FSL91030(M)芯片

SDK 命令行配置业务示例

手册版本: V1.33

武汉飞思灵微电子技术有限公司

2023 年 4 月

目 录

1 端口编号 5
2 端口描述 6
3 设备操作说明 8
3.1 启动命令行 8
3.2 配置文件 8
3.3 访问接口选择 9
3.4 调试打印开关 9
4 寄存器及表项配置命令行 10
4.1 寄存器配置命令 10
4.1.1 reglist 10
4.1.2 getreg 10
4.1.3 setreg 10
4.1.4 modreg 10
4.2 表项配置命令 11
4.2.1 memlist 11
4.2.2 dump 11
4.2.3 modify 11
5 基本配置 13
5.1 全局配置 13
5.2 端口属性配置 13
5.3 Vlan 属性配置 15
6 Vlan 和 Port 测试 17
6.1 添加和删除 stpid 17
6.2 创建 vlan 域 17
6.3 基于 port 的 VLAN 删除 17
6.4 基于 port 的 VLAN 修改 18

6.5 基于 port + vlan 的 VLAN 删除 19
6.6 基于 port + vlan 的 VLAN 修改 19
6.7 基于 VLAN 基本转发配置 20
6.8 状态设置和查询及相关显示 20
6.9 端口隔离业务测试 23
6.10 Pdu 功能测试 24
6.11 丢包统计功能 25
7 Stg 功能 28
7.1 Stg 指令介绍 28
7.1.1 istag 指令说明 28
7.1.2 estg 指令说明 29
7.2 stg 配置实例 29
7.2.1 istg 配置实例 30
7.2.2 estg 配置实例 33
8 Erps 功能 37
8.1 Erps 指令介绍 37
8.1.1 ierps 指令说明 37
8.1.2 eerps 指令说明 38
8.2 erps 配置实例 38
8.2.1 ierps 配置实例 38
8.2.2 eerps 配置实例 40
9 Lag 测试 41
10 MAC 地址学习 43
11 MAC 地址学习限制 44
12 MAC 地址老化 45
13 Acl、eAcl 及 vfp 47
13.1 创建相关命令 48

13.2 删除相关命令 52
13.3 显示相关命令 52
13.4 命令行示例 52
14 镜像 57
14.1 输入/输出端口、输入/输出转发 VLANID 镜像 57
14.1.1 创建镜像模式 57
14.1.2 删除镜像 57
14.1.3 配置输入镜像端口 57
14.1.4 配置输出镜像端口 57
14.1.5 显示镜像配置 58
14.2 MAC 地址镜像 58
14.2.1 流镜像 58
15 流量控制 59
15.1 命令行介绍 59
15.2 配置实例 59
15.2.1 实例 1 59
15.2.2 实例 2 60
16 TM 测试 61
16.1 SP Mode（优先级模式） 61
16.2 WFQ Mode（按权重分配模式） 62
17 风暴抑制 64
18 PKT DMA 功能 65
18.1 functest pktdma 指令介绍 65
18.2 functest pktdma 配置实例 65
18.2.1 应用层收发包测试配置实例 65
18.2.2 协议栈收发包测试配置实例 66
18.2.3 应用层发包测试配置实例 67

19 修订信息 69

1 端口编号

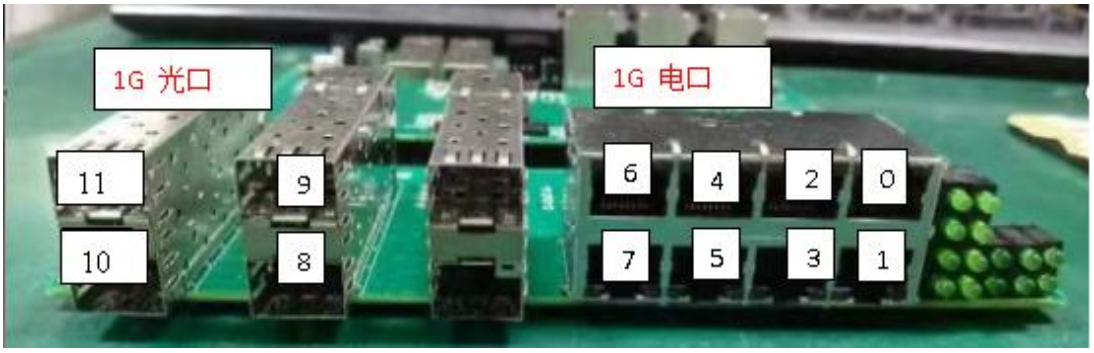
Demo 板上电后，通过串口登录：

Login：root

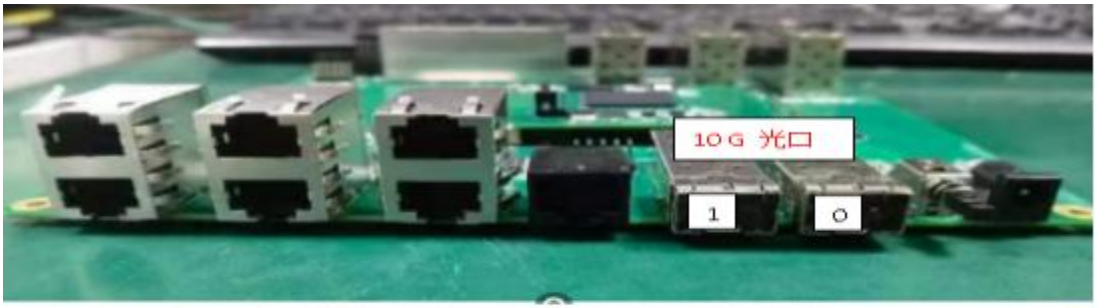
Password：fsl

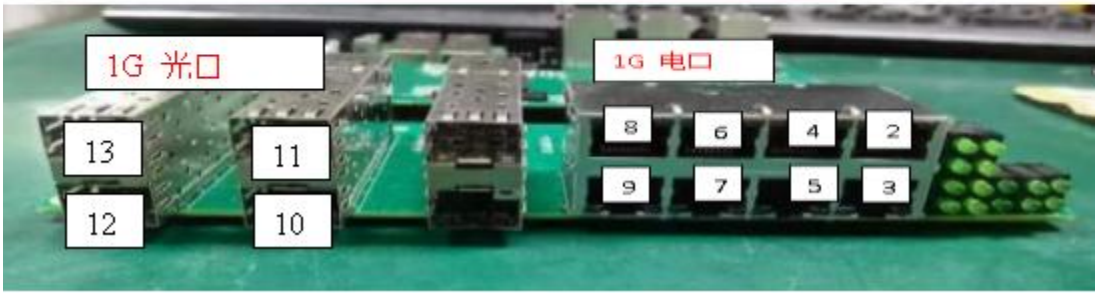
运行命令 fhcli：./fhcli

请参考 2.2 章节，将 demo 板端口形态切换成 8+4 模式：即 8 个电口加 4 个 1G 的光口的模式（芯片初始化必要执行）。



请参考 2.2 章节，将 demo 板端口形态切换成 8+4+2 模式：即 8 个电口加 4 个 1G 光口加 2 个 10G 光口的模式（芯片初始化必要执行）。





2 端口描述

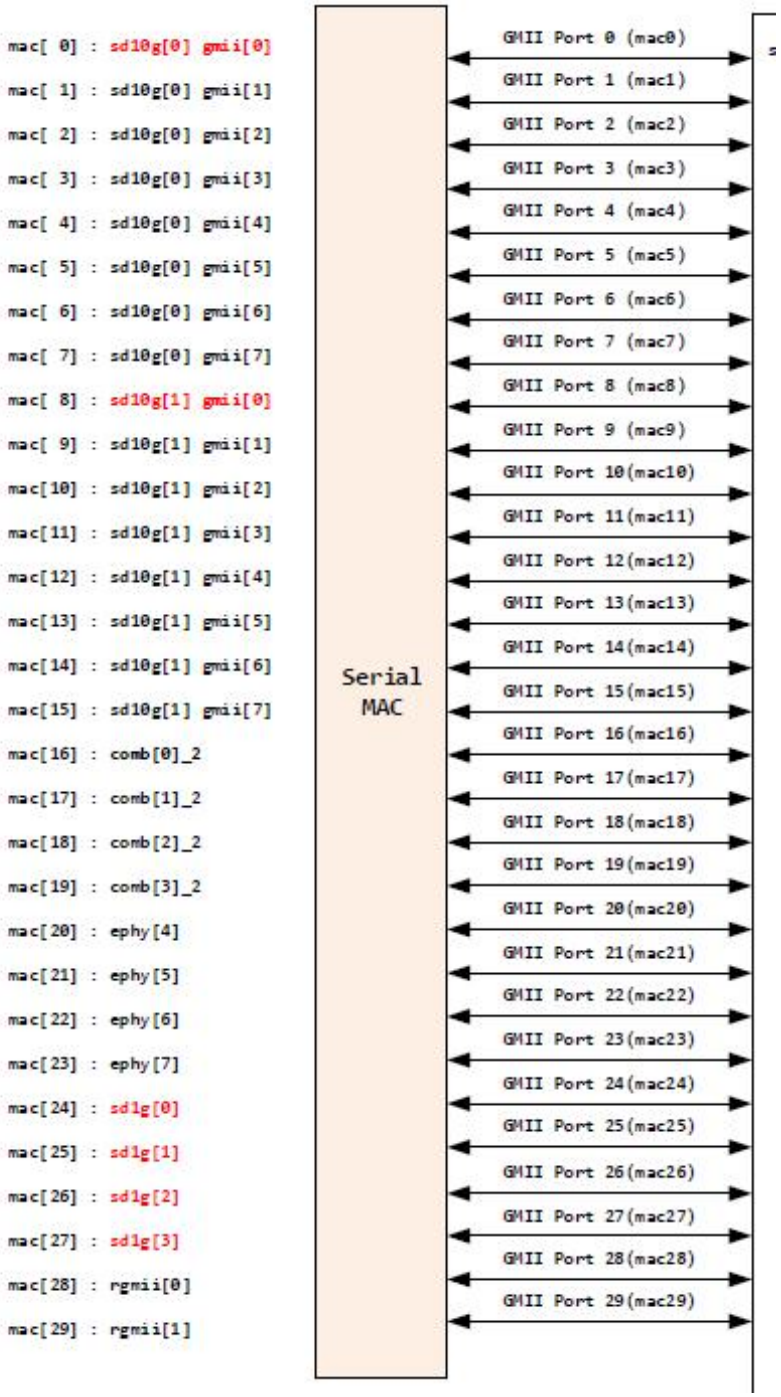
端口配置是入口管道的第一阶段，主要目的是获取数据包进入到芯片相关的端口配置信息。由于 FSL91030M 芯片具有不同的端口概念，即物理端口和数据包处理端口。广义上将物理端口 id 号和 mac id 一一对应，数据包处理端口和逻辑端口（同时和面板口）一一对应。

FSL91030M 在上电后通过应用场景和配置文件信息首先确定芯片的工作模式和端口形态，进而来确定端口的配置信息。

下表为 FSL91030M 单芯片常用的几种工作模式和端口配置信息：

业务口 物理端口/mac_id 端口形态	4 电口		8 电口		6*1G 光口		2*10G 光口 + 8 电口		4*1G 光口 + 8 电口		2*10G 光口 + 4 电口 + 4comb	
0	GE0	16	GE0	16	GE0	0	XE0	0	GE0	16	XE0	0
1	GE1	17	GE1	17	GE1	8	XE1	8	GE1	17	XE1	8
2	GE2	18	GE2	18	GE2	24	GE0	16	GE2	18	Comb0	16
3	GE3	19	GE3	19	GE3	25	GE1	17	GE3	19	Comb1	17
4			GE4	20	GE4	26	GE2	18	GE4	20	Comb2	18
5			GE5	21	GE5	27	GE3	19	GE5	21	Comb3	19
6			GE5	22			GE4	20	GE6	22	GE0	20
7			GE7	23			GE5	21	GE7	23	GE1	21
8							GE6	22	GE8	24	GE2	22
9							GE7	23	GE9	25	GE3	23
10									GE10	26		
11									GE11	27		
31 CPU												
28 RGMII0												
29 RGMII1												

