S17

Switch Software Development Kit

User Guide

Rev 1.4

1. Overview

The Switch Software Development Kit (SSDK) is a set of drivers to manage Atheros switch. It can be the foundation for customer's application to control the behaviors of the switch or as a reference to build customer's own low-level drivers.

The goals of the design are listed below:

1.1. Modularity and Hierarchy

The SSDK is based on a multi-layer architecture in which every layer has its own targets. This architecture ensures the SSDK can scale from small low-end system to future large multi-CPU and distributed system. Customers can select whether including every layer into their own applications or not by simply changing build options to meet their own requirements.

Meanwhile, the SSDK can be partitioned into separate function modules, thus they can only attach required functions into their systems to archive small-footprint implementations for cost sensitive systems.

1.2. Flexibility

For running on some UNIX-like Operation System in which operation of the CPU is divided into two distinct modes – Kernel mode and User mode, main modules of the SSDK can be executed in the kernel space or the user space by changing the related option on building. In this way, the

SSDK can meet different requirements of customers' various systems.

1.3. Hardware abstraction

The SSDK abstracts all hardware details of Atheros switch line by providing consistent APIs to customers, who do not have to understand the implementation and register details and can easily use APIs to build their own applications.

1.4. Portability

The SSDK is built upon a System Abstraction Layer (SAL), which abstracts difference between various OSs and CPU architectures and is easily to be ported to a new OS or CPU system. This can be achieved by adhering OS and Board Support Package (BSP) functions to some elaborated SAL APIs.

1.5. Robustness and Reliability

The SSDK ensures its production-quality from design phase to release phase. In addition, the SSDK handles possible error states and return a well-defined error code to reduce instability.

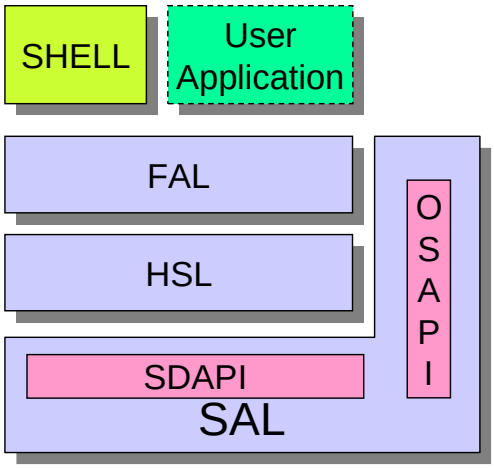
2. Architecture

This chapter gives an overview of the SSDK software architecture and supporting features.

2.1. Basic software architecture

The SSDK can be divided into three layers: FAL, HSL and SAL. Function Abstraction Layer (FAL) abstracts the implement details of Atheros switch chips by providing consistent Function APIs (FAPIs) to customers; Hardware Specific Layer (HSL) provides Hardware APIs (HAPIs) to implement functions of specific chip; System Abstraction Layer (SAL) that consists of OS abstraction APIs (OSAPI) and System Driver APIs (SDAPI) are used to abstract the OS and CPU system. In addition, the SSDK includes a CLI-like simple shell to manage the switch and to provide a reference of the usage of the SSDK to customers.

The basic software architecture can be illustrated in the below figure.



FAL is designed for providing unified APIs for Atheros switch line to user applications such as CLI, web and other protocols, for example, IGMP snooping, DHCP relay etc. Customers can build their applications through the well-defined APIs without the deep understanding of internal implementation and register details. Furthermore, because FAL abstracts the differences among Atheros switch products, when porting a system from one chip to another one, customers can build their applications with only slightly changes.

HSL as what implied by its name, is targeted for providing a set of APIs for specific Atheros switch chip, including for example, those for Garuda the name of the switch core for AR8316 and Athena the name for AR8216. HSL knows the internal structure of the switch and modifies the related registers directly by invoking SDAPIs in the SAL layer. User applications can invoke APIs provide by HSL directly as well. As the result, the SSDK can serve with impressive small foot-print by simply choosing related build options.

SAL consisting of OS abstraction APIs (OSAPI) and System Driver APIs (SDAPI) exists for portability where OSAPIs provide basic APIs such as memory, timer etc to abstract differences among common OSs as Linux, xBSD and vxWorks, and SDAPIs abstract the details of specific hardware platform by providing MDIO and future possible PCI functions for register access. By this mean, Efforts to porting to a new OS and hardware platform, which can be extremely relieved, are limited to only this layer, although at this time, the SSDK only provides supports for linux.

Furthermore, the SSDK includes a simple CLI-like SHELL on linux, which can be referenced as an example for customers' applications, to expedite experience of Atheros switch.

