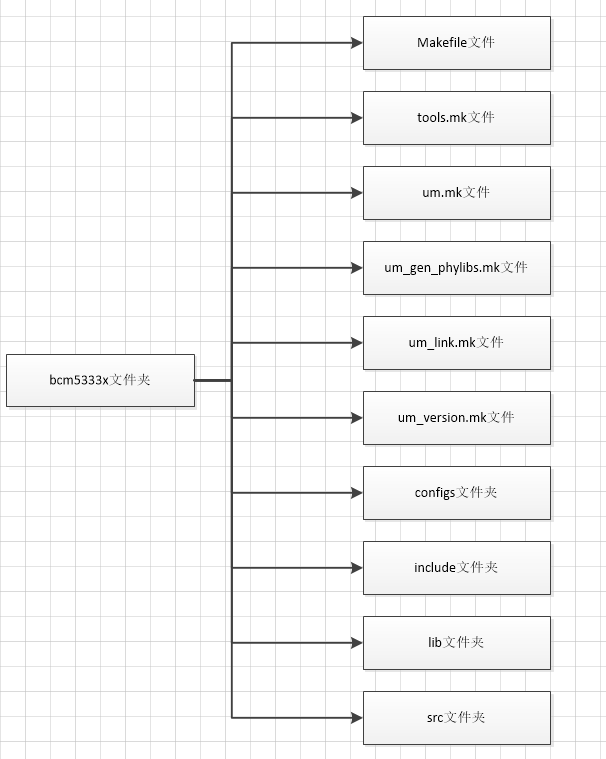
systems文件夹主要是针对不同的芯片



systems文件夹结构

## 5.1bcm5333x文件夹

bcm5333x是针对于bcm5333x芯片进行。



bcm5333x文件夹结构

## 5.1Makefile文件

此文件是整个bcm5333x项目的最顶层的Makefile文件。

## 5.2tools.mk文件

## 5.3um.mk文件

## 5.4um\_gen\_phylibs.mk文件

## 5.5um\_link.mk文件

## 5.6um\_version.mk文件

此文件用于定义设备的程序版本号。

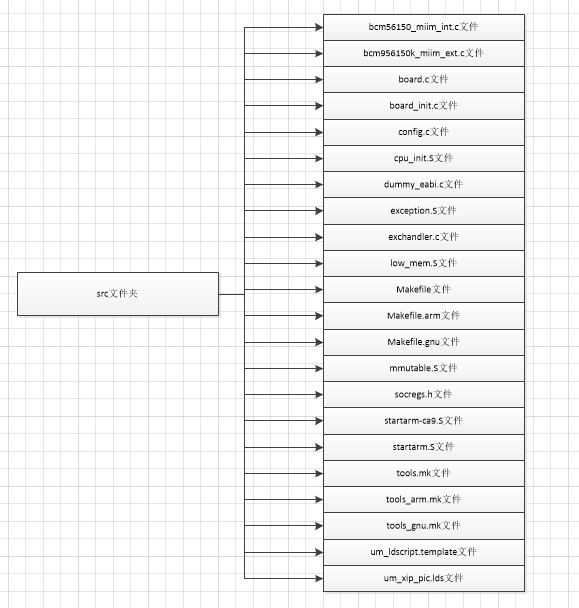
## 5.7configs文件夹

## 5.8include文件夹

## 5.9lib文件夹

## 5.10src文件夹

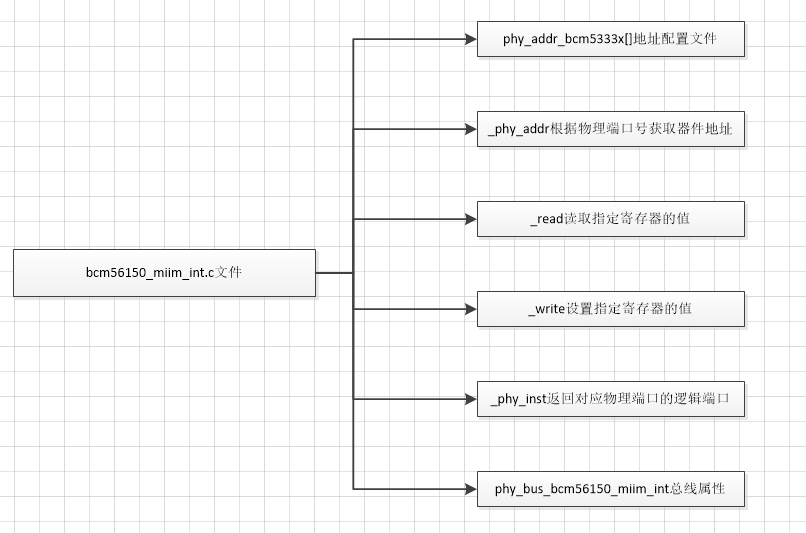
src文件



src文件夹

### 5.10.1bcm56150\_miim\_int.c文件

此文件是主交换芯片的mii管理接口相关功能的实现。



bcm56150\_miim\_int.c文件结构

（1）\_phy\_addr\_bcm5333x数组定义的是bcm5333x设备配置各个物理端口所对应的PHY的总线地址。