FSL91030M 上有两个 10G serdes 四个 1G serdes 以及八个 GEPHY，在物理上与 mac 对接的拓扑如下：



3 设备操作说明

3.1 启动命令行

Sdk 存放目录：

/var

文件：

libsdk.so、fhcli、fh_pcie2pata.ko

设备启动之后在 linux 界面输入如下命令：

cd /var

export LD_LIBRARY_PATH=/var

./fhcli

3.2 配置文件

关于 FSL91030M 芯片 demo 板初始化：

通过读取配置文件 config.fhme（配置文件 config.fhme 在/mnt/mtd 路径下）信息来初始化，在启动 fhcli 之前将 init_sw_by_hand 改为 0，switch_mode=6 运行 fhcli 即可将芯片初始化为 8+4 模式（配置文件默认为 0，配置为 8+4 模式），将 switch_mode 修改为 7 则初始化为 8+2 模式。

```
# $Copyright: (c) 2021 fsl Corporation All Rights Reserved.$
init_sw_by_hand=0
switch_mode=6
#cpu_port_enable and cpu_port is only to inner cpu, it determines whether use a given port as cpu's net port
#cpu_port_enable:1 means use cpu port, 0 means don't use cpu port
cpu_port_enable=0
#cpu_port:the port as the cpu's net port
cpu_port=0
linkscan_enable=0
#linkscan interval
fsl_linkscan_interval=1000000
#read counter by dma
counter_dma=1
#whether to start counter thread
counter_enable=0
#ext_intf, set the interface to access switch registers.
#for inner cpu:  1(by localbus direct access),2(mdio),3(i2c)
#for extern cpu: 1(spi),2(mdio),3(i2c)
ext_intf=1
#the attributer of ext_intf.
#    For mdio or i2c, it's addr.
#    For spi,it's spi mode(0,1,2,3).
#    For localbus, it means nothing.
addr=0
#the port need to be used
pbmp_valid=0
#the phy addr on 1030M chip
phy_addr=0
ierpsNum=0
eerpsNum=0
```

如果需要使用 cpu 端口，需要将 cpu_port_enable=1，并指定有效 cpu 端口。

如果需要使用 cpu 端口，需要将 cpu_port_enable=1,并指定有效 cpu 端口。

几种常用 switch_mode 值与端口形态对应关系如下表：

switch_mode	6	7	13	15	17
端口形态描述	8+4 模式：8 个电口加 4 个 1G 光口	8+2 模式：8 个电口加 2 个万兆口	24+4 模式：24 个电口加 4 个 1 光口（用于扩 phy 情景）	8+6 模式：8 个电口加 6 个 1G 光口	8+4+2 模式：8 个电口加 4 个 1G 光口加 2 个万兆口
是否支持 RGMII	是	是	是	是	否

3.3 访问接口选择

1. 设置寄存器访问接口

内置 cpu 访问交换芯片是通过 localbus 总线直接读写，使用默认值即可，无需设置；外置 cpu 访问交换芯片可以通过 spi、mdio 或 i2c 来访问，可以通过 ext_intf 命令查看或设置访问接口。

2. 查看访问接口信息：ext_intf set

//设置为 spi ：

ext_intf set mode=1

//设置为 mdio ：

ext_intf set mode=2 addr=0x10

//设置为 i2c ：

ext_intf set mode=2 addr=0x54

3.4 调试打印开关

设置打印信息：

可以通过 debug 命令设置调试信息是否显示。

打开读写寄存器时的打印信息：debug + REG

关闭读写寄存器时的打印信息：debug - REG

4 寄存器及表项配置命令行

4.1 寄存器配置命令

4.1.1 reglist

/*列举寄存器名，”[]”参数为可选。当不接参数时将列举所有寄存器；当接参数时仅列举包含该参数字符串的寄存器*/

reglist [keywords]

示例：

//将列举所有包含“I_NET”的寄存器名

reglist I_NET

4.1.2 getreg

//获取寄存器域值，”{}”为必选参数。

getreg {reg_name}

示例：

//获取 iNetDefVlanCtl 寄存器所有域的值

getreg I_NET_DEF_VLAN_CTL

4.1.3 setreg

//设置寄存器域值，未被设置的域的值为 0。

setreg {reg_name} {field1=val1} [field2=val2] [...]

示例：

/*设置 iNetDefVlanCtl 寄存器中 portBmp 域值为 0xff*/

setreg I_NET_DEF_VLAN_CTL PORT_BMP=0xff

4.1.4 modreg

//修改寄存器域值。

modreg {reg_name} {field1=val1} [field2=val2] [...]

示例：

//设置 iNetDefVlanCtl 寄存器 portBmp 域值为 0xff，其他未设置的域值不变

modreg I_NET_DEF_VLAN_CTL PORT_BMP=0xff

4.2 表项配置命令

4.2.1 memlist

/*列举表项名，”[]”参数为可选。当不接参数时将列举所有表项名；当接参数时仅列举包含该参数字符串的表项名*/

memlist [keywords]

示例：

//将列举所有包含“I_NET”的表项名

memlist I_NET

4.2.2 dump

/*获取表项域值。当不指定参数 index 和 cnt 时，会将该表项中所有条目列举出来(列举出的条目数取决于表项深度)；当指定 index 时，列举索引为 index 的条目；当指定 index 和 cnt 时，列举基址索引为 index 的 cnt 条条目*/

dump {mem_name} [index] [cnt]

示例：

//获取 iNetVlanSrm 表项第 100 个条目的域的值。

dump I_NET_VLAN_SRM 100

//获取 iNetVlanSrm 表项从第 100 个条目开始的三个条

dump I_NET_VLAN_SRM 100 3

4.2.3 modify

//修改表项域值。

modify {mem_name} {index} {cnt} {field1=val1} [field2=val2] [...]

示例：

/*修改 iNetVlanSrm 表中第 100 个条，目的 portBmp 域值为 0xff，其他未设置的域值不变*/

modify I_NET_VLAN_SRM 100 1 PORT_BMP=0xff

/*修改 iNetVlanSrm 表中从第 100 个条开始的三个条目的 portBmp 域值为 0xff，其他未设置的域值不变*/

modify I_NET_VLAN_SRM 100 3 PORT_BMP=0xff

5 基本配置

5.1 全局配置

//type：端口属性类型。取值及含义见下表

//val：端口属性值

Global set type={type} value={val}

Global get type={type}

Global 设置 type	Type 描述
passLportLkp=0	使能 lag 口功能，一般管理型芯片默认使能 Lag 功能，如果配置错误则可以通过该配置强制使能 lag 功能；非管理型不支持 lag 功能。
stpChkEn=1	使能全局 stp 检查
stpDisableDropEn=2	Stp disable 状态丢弃使能
stpDisableChkEn=3	Stp disable 状态检查使能
iFwdLrnLmtCtl=4	iFwd 学习门限使能

5.2 端口属性配置

//port_id：端口号。取值 0~29

//type：端口属性类型。取值及含义见下表

//val：端口属性值

Port attrset port={port_id} type={type} value={val}

Port type 类型	Type 描述取值
fslPortControlXlateEn0 = 0,	Xlate0 使能
fslPortControlXlateEn1,	Xlate1 使能
fslPortControlFlowVlan0En,	Vlan Flow 使能
fslPortControlProtoVlanEn,	Protocol vlan 使能
fslPortControlSvlanRangeEn,	Svlan Range 使能
fslPortControlCvlanRangeEn,	Cvlan Range 使能
fslPortControlScosMapEn,	Svlan cos 映射使能
fslPortControlCcosMapEn,	Cvlan cos 映射使能
fslPortControliVtEditEn,	入向 vlan 编辑使能
fslPortControlLrnDisable,	不学习指示
fslPortControlPriVld,	端口优先级有效标识

fslPortControlLeftAlgTp0,	左哈希表算法，具体编码如下： 0x0:使用 crc32 运算结果低位 0x1:使用 crc16-BISYNC 运算结果低位 0x2:使用 crc16-CCITT 运算结果低位 0x3:使用 key 值低位
fslPortControlRightAlgTp0,	右哈希表算法，具体编码如下： 0x0:使用 crc32 运算结果低位 0x1:使用 crc16-BISYNC 运算结果低位 0x2:使用 crc16-CCITT 运算结果低位 0x3:使用 key 值低位
fslPortControlLeftAlgTp1,	左哈希表算法，具体编码如下： 0x0:使用 crc32 运算结果低位 0x1:使用 crc32 运算结果高位 0x2:使用 crc16 运算结果 0x3:使用 key 值
fslPortControlRightAlgTp1,	右哈希表算法，具体编码如下： 0x0:使用 crc32 运算结果低位 0x1:使用 crc32 运算结果高位 0x2:使用 crc16 运算结果 0x3:使用 key 值
/*inet*/	
fslPortControlerpsLkpEn = 20,	环网保护检查使能
fslPortControlinPortMirEn,	输入端口镜像使能
fslPortControlLrnUpdDisable,	学习更新禁止使能
fslPortControlSmvFlag,	站点移位控制使能，各比特位的含义描述如下： [0]:丢弃使能 [1]:trap 到 CPU 使能 [2]:站点移位使能
fslPortControlHmClass,	站点移位情况下端口的优先级
fslPortControlAlwSameClsSmv,	允许相同优先级的端口进行站点移位操作
fslPortControlBrgEn,	二层桥接查找使能，高有效
fslPortControlVlanFilterEn,	Vlan 滤除使能
fslPortControlStpChEn,	生成树检查使能
fslPortControlUsePktSvid,	使用原始报文的 svid 作为内部 vlanId
fslPortControlUseUpdSvid,	使用修改后的 svid 作为内部 vlanId
/*ifwd*/	
fslPortControlDropUkwUc = 40,	未知单播丢弃使能
fslPortControlProtId,	保护组 id，可以创建 4 个保护组
fslPortControlProtInd,	保护端口指示，2’b00,非保护组，2'b10：1：1 保护组; 2'b11:1+1 保护组
fslPortControlLrnMissToCpu,	Mac 地址原地址查找失效丢弃使能
fslPortControlLrnMissToDrop,	Mac 地址原地址查找失效 trap 到 cpu 使能

fslPortControlProGroup,	保护端口指示，入端口属于保护端口，且报文为非 OAM 类型报文，或者开启丢弃 OAM 指示，报文丢弃
fslPortControlAllowPortToSrc,	环回使能，允许相同端口环回
fslPortControlPortIsotEnBmp,	端口隔离使能，各比特具体描述如下：[0]:已知单播 [1]:未知单播 [2]:已知组播 [3]:未知组播 [4]:广播
/*iacl*/	
fslPortControliAcl0LkpVld = 60,	iAcl0 查询开关
fslPortControliAcl1LkpVld,	iAcl1 查询开关
/*eee*/	
fslPortControlEvlanFilterEn = 70,	出口 vlan 过滤
fslPortControlEerpsLkEn,	出口环网保护查询开关
fslPortControlEoutPortMirEn,	出口镜像
fslPortControlEvtEditEn,	出口 vlan 编辑
/*epf*/	
fslPortControlUpdEn = 80,	Trunk 端口失效分担使能
fslPortControlOutStpChkEn,	出口 stp 检查使能
fslPortControlOutPortIsUnStag,	出口剥离外层 Stag
fslPortControlOutPortIsUnCtag,	出口剥离内层 Ctag
fslPortControlPortIsotEn,	端口隔离使能
fslPortControlNetworkPort,	水平分割使能
/*eacl*/	
fslPortControleAcl0LkpVld =90,	eAcl0 查询开关
fslPortControleAcl1LkpVld,	eAcl1 查询开关

5.3 Vlan 属性配置

//vlan_id：vlan ID。取值 0~4091

//type：vlan 属性类型。取值及含义见下表

//val：vlan 属性值

vlan vlanset vlanid={vlan_id**}　type={**type**}　value={**val**}**

Vlan Type 类型	Type 描述取值
fslVlanDropUnknown = 0,	未知单播报文丢弃使能
fslVlanMacList,	Mac 地址黑白名单模式 1：黑名单 2:白名单

fslVlanFidStmCtlVld,	基于 fid 的风暴控制使能
fslVlanMacLearnDisable,	禁止 mac 地址学习使能
fslVlanMacLearnUp,	禁止 mac 地址学习更新使能
fslVlanIsotIdx = 5,	基于 vlan 的端口隔离指针
fslVlanIsotEn,	基于 vlan 的端口隔离使能
fslVlanPduBypassStp,	识别 pdu 的旁路生成树检查使能
fslVlanErpsId,	ERPS 环网保护查表 ID
fslVlanStpId = 20,	STP 生成树 ID
fslVlanPriVld,	内部优先级和颜色有效指示
fslVlanPriority,	内部优先级
fslVlanColor,	颜色：0x00: RED 0x01: YELLOW 0x02:GREEN
fslVlanQosProfileVld,	Qos 模板有效指示
fslVlanQosProfileIdx,	Qos 模板索引
fslVlanInFpolVld,	入口层次化 Meter 中小管道有效指示
fslVlanTrapEn,	trap 到 CPU 使能
fslVlanDropEn,	丢弃使能
fslVlanBypassEn,	旁路使能
fslVlanDot1xEn,	1x 认证使能，高有效
fslVlanInMirEn,	入口的镜像使能
fslVlanOutErpsId = 40,	出端口 ERPS 环网保护查表 ID
fslVlanOutStpId,	出端口 STP 生成树 ID
fslVlanOutFpolVld,	出口层次化 Meter 中小管道有效指示