2.2. Basic software feature

In most cases, APIs of HSL could be a subset of those of FAL, so we only describe features of FAL layer here. Usually APIs in the same feature set are collected to a file with a meaningful suffix as fal_acl, fal_fdb, fal_igmp, fal_leaky, fal_mib, fal_mirror, fal_port_ctrl, fal_portvlan, fal_vlan, fal_qos, fal_rate, fal_stp, fal_misc, fal_cosmap, fal_ip, fal_nat, fal_trunk and fal_sec..

2.2.1. Port Control

Port control provides APIs for relate port control operations. It supports following functions:

- Set and get duplex/speed mode on one port
- Set and get ability/status of auto negotiation on one port
- Set and get Atheros header status on one port
- Set and get flow control status on one port
- Set and get power saving status on one port
- Set and get hibernate status on one port
- Run cable diagnostic test on one port

2.2.2. VLAN

Vlan provides APIs for operation vlan entry. It supports following functions:

- Write vlan entry in switch chip
- Create delete and update vlan entry in switch chip
- Next and find vlan entry in switch chip
- Flush all vlan entries in switch chip
- Add, deleted and update VLAN port member
- Set and get FID VLAN binded
- Set and get VLAN based source address auto learning status

2.2.3. Port VLAN

Port VLAN provides APIs for port-based VLAN feature and QinQ feature based on vlan translation entry. It supports following functions:

- Set and get 802.1q VLAN mode on one port
- Set and get egress VLAN mode on one port
- Add, delete, update and get port VLAN member on one port
- Set and get default VLAN id on one port
- Set and get force port default VLAN id status on one port
- Set and get force port-based VLAN status on one port
- Set and get nest VLAN status on one port
- Set and get tpid for nest VLAN on switch chip
- Set and get ingress VLAN mode mode on one port
- Set and get TLS status on one port
- Set and get priority propagation status on one port
- Set and get default s-vid on one port
- Set and get default c-vid on one port
- Set and get VLAN propagation status on one port

- Add, delete and get a VLAN translation entry on one port
- Set and get switch QinQ work mode on switch chip
- Set and get QinQ role on one port
- Set and get dot1q work mode on switch chip
- Iterate all VLAN translation entries on one port

The key points for user application to implement QinQ based on VLAN translation entry feature are below:
Set switch chip QinQ work mode, s-tag mode or c-tag mode.
Set switch port QinQ work role, core port or edge port.
Add one or some VLAN translation entries on one port.

2.2.4. FDB

FDB also called Address resolution lookup (ARL) Table or MAC Table by other vendors provides APIs for maintaining forwarding data base in switch chip. It supports the following functions:

- Add and delete fdb entry
- First, next and find fdb entry
- Set and get fdb dynamic learning status on one port
- Set and get fdb aging status on switch chip
- Set and get fdb aging timer on switch chip
- Iterate all fdb entries on switch chip
- Next fdb entries in extend mode
- First fdb entries in extend mode
- Set and get source address learning limit on one port
- Set and get forwarding command for learning limit exceed on one port
- Set and get source address learning limit
- Set and get forwarding command for learning limit exceed
- Add, delete, find and iterate reserved fdb entries

2.2.5. ACL

ACL provides APIs for define policy and releted actions for specific flow. It supports the following functions:

- Enable and disable ACL engine
- Create and destroy ACL list
- Bind and unbind ACL list on one port
- Add, delete and query ACL rules in a ACL list

The sequence for user applications to implement ACL should as below:

- Create an ACL list.
- Add acl rules to the created acl list.
- Bind the list to a port. The ACL will take effect only when it is bind to port(s)

2.2.6. QoS

QoS provides APIs for QoS feature. It supports following functions:

- Set and get the scheduling mode on switch chip
- Set and get port/queue transmission buffer status/number on one port
- Set and get dot1p tag to queue mapping entry on switch chip
- Set and get DSCP to queue mapping entry on switch chip
- Set and get QoS parameters for precedence of queue mapping modes on one port
- Set and get default dot1p tag on one port
- Set and get port receiving buffer number on one port
- Set and get the scheduling mode on one port
- Set and get port default stag priority
- Set and get port default ctag priority
- Set and get queue based QoS remark

2.2.7. IGMP/MLD

IGMP provides APIs for IGMP/MLD packets identification and hardware join/leave. It supports the following functions:

- Set and get for IGMP/MLD identification status on one port.
- Set and get IGMP/MLD packet forwarding method when enabling IGMP/MLD identification on switch chip.
- Set and get IGMP/MLD hardware join/leave status on one port.
- Set and get router port(s) for IGMP/MLD hardware join/leave on switch chip.
- Set and get the status for creating/deleting a multicast entry in FDB for hardware join/leave on switch chip.
- Set and get the static status of multicast entry which learned by hardware
- Set and get the leaky status of multicast entry which learned by hardware
- Set and get igmpv3/mldv2 packets hardware acknowledgement status on switch chip
- Set and get the queue status of multicast entry which learned by hardware
- Set and get multicast entry learn limit
- Set and get multicast entry learn limit exceed command