（2）\_phy\_addr函数传入物理地址返回端口PHY的控制地址。

（3）\_read函数用于读取寄存器的值。

（4）\_write函数用于设置寄存器的值。

（5）\_phy\_inst函数用于返回设备物理端口号对应的逻辑端口号。

（6）phy\_bus\_bcm56150\_miim\_int定义这个芯片的总线相关的属性。

### 5.10.2bcm956150k\_miim\_ext.c文件

此文件是用于配置扩展PHY的mii控制接口。

（1）\_phy\_addr函数传入物理地址返回端口PHY的控制地址。

（2）\_read函数用于读取寄存器的值。

（3）\_write函数用于设置寄存器的值。

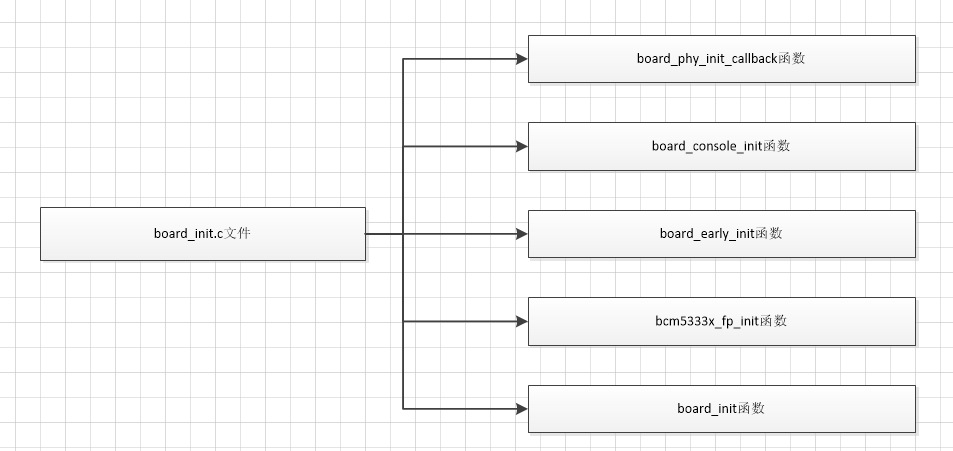
（4）\_phy\_inst函数用于返回设备物理端口号对应的逻辑端口号。

（5）phy\_bus\_bcm956150k\_miim\_ext定义这个芯片的外置PHY总线的相关属性。

### 5.10.3board.c文件

### 5.10.4board\_init.c文件

board\_init.c文件



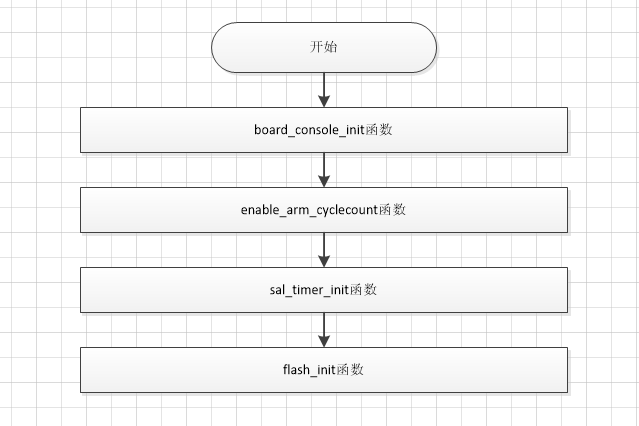
#### 5.10.4.1board\_phy\_init\_callback函数