6 Vlan 和 Port 测试

6.1 添加和删除 stpid

1. 入方向：

port addtpid Port=14 Tpid=0x8100

port deltpid Port=20 Tpid=0x8100

2. 出方向：

port settpid Port=14 Tpid=0x9100

6.2 创建 vlan 域

1. 端口 5 使用原始报文的 svid 作为内部 vlanId

port attrset port=5 type=29 value=1

2. 逻辑端口成员

//100 表示 VLAN ID，0xfff 表示添加的逻辑端口成员 0~11

vlan create 100 pbmp=0xfff

3. 对 vlan 域内的出端口剥 vlan

/*100 表示 VLAN ID，0x1 表示对出端口 0 进行 untag 去除 vlan 动作，注意，配置 ubmp 时，pbmp 成员也要带上，因为 vlan 原理规定 ubmp 必须是 pbmp 的子集*/

vlan create 100 pbmp=0xfff ubmp=0x1

4. 删除 vlan 域内端口成员

//100 表示 VLANID，0x2 表示删除 vlan 100 的端口成员 1

vlan remove 100 pbmp=0x2

6.3 基于 port 的 VLAN 删除

1. 业务配置

在端口 6（电口）编辑 8 条单播以太报文，格式为：MacDa、MacSa、Vlan（tpid=0x8100，cos=0，cfi=0，vid=0x100）；

2. 配置命令：

//pbmp：一个端口占 1bit，1111110

vlan create 0x100 pbmp=0x7E

//type=8 为 vlan edit 使能

port attrset port=6 type=8 value=1

//port 6 端口默认 vlan 编辑操作为删除外层 Vlan

vlan action port default add port=6 sotovid=delete

3. 期望结果：

在端口 1~5 收到报文，其中报文 VLAN 被剥离。

6.4 基于 port 的 VLAN 修改

1. 业务配置：

在端口 6（电口）编辑 8 条单播以太报文，格式为：MacDa、MacSa、Vlan（tpid=0x8100，cos=0，cfi=0，vid=0x100）；

2. 配置命令：

vlan create 0x100 pbmp=0x7E

//type=8 为 vlan edit 使能

port attrset port=6 type=8 value=1

//port 6 端口默认 vlan 编辑操作为修改外层 Vlan id 为 0x64

vlan action port default add Port=6 newsvid=0x64 sotovid=replace

3. 期望结果：

在端口 1~5 收到报文，其中报文 VLANID 被修改为 0x64。

6.5 基于 port + vlan 的 VLAN 删除

1. 业务配置

在端口 6（电口）编辑 8 条单播以太报文，格式为：MacDa、MacSa、Vlan（tpid=0x8100，cos=0，cfi=0，vid=0x100）；

2. 配置命令

vlan create 0x100 pbmp=0x7E

//type=8 为 vlan edit 使能

port attrset port=6 type=8 value=1

//type=0 为 vlanxlate0 使能

port attrset port=6 type=0 value=1

//Xlate0 使用 port+vlanid key，将匹配的外层 vlan 删除

vlan action translate add port=6 keytype=PortOuter oldoutervLan=0x100 sotovid=delete

3. 期望结果

在端口 1~5 收到报文，其中报文 VLAN 被剥离。

6.6 基于 port + vlan 的 VLAN 修改

1. 业务配置

在端口 6（电口）编辑 8 条单播以太报文，格式为：MacDa、MacSa、Vlan（tpid=0x8100，cos=0，cfi=0，vid=0x100）；

2. 配置命令

vlan create 0x100 pbmp=0x7E

//type=8 为 vlan edit 使能

port attrset port=6 type=8 value=1

//type=0 为 vlanxlate0 使能

port attrset port=6 type=0 value=1

//Xlate0 使用 port+vlanid key，将匹配的外层 vlan id 修改为 0x64

vlan action translate add port=6 keytype=PortOuter oldoutervLan=0x100 newsvid=0x64 sotovid=replace

3. 期望结果

在端口 1~5 收到报文，其中报文 VLANID 被修改为 0x64。

6.7 基于 VLAN 基本转发配置

1. 创建 Vlan 域

//端口 5 使用原始报文的 svid 作为内部 vlanId

port attrset port=5 type=29 value=1

//100 表示 VLANID，0xfff 表示添加的逻辑端口成员 0~11

vlan create 100 pbmp=0xfff

2. 对 vlan 域内的出端口剥 vlan 功能

//100 表示 VLANID，0x1 表示对出端口 0 进行 untag 去除 vlan 动作

vlan create 100 ubmp=0x1

3. 删除 vlan 域内端口成员

//100 表示 VLANID,0x2 表示删除 vlan 100 的端口成员 1

vlan remove 100 pbmp=0x2

6.8 状态设置和查询及相关显示

1. 端口状态设置

//配置 1 号端口使能，1 使能、 0 关闭使能

PORT ctlset port=1 enable=0/1

//配置端口自协商状态 1 开启、0 关闭

PORT ctlset port=1 an=0/1

//配置端口速率(目前支持电口)，单位 M

PORT ctlset port=1 speed=10/100/1000

//配置端口双工状态(目前支持电口)，0 半双工、1 全双工

PORT ctlset port=1 duplex=0/1

//配置端口在自协商开启情况下，协商能力

PORT ctlset port=<> ability=<0x7e0>(bit[5]:10M H bit[6]:10M F bit[7]:100M H bit[8]:100M F bit[9]:1000M H bit[10]:1000M F)

//重置 phy

PORT ctlset port=1 resetphy=<0|1>

2. 端口状态显示命令

Portsta

num	mac_id	port	ena/ link	speed/ duplex	link scan	auto nego	STP state	pause	dis card	lrn ops	inter face	max frame	loop back
0	16	ge_phy	up	1000M FD	SW	Yes	Forward	- -	NULL	NULL	NULL	16000	MAC
1	17	ge_phy	up	100M HD	SW	Yes	Forward	- -	NULL	NULL	NULL	16000	MAC
2	18	ge_phy	down	1000M FD	SW	Yes	Forward	- -	NULL	NULL	NULL	16000	MAC
3	19	ge_phy	down	1000M FD	SW	Yes	Forward	- -	NULL	NULL	NULL	16000	MAC
4	20	ge_phy	down	1000M FD	SW	Yes	Forward	- -	NULL	NULL	NULL	16000	MAC
5	21	ge_phy	down	1000M FD	SW	Yes	Forward	- -	NULL	NULL	NULL	16000	MAC
6	22	ge_phy	down	1000M FD	SW	Yes	Forward	- -	NULL	NULL	NULL	16000	MAC
7	23	ge_phy	down	1000M FD	SW	Yes	Forward	- -	NULL	NULL	NULL	16000	MAC
8	24	ge	down	1000M FD	SW	Yes	Forward	- -	NULL	NULL	NULL	16000	MAC
9	25	ge	up	1000M FD	SW	Yes	Forward	- -	NULL	NULL	NULL	16000	MAC
10	26	ge	down	1000M FD	SW	Yes	Forward	- -	NULL	NULL	NULL	16000	MAC
11	27	ge	up	1000M FD	SW	Yes	Forward	- -	NULL	NULL	NULL	16000	MAC

3. 端口包计数显示命令

1) 直接读取寄存器来显示某个端口的计数：

//端口号为图中的 mac_id

port showcounter port=16

port	mac_id	frame_type	Tx_count	Rx_count
16	16	Totol_pkt	0	0
16	16	Totol_bytes	0	0
16	16	Control	0	0
16	16	Pause	0	0
16	16	Good	0	0
16	16	Ucast	0	0
16	16	Mcast	0	0
16	16	Bcast	0	0
16	16	undersize	0	0
16	16	fragment	0	0
16	16	oversize	0	0
16	16	jabber	0	0
16	16	bad	0	0
16	16	[0-64]	0	0
16	16	[65-127]	0	0
16	16	[128-255]	0	0
16	16	[256-511]	0	0
16	16	[512-1023]	0	0
16	16	[1024-1518]	0	0
16	16	[1519-2047]	0	0
16	16	[2048-4095]	0	0
16	16	[4096-9215]	0	0
16	16	[9215-MTU]	0	0

2) 通过 dma 方式来读取计数

启动 fhcli 前需要在配置文件 config.fhme 中设置 counter_dma=1

命令：

//port 0 两次 show 计数的改变情况

Show c c pbmp(指令不带 pbmp 时显示全部端口)

port	mac_id	frame_type	Tx_count	Rx_count	Tx_change	Rx_change
0	16	Totol_pkt	0	8545657	+0	+100
0	16	Totol_bytes	0	1093844096	+0	+12800
0	16	Good	0	8545657	+0	+100
0	16	Ucast	0	8545657	+0	+100
0	16	[128-255]	0	8545657	+0	+100

//显示端口 0 中计数不为零的情况

Show c nz pbmp(指令不带 pbmp 时显示全部端口)


```
port  mac_id  frame_type  Tx_count                Rx_count
0     16      fps         0/s                     84458/s
0     16      bps         0/s                     86485289/s
0     16      Totol_pkt   0                       718887
0     16      Totol_bytes 0                       92032384
0     16      Good        0                       719343
0     16      Ucast       0                       719455
0     16      [128-255]   0                       719740
```

4. Mac 地址 display

命令：

l2 show

```
              mac  vlan    port   fecpath  fecpatTp  static
00:00:00:00:01:23     2       1         0   FEC_NOP       1

******************TOTAL:1
```

5. Vlan display

命令：

vlan show

```
vlanId          PBMP        LAGBMP       UNTAGPBMP
     2          0xff           0x0             0xf
     3         0xfff           0x0            0xf7

******************TOTAL:2
```

vlan show <vid>

```
FSLcli.0> [/root] vlan show 100
vlanId          PBMP        LAGBMP       UNTAGPBMP
   100          0x30           0x2             0x0
```

6.9 端口隔离业务测试

端口隔离业务主要是将入端口和某几个端口进行业务隔离；

配置命令行：

Port portisolate inport=4 pbmp=0xf enable=1

端口 4 进来的报文，端口 0 1 2 3 收不到报文

//inport 为入端口

//pbmp 为隔离端口位图

//enable 为端口隔离使能(0 关闭，1 开启)

6.10 Pdu 功能测试

Pdu 支持 32 个条目，顺序匹配，支持匹配的域和对应掩码如下：

掩码 0：不匹配，1 匹配。

匹配域	命令行字段	掩码	命令行字段
smac（全局，32 条共用）	MACaddress	具体条目中配置，掩码为 0/1	mask_smac
dmac	dmac	16-47bit 掩码为 0/1； 0-15bit 为逐 bit 掩，0xFFFF，所以 dmac 掩码设置范围为 0-0x1FFFF	mask_dmac
ethType	ethType	0/1	mask_ethType
stag.isValid	stagVld	0/1	mask_stagVld
svid	svid	0/1	mask_svid
ctag.isValid	ctagVld	0/1	mask_ctagVld
cvid	cvid	0/1	mask_cvid
l2Tp	l2Tp	0/1	mask_l2Tp

32 个 pdu 条目中，第 0 条是默认匹配 dmac 为 0x0180C2000000，dmac 掩码为 0x1FFFE 的报文，第一条是默认匹配 dmac 为 0x0180C2000000，dmac 掩码为 0x1FFF0 的报文，其他条目都是空配置。