2.2.8. Leaky

Leaky provides APIs for leaky function. It supports the following functions:

- Set and get unicast/multicast packets leaky mode (port-based control or fdb-based control)
- Set and get unicast/multicast/arp packets leaky status on one port

2.2.9. Mirror

Mirror provides APIs for mirror feature. It supports following functions:

- Set and get analyzer port in switch chip for mirror
- Set and get ingress/egress mirror status for one port

2.2.10. Rate

Rate provides APIs for rate feature. It supports following functions:

- Set and get port ingress/port egress/queue egress rate limit status on one port
- Set and get packets type of storm control on one port
- Set and get the rate of storm control on one port
- Set and get the rate of storm control on one port
- Set and get port based policer
- Set and get ACL based policer
- Set and get port based shaper
- Set and get queue based shaper

2.2.11. STP

STP provides API for STP feature. It supports following functions:

- Set and get spanning tree state on one port

2.2.12. MIB

MIB provides APIs for getting MIB information from switch chip. It supports following functions:

- Get MIB information on one port
- Set and get mib status

2.2.13. LED

On some atheros switch serials the LED blinking mode can be controlled by setting control pattern. Usually the LEDs are divided into different groups such as WAN groups and LAN groups for each of which has one or several LEDs depending on by switch chips. Different LEDs in different groups may have different control pattern such as always ON, always OFF, always blinking and blinking depending on port speed, port duplex, etc.

The SSDK provides API to control led blinking pattern for each group. It supports following

functions:

- Set and get led control pattern on one led group.

2.2.14. Misc

Misc provides APIs for some miscellaneous features. It supports following functions:

- Set and get ARP status on switch chip
- Set and get forwarding command for unknown source address packets on one port
- Set and get supported max frame size on one chip
- Set and get destination ports for unknown unicast/multicast packets on switch chip
- Set and get CPU port status on switch chip
- Set and get status of PPPoE packets on switch chip
- Set and get status of DHCP packets on one port
- Set and get status of broadcast packets forwarding to CPU on switch chip
- Set and get arp packets forwarding command on switch chip
- Set and get eapol(802.1x) packets forwarding command on switch chip
- Add, delete and get a pppoe session entry on switch chip
- Set and get eapol(802.1x) packets hardware acknowledgement status on one port
- Set and get rip v1 packets hardware acknowledgement status on one port
- Set and get port based arp request packets hardware acknowledgement status on one port
- Set and get port based arp ack packets hardware acknowledgement status on one port
- Add, delete and get pppoe entries in extend mode

The pppoe session entry offer capability to forward ip multicast packets encapsulated in pppoe format to multi switch ports by switch hardware.

2.2.15. CoSMap

CoSMap provides APIs for cos mapping features. It supports following functions:

- Set and get DSCP to priority mapping
- Set and get DSCP to drop precedence mapping
- Set and get Dot1p to priority mapping
- Set and get Dot1P to drop precedence mapping
- Set and get priority to queue mapping
- Set and get priority to enhanced queue mapping

2.2.16. IP

IP provides APIs for IP features. It supports following functions:

- Add, delete, get and next host entry

- Bind counter and pppoe session to host entry
- Set and get port based ARP learn flag
- Set and get ARP learn mode
- Set and get IP/ARP source guard mode
- Set and get routing status
- Add, delete and next interface entry
- Set and get IP/ARP source unknown forwarding command
- Set and get IPv6 base address

2.2.17. NAT

NAT provides APIs for NAT/NAPT features. It supports following functions:

- Add, delete, get and next NAT entry
- Bind counter to NAT entry
- Add, delete, get and next NAPT entry
- Bind counter to NAPT entry
- Set and get NAT status
- Set and get NAPT status
- Set and get NAT hash flag
- Set and get NAPT hash mode
- Set and get NAT private base address
- Set and get NAT private base address mapping mode
- Add, delete and next public address
- Set and get unknown NAT session forwarding command

2.2.18. Trunk

Trunk provides APIs for trunk features. It supports following functions:

- Set and get trunk group port member information
- Set and get trunk hash mode

2.2.19. Sec

Sec provides APIs for Security features. It supports following functions:

- Set and get security check items information

2.2.20. Initialization

Initialization provides API for user applications to init the SSDK.

3. Building

This chapter gives an overview of directory architecture of SSDK and demonstrates how to build it on Linux.

3.1. Directory structure

[/] This is the root directory of the SSDK under which there are some files such as make files and targets files which defines the components of building.

[/src] All source code files are kept under this directory.

[/src/api] This directory contains source code files for all API interfaces description.