#### 5.10.4.2board\_console\_init函数

board\_console\_init函数是用于进行串口的初始化，根据传入的波特率和时钟频率。设置串口的相关参数。

#### 5.10.4.3board\_early\_init函数

board\_early\_init函数



#### 5.10.4.4bcm5333x\_fp\_init函数

#### 5.10.4.5board\_init函数

### 5.10.5config.c文件

### 5.10.6cpu\_init.S文件

### 5.10.7dummy\_eabi.c文件

### 5.10.8exception.S文件

### 5.10.9exchandler.c文件

### 5.10.10low\_mem.S文件

### 5.10.11Makefile文件

### 5.10.12Makefile.arm文件

### 5.10.13Makefile.gnu文件

### 5.10.14mmutable.S文件

### 5.10.15socregs.h文件

### 5.10.16startarm-ca9.S文件

### 5.10.17startarm.S文件

### 5.10.18tools.mk文件

### 5.10.19tools\_arm.mk文件

### 5.10.20tools\_gnu.mk文件

### 5.10.21um\_ldscript.template文件

### 5.10.22um\_xip\_pic.lds文件

# 常用寄存器和调试

## 6.1PHY IEEE标准寄存器

### 6.1.1寄存器0

用于控制PHY。

bit15:1设置PHY复位。

bit14:1 设置内部PHY环回。

bit[6,13]:设置手动设置PHY的模式。

bit12:1设置使用电口的自协商模式。

bit11:1设置关闭电口的电。

bit9:1重新进行电口的自协商。

bit8:1使能双工模式。

### 6.1.2寄存器1

获取PHY的状态。

bit15:1具有100BASE-4T的能力。

bit14:1具有100BASE-X的双工能力。

bit13:1具有100BASE-X的半双工能力。

bit12:1 具有10BASE-X的双工能力。

bit11:1 具有10BASE-X的半双工能力。

bit10:1 具有100BASE-2T的全双工能力。

bit9:1 具有100BASE-2T的半双工能力。

bit8:1 具有扩展信息。

bit5:1 自协商完成。

bit4:1 检测到远端故障。

bit3:1 具有自协商能力。

bit2:1 端口link。

bit1:1 具有毛刺检测。

bit0:1 具有扩展支持表。

### 6.1.3寄存器2

PHY OUI的高位。

### 6.1.4寄存器3

PHY OUI的低位。

### 6.1.5寄存器4

自协商广播寄存器

bit15:1下一页支持。

bit12:1发送远端故障指示。

bit[8,7]:暂停的支持。

bit6:1 广播半双工。

bit5:1 广播全双工。

### 6.1.6寄存器5

获取对端PHY的能力。

bit15:1具有下一页能力。

bit14:1 对端PHY接收到link码。

bit13:1 对端检测到link失败。

bit11:1 对端端口支持非对称暂停。

bit10:1 对端端口支持暂停。

bit9:1 对端端口具有100BASE-4T的能力。

bit8:1 对端端口具有100BASE-T全双工的能力。

bit7:1对端端口具有100BASE-T半双工的能力。

bit6: 1对端端口具有10BASE-T全双工的能力。

bit5: 1 对端端口具有10BASE-T半双工的能力。

### 6.1.7寄存器9

设置千兆寄存器。

bit12:1 主从配置使能。

bit11:1 配置PHY作为主设备。

bit10:1设备为连接设备，0设备为终端设备。

bit9:1 具有1000BASE-T全双工能力。

bit8:1 具有1000BASE-T半双工能力。

### 6.1.8寄存器10

获取千兆寄存器状态。

bit15:1 主从配置失败。

bit14:1此PHY作为主设备。