1. 创建 vlan=100 转发域

vlan create 100 pbmp=0xfff

2. 设置 pdu 全局源 mac。

vlan action pdu smac set MACaddress=0x2345678

3. 设置 pdu 条目 5 的匹配 smac 为 0x2345678，dmac 为 0xaabbcc 的报文上送 cpu

vlan action pdu config add index=5 dmac=0xaabbcc mask_dmac=0x1FFFF mask_smac=1

4. 设置条目 5 的 pdu 动作，0：不处理；1：丢弃；2：trap；3：丢弃+trap；

vlan action pdu option set index=5 option=2

5. 设置 trap 口为 cpu 口

dp trap set port=31

6.11 丢包统计功能

1. 丢包统计初始化（命令行启动时已预设，不需要用户调用）

show drop init

2. 查看端口 3 入向丢包

show drop count port=3 direc=0

3. 查看端口 5 出向丢包

show drop count port=5 direc=1

4. 清除所有端口的丢包统计

show drop clear

丢包原因汇总：

0	NOP	--
1	IPR0ERR	包解析错误
2	MCSMACERR	源 MAC 为组播 MAC 地址
3	PORTPDUIDFY	PDU 匹配丢包
4	PORTDOT1XPROTO	基于端口的 802.1x 认证丢包
5	PORTDOT1X	基于端口的 802.1x 认证不通过
6	IAFT	AFT 对 tag、utag 报文滤除行为丢包
7	IXLATE	入向 vlan 翻译匹配动作丢包
8	MACIPBIND	MAC、IP 绑定 miss 丢包
9	IVLAN	入向 vlan 域动作丢包
10	ISTPDISABLE	入向 stp 状态为 disable
11	VLANDOT1XPROTO	基于 vlan 的 802.1x 认证丢包
12	VLANDOT1X	基于 vlan 的 802.1x 认证不通过
13	IVLANFILTER	入向 vlan 过滤
14	VLANPDUIDFY	PDU 匹配丢包
15	ISTPCHECK	入向 stp 状态为非 forwarding
16	IPOPTION	带扩展头的 ip 报文端口行为丢包
17	IPHDRERR	ip 报文头错误
18	V4HDRCHECKFAIL	ipv4 报文 ttl 等于 0 或 1 或者源 ip 等于目的 ip
19	V6HDRCHECKFAIL	ipv6 报文 ttl 等于 0 或 1 或者源 ip 等于目的 ip
20	L4HDRERR	--
21	TCPDOSCHECKFAIL	tcp 或 udp DOS 攻击报文命中丢弃

22	ICMPDOSCHECKFAIL	icmp DOS 攻击报文命中丢弃
23	RXPROT	入端口保护时 OAM 包丢弃
24	DMAC	目的 mac 地址命中动作为 trap
25	DMACFILTER	目的 mac 地址命中动作为 drop
26	CCPORTNOMATCH	SCC/DCC 源路径检查端口失败转发时目的 mac 为 SCC、DCC 类型时命中丢弃
27	CCNOMATCH	转发时目的 mac 为 SCC、DCC 类型时不命中丢弃
28	PFMFILTER	组播 pfm 过滤丢包
29	UCLOOPAVOIDPORT	单播报文目的端口等于源端口
30	UCLOOPAVOIDPORTCC	SCC、DCC 转发类型出端口等于源端口
31	STORMCTL	风暴抑制丢包
32	SMAC	学习时源 MAC 命中 trap
33	LRNSMVCHKFAIL	学习时站点移位失败
34	LRNBKTOVFW	学习 MAC 地址表溢出
35	SMACFILTER	学习时源 MAC 命中丢弃
36	SMACMISSDROP	源 MAC 地址查找失败丢弃
37	SMACBLACKLIST	黑名单命中丢弃
38	SMACWHITELIST	白名单不命中丢弃
39	LRNSMV	站点移位 trap
40	LRNSMVCLS	站点移位丢弃
41	LRNMACNUMLMT	MAC 地址学习条目超过门限
42	IACL0LKP	入向 acl0 匹配动作丢包
43	IACL1LKP	入向 acl1 匹配动作丢包
44	IPOLDROP	入向 Policing 限速依据报文颜色丢包
45	FLDNOVLDBMP	PortBitmap 里没有有效端口
46	EVLAN	出向 vlan 域动作丢包
47	EXLATE	出现 vlan 翻译匹配动作丢包
48	EXLATEMISS	出现 vlan 翻译不匹配动作丢包
49	LAGNOMEMBER	聚合组没有端口成员
50	OUTPORT	出口动作丢包
51	REMOTELOOPBACK	--
52	HORIZONSPLTPORT	水平分割
53	ICMPREDIR	--
54	MCLOOPAVOIDPORT	组播报文目的端口等于源端口
55	MCLOOPAVOIDPORTCC	出方向 SCC、DCC 转发类型出端口等于源端口
56	ESTPDISABLE	出向 stp 状态为 disable
57	ESTPCHECK	出向 stp 状态为非 forwarding
58	EVLANFILTER	出向 vlan 过滤
59	PORTISOLATE	端口隔离
60	EACL0LKP	出向 acl0 匹配动作丢包
61	EACL1LKP	出向 acl1 匹配动作丢包
62	EPOLDROP	出向 Policing 限速依据报文颜色丢包
63	SAMEMAC	报文源 mac 等于目的 mac

7 Stg 功能

根据 mstp 协议，将多个不同的 vlan 合在一起生成一个 stp 组（即 stg），相同 stg 组内 vlan 共享端口的 stp 状态。在实现 stg 功能时，将 I_NET_VLAN_SRM 表项域 STP_ID 理解为 stg ID，通过该 ID 去查 I_NET_STP_SRM 表来获取 stg 的端口状态。因此，stg ID 必须小于 I_NET_STP_SRM 的表项深度。

Stp 分为 istp 和 estp 两部分，为了简化使用，将 stg 分为 istg 和 estg 两个部分，具体指令如下，“[ ]”表示可选，“|”表示多选 1。

7.1 Stg 指令介绍

7.1.1 istag 指令说明

/*istg 初始化，初始化之后，会创建默认 stgid=0，包含 vlan1-vlan4095，stg0 所有 port 和 lag 口的 stp 状态都是 diable (端口 stp_check 使能之后才生效，使能之前是 forward 状态)。*/

istg init

//istg 功能关闭

istg deinit

//对应端口的 stpcheck 使能

istg enable pbmp=<pbmp>

//对应端口的 stpcheck 去使能

istg disable pbmp=<pbmp>

//创建 STGs，可以创建多个 stg

istg create [<id>] [...]

//取消 STGs 配置

istg destroy <id> [...]

//查看已配置的 STG(s)，显示 stg 包含的 vlan 以及对应的 stp 状态

istg show [<id>]

//添加 VLAN(s)到一个 STG

istg add <id> <vlan_id> [...]

//从一个 STG 中移除 VLAN(s)

istg remove <id> <vlan_id> [...]

//显示所有 stg 端口状态

istg stp

//显示某个 stg 端口状态

istg stp <id>

//设置某个 STG 的 stp 状态

istg stp <id> <pbmp=xx>[<lbmp=xx>] <state=disable|block|learn|forward>

7.1.2 estg 指令说明

estg 指令和 istg 类似，这里不再详细说明。

```
estg init

estg deinit

estg enable pbmp=<pbmp>

estg disable pbmp=<pbmp>

estg create [<id>][...]

estg destroy <id>[...]

estg show [<id>]

estg add <id> <vlan_id> [...]

estg remove <id> <vlan_id> [...]

estg stp

estg stp <id>

estg stp <id> <pbmp=xx>[<lbmp=xx>] <state=disable|block|learn|forward>
```

7.2 stg 配置实例

7.2.1 istg 配置实例

创建 4 个 stg，分别为 stg10 到 stg13，配置 lag0 包含 port2 和 port3。

stg10 包含 vlan 100-104，配置 port0 、lag0 为 disable 状态

stg11 包含 vlan 105-109，配置 port0 、lag0 为 block 状态

stg12 包含 vlan 110-114，配置 port0 、lag0 为 learn 状态

stg13 包含 vlan 115-119，配置 port0 、lag0 为 forward 状态

1. Lag 配置

trunk init

//创建一个 lag 组（lagID 为 0），端口成员为 2、3， hashkey 为 DstIp 和 SrcMac。

trunk add Id=0 Rtag=0x2001 Pbmp=0xC

/*使能 lag 口功能，一般管理型芯片默认使能 Lag 功能，如果配置错误则可以通过该配置强制使能 lag 功能；非管理型不支持 lag 功能*/

global set type=0 value=0

2. Vlan 配置

```
//创建 100-119 的 vlan 域，包含 port0-1 以及 lag0
vlan create 100 pbmp=0x3  lbmp=0x1
vlan create 101 pbmp=0x3  lbmp=0x1
vlan create 102 pbmp=0x3  lbmp=0x1
vlan create 103 pbmp=0x3  lbmp=0x1
vlan create 104 pbmp=0x3  lbmp=0x1
vlan create 105 pbmp=0x3  lbmp=0x1
vlan create 106 pbmp=0x3  lbmp=0x1
vlan create 107 pbmp=0x3  lbmp=0x1
vlan create 108 pbmp=0x3  lbmp=0x1
vlan create 109 pbmp=0x3  lbmp=0x1
```



```
vlan create 110 pbmp=0x3  lbmp=0x1

vlan create 111 pbmp=0x3  lbmp=0x1

vlan create 112 pbmp=0x3  lbmp=0x1

vlan create 113 pbmp=0x3  lbmp=0x1

vlan create 114 pbmp=0x3  lbmp=0x1

vlan create 115 pbmp=0x3  lbmp=0x1

vlan create 116 pbmp=0x3  lbmp=0x1

vlan create 117 pbmp=0x3  lbmp=0x1

vlan create 118 pbmp=0x3  lbmp=0x1

vlan create 119 pbmp=0x3  lbmp=0x1

//使用包的 vlan 信息

port attrset port=0 type=29 value=1

port attrset port=1 type=29 value=1

port attrset port=2 type=29 value=1

port attrset port=3 type=29 value=1
```

3. istg 配置

```
istg init

//是能端口 0、2、3 stp 检查

istg enable pbmp=0xD

//创建 istg 10-13

istg create 10 11 12 13

//istg10 中添加 vlan 100-104

istg add 10 100 101 102 103 104

//istg11 中添加 vlan 105-109

istg add 11 105 106 107 108 109

//istg12 中添加 vlan 110-114
```

```
istg add 12 110 111 112 113 114

//istg13 中添加 vlan 115-119

istg add 13 115 116 117 118 119

//istg10 port0，lag0 stp 状态为 diable

istg stp 10 pbmp=0x1  lbmp=0x1 state=disable

//istg11 port0，lag0 stp 状态为 block

istg stp 11 pbmp=0x1  lbmp=0x1 state=block

//istg12 port0，lag0 stp  状态为 learn

istg stp 12 pbmp=0x1  lbmp=0x1 state=learn

//istg13 port0，lag0 stp 状态为 forward

istg stp 13 pbmp=0x1  lbmp=0x1 state=forward

//查看 stg 配置

istg show
```

测试 1：

port0 配置 SMAC、DMAC 默认值开始，每次递增 1，vlan 从 100 到依次递增到 119，burst 模式发送 20 条，查看 port1-3 的收包情况。

期望结果：port0 发送 20 条报文，port1 收到 5 条，port2 和 3 一起收到 5 条。

测试 2：

Port1 配置 SMAC、DMAC 和测试 1 互换，vlan 从 100 到依次递增到 119，burst 模式发送 20 条，查看 port0，port2-3 的收包情况。

期望结果：port0 收到 20 条报文，port2 和 3 一起收到 10 条。

测试 3：

Port2 配置 SMAC、DMAC 和测试 1 互换，vlan 从 100 到依次递增到 119，burst 模式发送 20 条，查看 port0，port1 的收包情况。

期望结果：port0 收到 5 条报文，port1 未收到包。

7.2.2 estg 配置实例

创建 8 个 estg，分别为 stg10 到 stg17，配置 lag0 包含 port2 和 port3。

estg10 包含 vlan 100-104，配置 port1 为 disable 状态，lag0 disable

estg11 包含 vlan 105-109，配置 port1 为 block 状态，lag0 disable

estg12 包含 vlan 110-114，配置 port1 为 learn 状态，lag0 disable

estg13 包含 vlan 115-119，配置 port1 为 forward 状态，lag0 disable

estg14 包含 vlan 120-124，配置 lag0 为 disable 状态，port1 disable

estg15 包含 vlan 125-129，配置 lag0 为 block 状态，port1 disable

estg16 包含 vlan 130-134，配置 lag0 为 learn 状态，port1 disable

estg17 包含 vlan 135-139，配置 lag0 为 forward 状态，port1 disable

1. 将 istg 配置清除

istg deinit

2. Lag 配置

trunk init

//创建一个 lag 组（lagID 为 0），端口成员为 2、3， hashkey 为 DstIp 和 SrcMac。

trunk add Id=0 Rtag=0x2001 Pbmp=0xC

3. Vlan 配置

```
//创建 100-139 的 vlan 域，包含 port0-1 以及 lag0
vlan create 100 pbmp=0x3  lbmp=0x1
vlan create 101 pbmp=0x3  lbmp=0x1
vlan create 102 pbmp=0x3  lbmp=0x1
vlan create 103 pbmp=0x3  lbmp=0x1
vlan create 104 pbmp=0x3  lbmp=0x1
vlan create 105 pbmp=0x3  lbmp=0x1
```



```
vlan create 106 pbmp=0x3  lbmp=0x1

vlan create 107 pbmp=0x3  lbmp=0x1

vlan create 108 pbmp=0x3  lbmp=0x1

vlan create 109 pbmp=0x3  lbmp=0x1

vlan create 110 pbmp=0x3  lbmp=0x1

vlan create 111 pbmp=0x3  lbmp=0x1

vlan create 112 pbmp=0x3  lbmp=0x1

vlan create 113 pbmp=0x3  lbmp=0x1

vlan create 114 pbmp=0x3  lbmp=0x1

vlan create 115 pbmp=0x3  lbmp=0x1

vlan create 116 pbmp=0x3  lbmp=0x1

vlan create 117 pbmp=0x3  lbmp=0x1

vlan create 118 pbmp=0x3  lbmp=0x1

vlan create 119 pbmp=0x3  lbmp=0x1

//使用包的 vlan 信息

port attrset port=0 type=29 value=1

port attrset port=1 type=29 value=1

port attrset port=2 type=29 value=1

port attrset port=3 type=29 value=1
```

4. estg 配置

```
estg init

//是能端口 1、2、3 stp 检查

istg enable pbmp=0xE

//创建 estg 10-17

estg create 10 11 12 13 14 15 16 17

//estg10 中添加 vlan 100-104

estg add 10 100 101 102 103 104
```

```
estg add 11 105 106 107 108 109

estg add 12 110 111 112 113 114

estg add 13 115 116 117 118 119

estg add 14 120 121 122 123 124

estg add 15 125 126 127 128 129

estg add 16 130 131 132 133 134

estg add 17 135 136 137 138 139

estg stp 10 pbmp=0x2  lbmp=0x1 state=disable

estg stp 11 lbmp=0x1   state=disable

estg stp 11 pbmp=0x2  state=block

estg stp 12 lbmp=0x1  state=disable

estg stp 12 pbmp=0x2  state=learn

estg stp 13 lbmp=0x1 state=disable

estg stp 13 pbmp=0x2  state=forward

estg stp 14 pbmp=0x2  lbmp=0x1 state=disable

estg stp 15 pbmp=0x2  state=disable

estg stp 15 lbmp=0x1   state=block

estg stp 16 pbmp=0x2  state=disable

estg stp 16 lbmp=0x1  state=learn

estg stp 17 pbmp=0x2  state=disable

estg stp 17 lbmp=0x1  state=forward

//查看配置是否和预期一样

estg show
```

测试 4：

port0 配置 SMAC、DMAC 默认值开始，每次递增 1，vlan 从 100 到依次递增到 139，burst 模式发送 40 条，查看 port1-3 的收包情况。

期望结果：port0 发送 40 条报文，port1 收到 5 条，port2 和 3 一起收到 5 条。

测试 5：

Port1 配置 SMAC、DMAC 和测试 1 互换，vlan 从 100 到依次递增到 139，burst 模式发送 40 条，查看 port0，port2-3 的收包情况。

期望结果：port0 收到 40 条报文 port2 和 3 一起收到 0 条。

测试 6：

Port2 配置 SMAC、DMAC 和测试 1 互换，vlan 从 100 到依次递增到 139，burst 模式发送 40 条，查看 port0，port1 的收包情况。

期望结果：port0 收到 40 条报文，port1 到 0 条。

8 Erps 功能

erps 功能和 Stp 类似，也是分为 ierps 和 eerps 两部分，具体指令如下，“[ ]”表示可选，“|”表示多选 1。

8.1 Erps 指令介绍

8.1.1 ierps 指令说明

/*ierps 初始化，初始化之后，会创建默认 ierpsid=0，包含 vlan1-vlan4095，ierps0 的所有 port 和 lag 口的 erps 状态都是 diable (端口 erps_check 使能之后才生效，使能之前是 forward 状态)。*/

ierps init

//ierps 功能关闭

ierps deinit

//对应端口的 erps check 使能

ierps enable pbmp=<pbmp>

//对应端口的 erps check 去使能

ierps disable pbmp=<pbmp>

//创建 ERPSs，可以创建多个 erps

ierps create [<id>] [...]

//取消 ERPSs 配置

ierps destroy <id> [...]

//查看已配置的 ERPS(s)，显示 erps 包含的 vlan 以及对应的 erps 状态

ierps show [<id>]

//添加 VLAN(s)到一个 ERPS

ierps add <id> <vlan_id> [...]

//从一个 ERPS 中移除 VLAN(s)

ierps remove <id> <vlan_id> [...]

//显示所有 erps 状态

ierps state

//显示某个 erps 状态

ierps state<id>

//设置某个 ERPS 的状态

ierps state<id> <pbmp=xx>[<lbmp=xx>] <state=disable|forward>

8.1.2 eerps 指令说明

eerps 指令和 ierps 类似，这里不再详细说明。

```
eerps init

eerps deinit

eerps enable pbmp=<pbmp>

eerps disable pbmp=<pbmp>

eerps create [<id>][...]

eerps destroy <id>[...]

eerps show [<id>]

eerps add <id> <vlan_id> [...]

eerps remove <id> <vlan_id> [...]

eerps state

eerps state<id>

eerps state<id> <pbmp=xx>[<lbmp=xx>] <state=disable|forward>
```

8.2 erps 配置实例

8.2.1 ierps 配置实例

创建 2 个 ierps，分别为 erps 2 和而 erps 3，配置 lag0 包含 port2 和 port3。

erps2 包含 vlan 100-104，配置 port0、lag0 为 disable 状态

erps3 包含 vlan 105-109，配置 port0、lag0 为 forward 状态

1. Lag 配置：

trunk init

//创建一个 lag 组（lagID 为 0），端口成员为 2、3， hashkey 为 DstIp 和 SrcMac。

trunk add Id=0 Rtag=0x2001 Pbmp=0xC

/*使能 lag 口功能，一般管理型芯片默认使能 Lag 功能，如果配置错误则可以通过该配置强制使能 lag 功能；非管理型不支持 lag 功能*/

global set type=0 value=0

2. Vlan 配置：

//创建 100-119 的 vlan 域，包含 port0-1 以及 lag0

```
vlan create 100 pbmp=0x3  lbmp=0x1

vlan create 101 pbmp=0x3  lbmp=0x1

vlan create 102 pbmp=0x3  lbmp=0x1

vlan create 103 pbmp=0x3  lbmp=0x1

vlan create 104 pbmp=0x3  lbmp=0x1

vlan create 105 pbmp=0x3  lbmp=0x1

vlan create 106 pbmp=0x3  lbmp=0x1

vlan create 107 pbmp=0x3  lbmp=0x1

vlan create 108 pbmp=0x3  lbmp=0x1

vlan create 109 pbmp=0x3  lbmp=0x1
```

//使用包的 vlan 信息

```
port attrset port=0 type=29 value=1

port attrset port=1 type=29 value=1

port attrset port=2 type=29 value=1

port attrset port=3 type=29 value=1
```

3. ierps 配置：

```
ierps init
```

//是能端口 0、2、3 erps 检查


```
Ierps enable pbmp=0xD

//创建 ierps 2 3

ierps create 2 3

//ierps 2 中添加 vlan 100-104

ierps add 2100 101 102 103 104

//ierps 3 中添加 vlan 105-109

ierps add 3105 106 107 108 109

//ierps 2 port0，lag0 erps 状态为 diable

ierps state 2 pbmp=0x1  lbmp=0x1 state=disable

//ierps 3 port0，lag0 erps 状态为 forward

ierps state 3 pbmp=0x1  lbmp=0x1 state=forward

//查看 ierps 配置

ierps show
```

测试 1：

port0 配置 SMAC、DMAC 默认值开始，每次递增 1，vlan 从 100 到依次递增到 109，burst 模式发送 10 条，查看 port1-3 的收包情况。

期望结果：port0 发送 10 条报文，port1 收到 5 条，port2 和 3 一起收到 5 条。

测试 2：

Port1 配置 SMAC、DMAC 和测试 1 互换，vlan 从 100 到依次递增到 109，burst 模式发送 10 条，查看 port0，port2-3 的收包情况。

期望结果：port0 收到 10 条报文，port2 和 3 一起收到 5 条。

8.2.2 eerps 配置实例

eerps 配置实例可参照 ierps，这里不再详细描述。

9 Lag 测试

支持 8 个 lag 组，lag ID 为 0~7。每个 lag 组最多支持 32 个端口，同一个端口只能属于一个 lag 组。

1. 创建 vlan 域，vlan 域端口为 lag 口

//创建 100 的 vlan 域，该 vlan 域中加入三个 lag 组，lagID 为 0、1、2

vlan create 100 lbmp=0x7

2. Lag 功能初始化

//使用 lag 功能时应首先初始化

trunk init

3. Lag 功能关闭

trunk deinit

4. 创建 lag 组

trunk add ld=<lagID> Rtag=<psc> Pbmp=<pBmp>

创建 lag 组并指定 lag 组中的端口成员和 hash key，默认的 hash 算法为 crc8。hash key 支持 L2、L3、L4 层部分域组合，如 SrcMac、DstMac、SrcIp、DstIp、L4SrcPort、L4DstPort 等。对应的 psc 的值如下：

//SrcMac=0x1	UpdCtag=0x80	Tos=0x8000
//DstMac=0x2	CtagVid=0x100	L4SrcPort=0x10000
//EthType=0x4	CtagCfi=0x200	L4DstPort=0x20000
//UpdStag=0x8	CtagCos=0x400	
//StagVid=0x10	SrcIp=0x1000	
//StagCfi=0x20	DstIp=0x2000	
//StagCos=0x40	Protocol=0x4000	

//当 UpdStag 置位时，StagVid/StagCfi/StagCos 表示用更新后的 stag 值作为 //hashkey；UpdStag 不置位时，StagVid/StagCfi/StagCos 表示用报文头部的 stag //值作为 hashkey。UpdCtag 同理

例：

/*创建一个 lag 组（lagID 为 2），端口成员为 1、2、3，hashkey 为 DstIp 和 SrcMac。如若发送 DstIp 和 SrcMac 随机的报文，则报文会在端口 1、2、3 中负载均衡输出。若发送 DstIp 和 SrcMac 固定的报文，则报文会从端口 1、2、3 中某个固定端口输出。*/

trunk add Id=2 Rtag=0x2001 Pbmp=0xE

5. 设置 hash key

trunk psc Id=<lagID> Rtag=<psc>

6. 设置 hash 算法

trunk alg Id=<lagID> Hash=<alg>

hash 算法支持 crc8、crc16、crc32 等。alg 取值可为：crc8=0，crc32Lo=1，crc32Hi=2，crc16Bs=3，crc16cc=4，xor16=5，crc16ccHiXor8=6，crc16ccHiXor4=7，crc16ccHiXor2=8，crc16ccHiXor1=9

例：

//设置 lag 组 3 的 hash 算法为 crc16Bs

trunk alg Id=3 Hash=3

7. 设置 lag 组成员失效分担

trunk failover Id=<lagID> Able=<1|0>

例：

/*开启 lag 组 2 的失效分担功能，当 lag 组 2 中的端口成员 link 状态发生改变，如 up 变为 down 时，则该端口的流量会分担到 lag 组中其他端口输出。当端口 link 状态从 down 变为 up 时，则流量会恢复从该端口输出。*/

trunk failover Id=2 Able=1

8. 销毁 trunk 组

trunk destroy Id=<trunk_id>

9. 显示 trunk 组信息

trunk show

10 MAC 地址学习

1. 业务配置

在端口 1（光口）编辑 1 条单播以太报文，MacSa 递增，格式为：MacDa、MacSa、Vlan（tpid=0x8100，cos=0，cfi=0,vid=0x100）。