bit13:1本地接收状态良好。

bit12:1 远端接收状态良好。

bit11:1 对端设备支持1000BASE-T全双工。

bit10:1 对端设备支持1000BASE-T半双工。

## 6.2交换机寄存器

### 6.2.1交换机MAC表老化时间

寄存器地址0x02000400，用于定义学习的MAC地址的老化时间。

### 6.2.2交换机二层实例表

寄存器地址0x1c000000，，用于记录学习到的二层信息。

bit103:1 本地更新标志位。

bit102:1 源地址更新标志位。

bit101:1 目的地址更新标志位。

bit99:1 实例是有效。

bit98:1等待软件进行验证。

bit97:1实例是静态的。

bit96:1实例是学习于本地0，学习于远程端口1。

bit95:1按照MAC限制统计实例。

bit94:1源端的消费值覆盖目的端点消费值。

bit93:1基于源地址丢弃数据包。

bit92:1基于目的地址丢弃数据包。

bit91:1发送数据包到CPU。

bit[90:88]:数据包的优先级。

bit86:1监测输出数据。

bit85:1重新映射优先级，

bit[82:78]:端口MAC映射表的索引值。

bit77:1 三层接口的配置。

### 6.2.3端口配置表

寄存器地址0X04000000，用于配置交换机的基本交换规则。

bit56:1当源的module的ID和端口设置的module ID相等不进行丢包。

bit49:1此端口3层禁能IPMC数据包。

bit48:1此端口2层禁能IPMC数据包。

bit47:1VLAN检测优先级基于子网的高于基于端口的。

bit[46:40]:模块的ID。

bit39:1使能输入包的优先级。

bit38:1使能双标签。

bit[37:36]:端口的类型。

bit[35:24]:端口的VLAN的ID。

bit23:1使能基于MAC的VLAN。

bit22:1使能基于子网的VLAN。

bit20:1丢弃所有未Tag的数据包。

bit19:1丢弃所有Tag的数据包。

bit[10:8]:端口的优先级。

bit6:1使能检测。

bit5:1使能输入过滤。

bit2:1 VLAN转换使能。

bit1:1 VLAN转换未命中，丢弃。

bit0:1 使能过滤。

### 6.2.4VLAN配置表

寄存器地址0x14000000，用于获取指定VLAN的ID的配置情况。

bit[124:113]:对于未知的组播发送到相关的VLAN的VLAN的ID。

bit[112:101]:对于未知的单播发送到相关的VLAN的VLAN的ID。

bit[100:89]:对于未知的广播发送到相关的VLAN的VLAN的ID。

bit[88:87]:数据的转发行为。

bit[86:79]:VLAN的类型ID。

bit[78:66]:三层查找的接口。

bit[63:60]:VLAN的特殊属性。

bit[59:48]:转发数据库ID。

bit[47:39]:生成树组ID。

bit38:1 此VLAN配置表有效。

bit[29:0]:此VLAN配置表的端口的位图。

### 6.2.5输出端口VLAN配置表

寄存器地址0x100c0000，用于配置VLAN成员和成员的状态。

bit68:1此实例是有效。

bit[67:60]:端口STP组IP。

bit[59:30]:此VLAN的端口的位图。

bit[29:0]:无标签端口位图。

### 6.2.6输出生成树状态表

寄存器地址0x10100000，

### 6.2.7VLAN生成树状态表

寄存器地址0x14040000，每个端口占用两个bit位表明端口的状态。

0：表示所有的数据包都将被丢弃。

1：表示除了控制数据包将会发送到CPU，其他数据包将会被丢弃，不进行2层学习。

2：表示除了控制数据包将会发送到CPU，其他数据包将会被丢弃，进行2层学习。

3：表示常规状态，接收所有的数据包。

### 6.2.8二层组播表

寄存器地址0x24000000，二层的组播表。

bit38:1 实例有效。

bit[37:8]:端口映射表。

### 6.2.9输出端口MTU

寄存器地址0x44001b00，用于获取和设置交换机发送数据的最大传输单元值。

1. [↑](#footnote-ref-1)