2. 配置命令

默认 mac 地址学习功能是开启的，不需要相关配置。

3. 测试 1

MacSA 递增，step 模式从 MacSa0 开始发送 1000 个报文。

期望结果 1

端口 2~6 均收到来自端口 1 的 1000 个报文。

//组播复制功能。

4. 测试 2

在端口 2~6 中选择一个端口，编辑 macDa 从 MacSa0 开始递增的报文，格式为：MacDa、MacSa、Vlan（tpid=0x8100，cos=0，cfi=0,vid=0x100）；

期望结果 2

在端口 1 收到来自选中端口的 1000 个报文，其他端口无收包。

//学习到 1000 个 mac 地址。

11 MAC 地址学习限制

1. 业务配置

在端口 1（光口）编辑 1 条单播以太报文，MacSa 递增，格式为：MacDa、MacSa、Vlan（tpid=0x8100，cos=0，cfi=0,vid=0x100）。

2. 配置命令：

//设置地址学习限制数目为 500 个

L2 learnlimit global=1 limit=500

开启地址学习限制使能：

global set type=4 value=1

3. 测试 1

MacSA 递增，step 模式从 MacSa0 开始发送 1000 个报文。

期望结果 1

端口 2~6 均收到来自端口 1 的 1000 个报文。

//组播复制功能。

4. 测试 2

在端口 2~6 中选择一个端口，编辑 macDa 从 MacSa0 开始递增的报文，格式为：MacDa、MacSa、Vlan（tpid=0x8100，cos=0，cfi=0,vid=0x100）；

期望结果 2：

在端口 1 收到来自选中端口的 1000 个报文，其中 500 个为单播转发过来，另外 500 个为泛洪而来。其他端口收到因泛洪而来 500 个报文。

//学习到 500 个 mac 地址，不在此范围的 MAC 地址泛洪。

12 MAC 地址老化

1. 业务配置

在端口 1（光口）编辑 1 条单播以太报文，MacSa 递增，格式为：MacDa、MacSa、Vlan（tpid=0x8100，cos=0，cfi=0,vid=0x100）。

2. 配置命令

//设置老化时间为 60s，单位为秒

age 60

3. 测试 1

MacSA 递增，step 模式从 MacSa0 开始发送 1000 个报文。

期望结果 1

端口 2~6 均收到来自端口 1 的 1000 个报文。

//组播复制功能

4. 测试 2

在端口 2~6 中选择一个端口，编辑 macDa 从 MacSa0 开始递增的报文，格式为：MacDa、MacSa、Vlan（tpid=0x8100，cos=0，cfi=0,vid=0x100）；

期望结果 2

在端口 1 收到来自选中端口的 1000 个报文，其他端口无收包。

//学习到 1000 个 mac 地址。

5. 测试 3

等待 60s 以上，重复发包 2。

期望结果 3

由于地址已老化，根据 MacDa 找不到出端口，所有端口均收到 1000 个报文。

额外相关命令：

端口老化使能开关：

age penable port=<port_id> value=<0|1>

// port 端口号

//value 1 开启 0 关闭，默认所有端口老化使能开启

基于匹配规则快速老化

age rule Port=<port_id> MACaddress=<0x> Vlanid=<> TrunkGroupID=<> value=<0|1>

上面命令分别可以基于端口号，mac 地址，vlan id, trunk id 进行快速老化，value 0 表示关闭匹配规则，value 1 表示开启规则。

13 Acl、eAcl 及 vfp

芯片包括多个流识别引擎，分别为 ACL、EACL 和 vfp，最大条目数分别为 512、256 和 128。这些引擎都是采用 TCAM 查表方式。

ACL 的条目数为 512 条。可以根据配置划分为 MacKey、IPv4Key、IPv6Key 和 MixKey，每张表存储不同类型的 Key 值。当 ACL 表全部存储完整的 MacKey 或 IPv4Key 条目时，Key 值的最大宽度会达到 320 比特，此时条目数减半为 256 条。当 ACL 表全部存储完整的 IPv6Key 或 MixKey 条目时，Key 值的最大宽度会达到 640 比特，此时条目数减为四分之一为 128 条。当 Key 的位宽为 160 比特时，条目数达到最多为 512 条。 ACL 包括两个流查找引擎 ACL0 和 ACL1，共享所有的 ACL 查找条目。

ACL 可以基于端口、L2、L3、L4 的内容进行流分类。其中针对不同的输入包类型，可以有不同的 Key 组合，各种 Key 包含的具体内容如表 13-1 所示。

表 13-1 ACL_KEY

Key 类型	内容
MacKey	DMAC[47:0],SMAC[47:0],STAG[15:0],CTAG[15:0], ETHERTYPE[15:0],PORTS[34:0],L3UDF[7:0]等
IPv4Key	DIP[31:0],SIP[31:0],PROTOCOL[7:0],TOS[7:0],PORTS[34:0], L4SrcPort[15:0],L4DsrPort[15:0],L3UDF[7:0],L4UDF[7:0]等
IPv6Key	DIP[127:0],SIP[127:0],PROTOCOL[7:0],TOS[7:0],PORTS[34:0],L4SrcPort[15:0],L4DsrPort[15:0], L3UDF[7:0],L4UDF[7:0],TTL[7:0],DMAC[47:0],TCPFLAG[7:0],STAG[15:0],CTAG[15:0]等
MixKey	DIP[127:0],SIP[127:0],PROTOCOL[7:0],TOS[7:0],PORTS[34:0],L4SrcPort[15:0],L4DsrPort[15:0], L3UDF[7:0],L4UDF[7:0],TTL[7:0],DMAC[47:0],SMAC[47:0],ICMPCODE[7:0],ICMPTYPE[7:0],TTL [7:0],TCPFLAG[7:0],STAG[15:0],CTAG[15:0]、 IPOPTION[0:0],IPHDRERR[0:0],L3TP[3:0],L4TP[2:0]等

默认情况下 IPv4 和 Arp 报文查找 IPv4Key，IPv6 报文查找 IPv6Key，其他报文查找 MacKey。也可以基于端口来控制是否强制采用 MacKey、IPv4Key、IPv6Key 或者 MixKey 进行 ACL 查找。ACL0 和 ACL1 可以配置不同的 Key 类型查找。

根据数据报文的不同类型和配置选择 ACL0 和 ACL1 两个流引擎的查找 Key，其中 ACL0 和 ACL1 的查找 Key 分别包含不同的 ACL 模板。两个 ACL 查找引擎会获取该业务流的两条处理行为，此时需要进行仲裁，仲裁原则是如果两条处理行为项之间存在冲突，则以 ACL1 查找引擎对应的处理行为为准。

EACL 的查找方式以及 Key 同 ACL 基本一致。

Vfp 根据配置可以划分为 VlanKey、MacKey、IPv4Key 和 IPv6Key，每张表存储不同类型的 Key 值。各种 Key 包含的具体内容如表 13-2 所示。

表 13-2 VFP_KEY

Key 类型	内容
VlanKey	SMAC[47:0],STAG[15:0],CTAG[15:0],SIP[17:0],L2Tp,L3Tp,L4Tp ETHERTYPE[15:0],GPORT 等
MacKey	SMAC[47:0],DMAC[47:0],STAG[15:0],CTAG[15:0],PORTS[34:0],

	ETHERTYPE[15:0],L3UDF[7:0],L4UDF[7:0],L2Tp,L3Tp,L4Tp 等
IPv4Key	DIP[31:0],SIP[31:0],PROTOCOL[7:0],TOS[7:0],PORTS[34:0],L4SrcPort[15:0],L4DsrPort[15:0],L3UDF[7:0],L4UDF[7:0],TTL[7:0],TCPFLAG[7:0],ipLen,ipOption,L3Tp,L4Tp 等
IPv6Key	DIP[127:0],SIP[127:0],PROTOCOL[7:0],TOS[7:0],PORTS[34:0],L4SrcPort[15:0],L4DsrPort[15:0],L3UDF[7:0],L4UDF[7:0],TTL[7:0],DMAC[47:0],ICMPCODE[7:0],ICMPTYPE[7:0],TCPFLAG[7:0],STAG[15:0],CTAG[15:0],ipOption,ipHdrErr,L3TP[3:0],L4TP[2:0]等

默认情况下 IPv4 和 Arp 报文查找 IPv4Key，IPv6 报文查找 IPv6Key，其他报文查找 MacKey。也可以基于端口来控制是否强制采用 VlanKey、MacKey、IPv4Key、IPv6Key 进行查找。

13.1 创建相关命令

1. 基于端口使能 ACL/EACL/VFP 查询开关

//port：端口 ID

//value：1 表示开启，0 表示关闭

/*type：60 表示 ACL0 查找，61 表示 ACL1 查找；90 表示 EACL0 查找，91 表示 EACL1 查找；2 表示 VFP 查找*/

port attrset port={portId} **value=**{0|1} **type=**{val}

2. 基于端口配置强制使用某种 Key 查询 ACL 或 EACL

```
//"{}"表示必选，"[]"表示可选，"|"表示二选一或多选一

fp forcekey port={portId} [ipv4ForceMacKey0=0|1] [ipv4ForceIpv6Key0=0|1]

[ipv4ForceMixKey0=0|1] [ipv6ForceMacKey0=0|1] [ipv6ForceIpv4Key0=0|1]

[ipv6ForceMixKey0=0|1] [macForceIpv6Key0=0|1] [macForceIpv4Key0=0|1]

[macForceMixKey0=0|1] [ipv4ForceMacKey1=0|1] [ipv4ForceIpv6Key1=0|1]

[ipv4ForceMixKey0=1|1] [ipv6ForceMacKey1=0|1] [ipv6ForceIpv4Key1=0|1]

[ipv6ForceMixKey1=0|1] [macForceIpv6Key1=0|1] [macForceIpv4Key1=0|1]

[macForceMixKey1=0|1]
```

3. 基于端口配置强制使用某种 Key 查询 VFP

```
//"{}"表示必选，"[]"表示可选，"|"表示二选一或多选一

fp forcekey port={portId} [useVlanKey=0|1]

[ipv4ForceMacKey=0|1][ipv4ForceIpv6Key=0|1]
```


[ipv6ForceMacKey=0|1] [ipv6ForceIpv4Key=0|1]

[macForceIpv4Key0=0|1][macForceIpv6Key=0|1]

4. 初始化，应首先被调用一次

fp init

5. 清除 ACL、EACL、VFP 指示配置

fp qset clear

6. 指示当前为 ACL、EACL 或 VFP 模块

//StageIngress 表示 ACL

// StageEgress 表示 EACL

//StageLookup 表示 VFP

fp qset add {StageIngress | StageEgress | StageLookup}

7. 创建组

/*

*创建 Group

*pri：未使用，固定填 1

*gid：组 ID，全局唯一

*keyTp：0(MacKey) 、1(IPv4Key)、 2(IPv6Key)、 3(MixKey)

*size：表示当前分配给该 Key 类型的 entry 条目数，值为 4 的整数倍

*mode：0(Key 最小位宽)、1(两条目拼成 Key--2 倍最小位宽)、3(四条目拼成 Key)

*/

fp group create {pri} {gid} {keyTp} {size} {mode}

8. 创建 entry

/*

*创建 Entry。Entry ID 指定优先级，ID 越小优先级越高。各个模块 entryID 范围分别如下：iAcl：1~512；eAcl：513~768； vfp：769~896；evfp：897~928

*gid：组 ID，全局唯一

*eid：entry ID，全局唯一

*/

fp entry create {gid} {eid}

9. 配置匹配域

```
/*
*eid: entry ID
*field: 匹配域。包含 SrcIp,DstIp,SrcIp6,DstIp6,IpProtocol,L4SrcPort,L4DstPort,
*Inports,InportL2Tp,L3Tp,L4Tp,L5Tp,L4SrcPortRng,L4DstPortRng,Color,
*OuterVlan,InnerVlan,Ttl,TcpControlSrcMac,DstMac,EtherType,SrcIpRng,
*DstIpRng,SrcIp6Rng,DstIp6Rng 等
*data: key 值
*mask: keyMask
*/
fp qual {eid} {field} {data} {mask}
```

10. 匹配后行为

```
/*
*eid: entryID，全局唯一
*action: 匹配后行为。
*支持 RedirectPort(重定向到端口，需指定端口 ID),
*RedirectTrunk(重定向到 trunk 组，需指定 trunk 组 ID),
*RedirectMcast(重定向到组播组，需指定组播组 ID),
*UpdateCounter(开启命中计数统计),Drop(丢弃),Permit(转发),Trap(trap 到指定端口),
*OuterVlanNew(修改或增加外层 vlan,需指定 vlanID)， OuterVlanDelete(删除外层 vlan)
*OuterVlanPrioNew(修改外层 vlan pri,需指定 vlanID)
```


*InnerVlanNew(修改或增加内层 vlan,需指定 vlanID)， InnerVlanDelete(删除内层 vlan)

*InnerVlanPrioNew(修改内层 vlan pri,需指定 vlanID)等

*VFP 支持 Drop,Trap,PrioIntNew(指定内部优先级,需指定优先级值)， DoNotLearn(不学习),

*DtOuterVlanCopyInner(双 Tag 报文,复制内层 vid 到外层 vid),

*DtOuterVlanPrioCopyInner(双 Tag 报文,复制内层 pri 到外层 pri),

*DtOuterVlanCfiCopyInner,DtInnerVlanCopyOuter,DtInnerVlanPrioCopyOuter,DtInnerVlanCfi CopyOuter,

*SotInnerVlanCopyOuter(单外层报文,复制外层 vid 到内层 vid),

*SotInnerVlanPrioCopyOuter,SotInnerVlanCfiCopyOuter,

*SitOuterVlanCopyInner(单内层报文,复制内层 vid 到外层 vid),

*SitOuterVlanPrioCopyInner,SitOuterVlanCfiCopyInner,

*DtOuterVlanNew <vid>(双 Tag 报文,修改外层 vid,需指定 vid 参数),

*DtOuterVlanPrioNew<pri>,DtOuterVlanCfiNew <cif>,

*DtInnerVlanNew,DtInnerVlanPrioNew,DtInnerVlanCfiNew,

*SotOuterVlanNew,SotOuterVlanPrioNew,SotOuterVlanCfiNew,

*SitInnerVlanNew,SitInnerVlanPrioNew,SitInnerVlanCfiNew,

*SotInnerVlanAdd,SotInnerVlanPrioAdd,SotInnerVlanCfiAdd,

*SitOuterVlanAdd(单内层报文添加外层 vlan,,需指定 vid),

*SitOuterVlanPrioAdd,SitOuterVlanCfiAdd,

*UtInnerVlanAdd(不带 Tag 报文,添加内层 vlan,需指定 vid),

*UtInnerVlanPrioAdd,UtInnerVlanCfiAdd,

*UtOuterVlanAdd,UtOuterVlanPrioAdd,UtOuterVlanCfiAdd,

*DtOuterVlanDelete,DtInnerVlanDelete,SotOuterVlanDelete,SitInnerVlanDelete

*/

fp action add {eid} {action} [val]

11. 下发硬件表项

fp entry install {eid}

12. 其他命令可通过 fp help 查看

13.2 删除相关命令

//反初始化

fp detach

//指定 eid 时仅删除对应的 entry，不带参数删除所有 entry

fp entry destroy [eid]

//指定 gid 时仅删除对应的 group，不带参数删除所有 group

fp group destroy [gid]

13.3 显示相关命令

//显示 fp 相关配置

fp show [group|entry] [id]

//显示所有支持的 quals 和 actions

fp list actions|quals [ifp|efp|vfp]

其他命令可通过 **fp help** 查看

13.4 命令行示例

//开启端口 1 的 ACL0 查询功能

port attrset port=1 value=1 type=60

fp init

fp qset clear

//指示为当前为 iACL 模块配置

fp qset add StageIngress

//二层报文强制查询 ipv4key

fp forcekey port=1 macForceIpv4Key0=1

/*创建 group1 为 IPv4Key，entry 条目数为 40 条(entryID 为 1~40)，模式为 2 拼 1(两条目拼成一个 Key, 实际占用 ACL 条目数为：40*2=100 条*/

fp group create 1 1 1 40 1

//基于 group1，创建 entry1

fp entry create 1 1

//匹配报文五元组

fp qual 1 DstIp 0x12345678 0xffffffff

fp qual 1 SrcIp 0x87654321 0xffffff00

fp qual 1 IpProtocol 0x33 0xff

fp qual 1 L4SrcPort 0x1244 0xffff

fp qual 1 L4DstPort 0x6789 0xffff

//匹配入端口 0、1、2 的报文(此时掩码为 key 值取反)

fp qual 1 inPorts 0x7 0xfffffff8

//行为为重定向到端口 31(cpu 口)

fp action add 1 redirectport 31

//统计命中计数

fp action add 1 UpdateCounter

//下发硬件表项

fp entry install 1

//获取 ACL 命中计数统计

fp counter get 1

//基于 group1，创建 entry2

fp entry create 1 2

//匹配 tclflag

fp qual 2 TcpControl 0x12 0xff

fp action add 2 OuterVlanNew 1000

fp action add 2 InnerVlanNew 1001

fp entry install 2

...

/*创建 group2 为 IPv6Key，entry 条目数为 60 条(entryID 为 41~100)，模式为 4 拼 1(四条目拼成一个 Key)，实际占用 ACL 条目数为：60*4=240 条*/

fp group create 1 2 2 60 3

//基于 group2，创建 entry41

fp entry create 2 41

fp qual 41 SrcIp6 5566:6677:7788:8899:1122:2233:3344:4455 ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff

fp qual 41 DstMac 12:34:56:77:90:32 ff:ff:ff:ff:ff:00

//行为丢弃

fp action add 41 drop

//下发硬件表项

fp entry install 41

/*创建 group3 为 MixKey，entry 条目数为 40 条(entryID 为 101~140)，模式为 4 拼 1(四条目拼成一个 Key)，实际占用 ACL 条目数为：40*4=160 条*/

fp group create 1 3 3 40 3

//基于 group3，创建 entry110

fp entry create 3 110

//匹配 ipv6 报文的 dip

fp qual 110 DstIp6 5566:6677:7788:8899:1122:2233:3344:4455 ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff

如想匹配 ipv4 报文的 dip，可配置如下：

fp qual 110 DstIp6 0:0:0:0:0:0:3344:4455 ffff:ffff:ffff:ffff:ffff:ffff:ffff:ff00

(低 32 比特表示 ipv4，高 96 比特 key 为 0，掩码为全 1，同时可增加匹配域

fp qual 110 EtherType 0x0800 0xffff)

//匹配报文 smac

fp qual 110 SrcMac 12:34:56:77:90:32 ff:ff:ff:ff:ff:00

//行为为 trap，trap 口由其他命令配置（如 trap 口为 cpu 口，dp trap set port=31）


```
fp action add 110 trap


//下发硬件表项
fp entry install 110
```


14 镜像

本芯片支持输入/输出端口、输入/输出转发 VLANID、MAC 地址以及输入/输出业务流的镜像功能。输入/输出镜像端口数各为 1 个。

14.1 输入/输出端口、输入/输出转发 VLANID 镜像

14.1.1 创建镜像模式

//"{}"表示必选，"[]"表示可选，"|"表示二选一或多选一

//PortIngress：基于输入端口镜像；

//PortEgress：基于输出端口镜像

//VlanIngress：基于入口转发 VLANID 镜像；

//VlanEgress：基于出口转发 VLANID 镜像

//src_port：被镜像端口号。当 Mode 为 PortIngress 或 PortEgress 时有效

//vlan_id：转发 VLANID。当 Mode 为 VlanIngress 或 VlanEgress 时有效

mirror create Mode={PortIngress | PortEgress | VlanIngress | VlanEgress}

{ SrcPort=src_port **| Vlan=**vlan_id **}**

14.1.2 删除镜像

mirror destroy Mode={PortIngress | PortEgress | VlanIngress | VlanEgress}

{ SrcPort=src_port **| Vlan=**vlan_id **}**

14.1.3 配置输入镜像端口

mirror IDestPort {dst_port}

14.1.4 配置输出镜像端口

mirror EDestPort {dst_port}

示例：

/*基于入端口 1 做输入镜像，镜像口为 2 口则端口 1 收到的流量会从端口 2 镜像转出*/

mirror create Mode=PortIngress SrcPort=1

mirror IDestPort 2

14.1.5 显示镜像配置

mirror show [port | vlan]

14.2 MAC 地址镜像

通过查找 MAC 地址表，对匹配的源 MAC 地址和目的 MAC 地址进行镜像操作。

14.2.1 流镜像

在 ACL 和 EACL 模块中可以基于流设置采样或镜像功能。

15 流量控制

15.1 命令行介绍

端口流控功能开启指令如下：

/*通过使能命令可以控制流控开启和关闭*/

port pause set enable port=<port_id> [TxEn=<1|0>] [RxEn=<1|0>]

/*当通过多个端口向一个端口打流量时，端口 Sport 阈值之和会大于 Dport 阈值，从而在流控功能生效前就会导致丢包。因此，需要对流控端口的 Sport 阈值进行修改，确保 Sport 阈值之和小于 Dport，同时 thdon 值要大于 thdoff 值。*/

```
port pause get threshold[enable] port=<port_id>

port pause set threshold port=<port_id> [thdon=<xx>] [thdoff=<xx>]
```

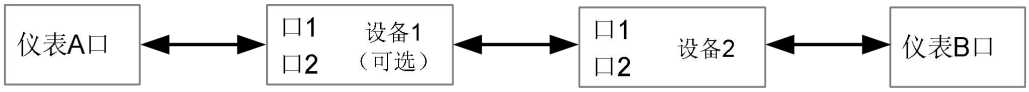
15.2 配置实例

15.2.1 实例 1

1. 业务配置

将两台设备串联，分别如前面章节所述打通业务通道，流量由仪表 A 口发往仪表 B 口,1G 流量：

假设连接如下图：



数据流

2. 设备 2 配置

port pause set enable port=0 TxEn=1

设备 1（可选）配置：

port pause set enable port=1 RxEn=1

port pause set enable port=0 TxEn=1

整形对设备 2 的口 1 进行整形限速：

dp shape port set Port=1 Mode=0 FillRate=10000 BurstSize=0xffff Quantum=2

3. 期望结果

仪表 B 口接收速率 10000 kbps，同时仪表 A 口可以接收到 pause 帧，如果仪表 A 支持流控，则发送速率会降低为 10000 kbps；当限速取消，或者将整形速率设置为大于入口速率，仪表 A 口会接收到 pause stop 帧。

15.2.2 实例 2

1. 业务配置

如图所示，分别如前面章节所述打通业务通道，流量由仪表 A 口和 B 口发往仪表 C 口，各 1G 流量：

数据流

2. 流控命令配置

由于端口默认 Sport 流控开启阈值 600，关闭阈值 400，Dport 绿包缓存门限 800。此时可以设置端口 0 和 1 的 Sport thdon=300，thdoff=200，同事开启端口 0 和 1 的流控功能。配置指令如下：

```
port pause set enable port=0 TxEn=1

port pause set enable port=1 TxEn=1

port pause set threshold port=0 thdon=300 thdoff=200

port pause set threshold port=1 thdon=300 thdoff=200
```

3. 期望结果

仪表 C 口接收速率 1Gbps，同时仪表 A 口和 B 口可以接收到 pause 帧，如果仪表支持流控，则 A 口和 B 口发送速率会降低为 500 Mbps；当限速取消，或者将 A 口和 B 口速率之和小于 1Gbps，仪表 A 口和 B 口会接收到 pause stop 帧。

16 TM 测试

16.1 SP Mode（优先级模式）

1. 业务配置

在端口 1（光口）编辑 8 条单播以太报文，格式为：MacDa、MacSa、Vlan（tpid=0x8100，cos=0~7，cfi=0，vid=0x200）。

2. 配置命令

```
vlan create 0x200 pbmp=0x7E

port attrset port=1 value=0 type=10

vlan vlanset vlanid=0x200 value=1 type=24

vlan vlanset vlanid=0x200 value=10 type=25

policing map profile create QosProIndex=10 UseL2Info=1 TrustCtag=0 PhbPtr=0

//vlan 优先级到内部优先级的映射

policing map vlan pri map set QosProIndex=10 PktPri=0 InternalPri=0 Color=2

policing map vlan pri map set QosProIndex=10 PktPri=1 InternalPri=1 Color=2

policing map vlan pri map set QosProIndex=10 PktPri=2 InternalPri=2 Color=2

policing map vlan pri map set QosProIndex=10 PktPri=3 InternalPri=3 Color=2

policing map vlan pri map set QosProIndex=10 PktPri=4 InternalPri=4 Color=2

policing map vlan pri map set QosProIndex=10 PktPri=5 InternalPri=5 Color=2

policing map vlan pri map set QosProIndex=10 PktPri=6 InternalPri=6 Color=2

policing map vlan pri map set QosProIndex=10 PktPri=7 InternalPri=7 Color=2

//内部优先级到队列的映射

dp port queue map set Port=2 Pri=0 Queue=0

dp port queue map set Port=2 Pri=1 Queue=1

dp port queue map set Port=2 Pri=2 Queue=2

dp port queue map set Port=2 Pri=3 Queue=3

dp port queue map set Port=2 Pri=4 Queue=4
```



```
dp port queue map set Port=2 Pri=5 Queue=5

dp port queue map set Port=2 Pri=6 Queue=6

dp port queue map set Port=2 Pri=7 Queue=7

//端口限速

dp shape port set Port=2 Mode=0 FillRate=rate1 BurstSize=0xffff Quantum=2

//关闭基于端口的准入控制，开启基于队列的准入控制

dp admin dport set port=2 enable=0 coloraware=1

dp admin global queue set enable=1
```

3. 期望结果

在端口限速为 rate1 kbps，8 条流流量总和为 rate2 kbps 情况下，若 rate1 < rate2，则高优先级报文优先通过，高优先级不丢包，低优先级丢包或者完全收不到报文（流量分配只分到一部分或者完全分不到）。

16.2 WFQ Mode（按权重分配模式）

1. 业务配置

在端口 1（光口）编辑 8 条单播以太报文，格式为：MacDa、MacSa、Vlan（tpid=0x8100，cos=0~7，cfi=0，vid=0x200）。

2. 配置命令

```
vlan create 0x200 pbmp=0x7E

port attrset port=1 value=0 type=10

vlan vlanset vlanid=0x200 value=1 type=24

vlan vlanset vlanid=0x200 value=10 type=25

policing map profile create QosProIndex=10 UseL2Info=1 TrustCtag=0 PhbPtr=0

policing map vlan pri map set QosProIndex=10 PktPri=0 InternalPri=0 Color=2

policing map vlan pri map set QosProIndex=10 PktPri=1 InternalPri=1 Color=2

policing map vlan pri map set QosProIndex=10 PktPri=2 InternalPri=2 Color=2

policing map vlan pri map set QosProIndex=10 PktPri=3 InternalPri=3 Color=2

policing map vlan pri map set QosProIndex=10 PktPri=4 InternalPri=4 Color=2
```



```
policing map vlan pri map set QosProIndex=10 PktPri=5 InternalPri=5 Color=2

policing map vlan pri map set QosProIndex=10 PktPri=6 InternalPri=6 Color=2

policing map vlan pri map set QosProIndex=10 PktPri=7 InternalPri=7 Color=2

dp port queue map set Port=2 Pri=0 Queue=0

dp port queue map set Port=2 Pri=1 Queue=1

dp port queue map set Port=2 Pri=2 Queue=2

dp port queue map set Port=2 Pri=3 Queue=3

dp port queue map set Port=2 Pri=4 Queue=4

dp port queue map set Port=2 Pri=5 Queue=5

dp port queue map set Port=2 Pri=6 Queue=6

dp port queue map set Port=2 Pri=7 Queue=7

dp shape port set Port=2 Mode=0 FillRate=rate1 BurstSize=0xffff Quantum=2

dp admin dport set port=2 enable=0 coloraware=1

dp admin global queue set enable=1

dp schedule queue set Port=2 pri0Weight=1 pri1Weight=2 pri2Weight=3 pri3Weight=4
pri4Weight=5 pri5Weight=6 pri6Weight=7 pri7Weight=8 wrrquantum=2 wrrpri=3 schmode=1
schbmp=0xFF
```

3. 期望结果

在端口限速为 rate1　kbps，8 条流流量总和为 rate2　kbps 情况下，若 rate1 < rate2，则流量按照 weight1：weight2：…：weight8 的比例分配。

17 风暴抑制

支持基于全局、端口和 fid 三种类型的风暴抑制，下面以基于端口的风暴抑制为例介绍其配置命令行。

在端口 2 的未知组播报文进行风暴抑制，配置如下：

1. 设置帧间隔和前导码的等效包长、全局配置

policing storm control global set PreambleLen=20

2. 基于端口的风暴控制的令牌更新设置：更新使能，最大更新条目数为 187，mode=1 为基于端口模式

policing storm control update set Mode=1 UpdEn=1 MaxUpdIdx=187 DelayInterval=10000

3. 使能未知组播的风暴控制功能，Index=2 表示端口为 2，FwdType=2 表示未知组播（0：已知单播和未知单播，1：已知组播，2：未知组播，3：广播）

policing storm control enable set Mode=1 Index=2 FwdType=2 Enable=1

4. 风暴控制的具体限速配置设置，PolType=0 表示基于字节限速，Limit 为限速值（kb 为单位），具体的限速值和时钟频率相关，BurstSize 为桶深（kb 为单位）

policing storm control config set Mode=1 index=2 FwdType=2 PolType=0 Limit=500000 BurstSize=10000000

注意：BurstSize 不能设置得太小，最好在 50 倍 Limit 之上。

18 PKT DMA 功能

注：该章仅适用于 FSL91030M。 PKT DMA 用于 cpu 收发交换芯片面板口（非调试网口）的包，该指令主要用于调试。调试应用层收发包功能，设置过滤器（仅针对非调试网口进来的包）功能。

18.1 functest pktdma 指令介绍

functest

functest 是一个测试 sdk 功能的命令。pktdma 是 functest 的一个子命令。该子命令下有 10 个子功能（mode=1~10）。

用如下命令查看用法：

functest pktdma mode=0

18.2 functest pktdma 配置实例

18.2.1 应用层收发包测试配置实例

配置面板口 port3 的包转发到 cpu，并且包类型为 0x88f7 的包不经过协议栈，直接上传到应用层，应用层将接收到的报文打印出来。应用层收到包后再原路径转发出来，然后打印出发送的报文。

收发包通道配置：

1. 复位交换芯片

```
modreg TOP_CFG_REG_RESET_GLOBAL UPI_RST_GLB_LOGIC_N=0 UPI_RST_GLB_UPI_N=0

modreg TOP_CFG_REG_RESET_GLOBAL UPI_RST_GLB_LOGIC_N=1 UPI_RST_GLB_UPI_N=1

modreg PKT_DMA_REG_PKT_DMA_AXI_RD_CFG UPI_AXI4_RLEN_MAX=0xf

modreg PKT_DMA_REG_PKT_DMA_AXI_WR_CFG UPI_AXI4_WLEN_MAX=0xf
```

2. 设置上送 CPU 通道：进 port3 的包转发到 cpu 口（port31）

```
modify I_VT_PORT_SRM 3 1 FWD_VLD=1 OUT_LPORT=31

modify I_NET_PORT_SRM 3 1 STP_CHK_EN=0 BRG_EN=0
```

3. 设置 CPU 下发通道：从 cpu 口发送出来的包不经过 pp

```
modreg E_ACL_LOOP_CTL LOOP_BYPASS=0x80

modreg E_EE_LOOP_CTL LOOP_BYPASS=0x80
```



```
modreg E_PF_LOOP_CTL LOOP_BYPASS0=0x80 LOOP_BYPASS1=0x80

modreg E_POL_LOOP_CTL LOOP_BYPASS=0x80

modreg I_ACL_LOOP_CTL LOOP_BYPASS=0xbe

modreg I_FWD_LOOP_CTL LOOP_BYPASS0=0xbe LOOP_BYPASS1=0xbe LOOP_BYPASS2=0xbe

modreg I_NET_LOOP_CTL LOOP_BYPASS0=0xbe LOOP_BYPASS1=0xbe

modreg I_POL_LOOP_CTL LOOP_BYPASS0=0xbe LOOP_BYPASS1=0xbe

modreg I_PR0_LOOP_CTL LOOP_BYPASS=0x80

modreg I_VT_LOOP_CTL LOOP_BYPASS0=0x3e LOOP_BYPASS1=0x3e

modreg E_DST_LOOP_CTL LOOP_BYPASS=0x0 LOOP_BYPASS1=0x0

modreg I_DST_LOOP_CTL LOOP_BYPASS0=0x0 LOOP_BYPASS1=0x0
```

4. 收发包过滤器

//dest=2，不经过协议栈，直接送到应用层

```
functest pktdma mode=7 type=0x88f7 dest=2
```

5. 应用层收发包计数清零

```
functest pktdma mode=2
```

6. 开启接收，并转发接收的包

//prtpkt 控制是否打印报文，bit0 是控制 rx 方向，bit1 控制 tx 方向。

//mode=4 是先接收，再转发到源端口，mode=3 是仅接收。

```
functest pktdma mode=4 prtpkt=3
```

7. 查看应用层收发包计数

```
functest pktdma mode=1
```

8. 停止接收

```
functest pktdma mode=6
```

18.2.2 协议栈收发包测试配置实例

配置面板口 port3 的包转发到 cpu，并且包 dmac 为 01:02:ff:1a:11:3e ，或类型为 0x0800 和 0x0806 的包上送内核协议栈。

收发包通道配置：

1. 复位交换芯片

```
modreg TOP_CFG_REG_RESET_GLOBAL UPI_RST_GLB_LOGIC_N=0 UPI_RST_GLB_UPI_N=0
modreg TOP_CFG_REG_RESET_GLOBAL UPI_RST_GLB_LOGIC_N=1 UPI_RST_GLB_UPI_N=1
modreg PKT_DMA_REG_PKT_DMA_AXI_RD_CFG UPI_AXI4_RLEN_MAX=0xf
modreg PKT_DMA_REG_PKT_DMA_AXI_WR_CFG UPI_AXI4_WLEN_MAX=0xf
```

2. 设置上送 CPU 通道：进 port3 的包转发到 cpu 口（port31）

```
modify I_VT_PORT_SRM 3 1 FWD_VLD=1 OUT_LPORT=31
modify I_NET_PORT_SRM 3 1 STP_CHK_EN=0 BRG_EN=0
```

3. 收发包过滤器

//dest=1，上送协议栈

```
functest pktdma mode=7 type=0x0800 dest=1
functest pktdma mode=7 type=0x0806 dest=1
functest pktdma mode=7 mac=01:02:ff:1a:11:3e dest=1
```

18.2.3 应用层发包测试配置实例

发包

/*设置 CPU 下发通道：从 cpu 口发送出来的包不经过 pp（发包时 ppbypass=0 无需配置）*/

```
modreg E_ACL_LOOP_CTL LOOP_BYPASS=0x80
modreg E_EE_LOOP_CTL LOOP_BYPASS=0x80
modreg E_PF_LOOP_CTL LOOP_BYPASS0=0x80 LOOP_BYPASS1=0x80
modreg E_POL_LOOP_CTL LOOP_BYPASS=0x80
modreg I_ACL_LOOP_CTL LOOP_BYPASS=0xbe
modreg I_FWD_LOOP_CTL LOOP_BYPASS0=0xbe LOOP_BYPASS1=0xbe LOOP_BYPASS2=0xbe
modreg I_NET_LOOP_CTL LOOP_BYPASS0=0xbe LOOP_BYPASS1=0xbe
modreg I_POL_LOOP_CTL LOOP_BYPASS0=0xbe LOOP_BYPASS1=0xbe
modreg I_PR0_LOOP_CTL LOOP_BYPASS=0x80
```

```
modreg I_VT_LOOP_CTL LOOP_BYPASS0=0x3e LOOP_BYPASS1=0x3e

modreg E_DST_LOOP_CTL LOOP_BYPASS=0x0 LOOP_BYPASS1=0x0

modreg I_DST_LOOP_CTL LOOP_BYPASS0=0x0 LOOP_BYPASS1=0x0

/*dmac 为 01:02:ff:1a:11:3e，type 为 0800，dest=3，dport 为 3，id=1，ppbypass 为 1*/

functest pktdma mode=5 mac=01:02:ff:1a:11:3e type=0x0800 dest=3 id=1
```


19 修订信息

修订时间	版本	描述
2021.5.8	V1.0	初始版本。
2022.12.22	V1.32	内容优化。
2022.4.25	V1.33	内容优化。