[/src/fal] This directory contains source code files for FAL. FAL can provide unified interfaces and wrap the difference among switch chips. Customers can access these interfaces and needn’t care internal details.

[/src/fal_uk] This directory contains an example for user applications that invoke the SSDK built in kernel mode. It can be built with user applications together.

[/src/hsl] This directory contains source code files for HSL that provides interfaces for a specific chip.

[/src/hsl/athena] This directory contains source code files for Athena the switch core name for AR8216.

[/src/hsl/garuda] This directory contains source code files for Garuda the switch core name for AR8316.

[/src/hsl/shiva] This directory contains source code files for Shiva the switch core name for AR8227/AR8228/AR8229.

[/src/hsl/horus] This directory contains header files for Horus the switch core name for AR8236.

[/src/hsl/isis] This directory contains header files for Horus the switch core name for ????.

[/src/shell] This directory contains source code files all source code files for the SHELL.

[/src/init] This directory contains source code files for the initialization of the SSDK.

[/src/sal] This directory contains source code files for SAL.

[/src/sal/os] This directory contains source code files for OSAPI.

[/src/sal/sd] This directory contains source code files for SDAPI.

[/src/util] This directory contains source code files for utility functions.

[/include] All header files for the SSDK are under this directory.

[/src/api] This directory contains header files for all API interfaces description

[/include/common] This directory contains public header files for the SSDK.

[/include/fal] This directory contains header files of FAL.

[/include/hsl] This directory contains header files of HSL.

[/include/hsl/athena] This directory contains header files for Athena the switch core name for AR8216.

[/include/hsl/garuda] This directory contains header files for Garuda the switch core name for AR8316.

[/include/hsl/shiva] This directory contains header files for Shiva the switch core name for AR8227/AR8228/AR8229.

[/include/hsl/horus] This directory contains header files for Horus the switch core name for AR8236.

[/include/hsl/isis] This directory contains header files for Horus the switch core name for ????

[/include/init] This directory contains header files for the initialization of the SSDK.

[/include/sal] This directory contains header files for SAL

[/include/sal/os] This directory contains header files for OSAPI.

[/include/sal/sd] This directory contains header files for SDAPI.

[/include/shell] This directory contains header files for SHELL.

3.2. How to Build

Main parts of the SSDK except the SHELL will be compiled or built into libraries by the Makefile. The library for kernel is named as “ssdk_ks_km.a” or “ssdk_ks_um.a” (It’s _km.a or _um.a depends on the building mode for SSDK) which can be linked to the kernel or kernel module. That for userland is built into ssdk_us_km.a or ssdk_us_um.a. The shell will be built into a user program with ssdk_us_km.a or ssdk_us_km.a together.
(Notes: ks means kernel space, us means user space, km means kernel module and um means user mmodule)

Currently we only provide supports on Linux kernel 2.6 and 2.4 although it can be ported the other Operation Systems. You should put together your Cross-Platform Development Tool-chain before you can build the SSDK.

On Linux-like OS, the SSDK can run in the kernel space and the user space. The main part of the SSDK is ssdk_ks_km.a when the SSDK runs in the kernel space. The user space static library ssdk_us_km.a which offers the interfaces to communicate between the kernel space and the user space can be used by user applications. The ssdk_us_km.a is not necessary if your application runs in the kernel space or you have your own user-kernel space communication methods.

The main part of the SSDK is ssdk_us_um.a when SSDK runs in the user space. We offer a static library ssdk_ks_um.a to access hardware register, which should be linked as a part of your kernel space driver in most cases. You don’t need it if you have your own interface, for example, a MDIO interface for some switch serials in the user space, to access hardware register.

3.2.1. Options

Usually you can use the following options when you make to satisfy particular requirements and you can change these options by editing the file config which locates in the root directory:
Cpu type:

CPU = mips (the CPU type of the target system, the default value is mips)

Operating system type:

OS = linux (we only support linux now, the default value is linux)

Operating system version:

OS_VER = 2_6 (We support linux kernel 2_4 and 2_6 version. The default version is 2_6)

Path:

SYS_PATH = the root path for the target operation system
PRJ_PATH = the root path for the project (The default value is current directory)
TOOLPATH = the path for cross-platform tool-chain

Switch chip type:
CHIP_TYPE = ATHENA, GARUDA , SHIVA or HORUS (The default value is GARUDA)

FAL included in the SSDK:
FAL = TRUE or FALSE (The default value is FALSE)

For example, if you don't need the FAL module you can set value of FAL to FALSE or use the default one, then the target of building will exclude the FAL module. However, the HSL should always be included.

The main part of the SSDK runs in the user space or the kernel space:
KERNEL_MODE=TRUE
If the main part of the SSDK runs in the kernel space then KERNEL_MODE should be defined as TRUE. Otherwise is FALSE.

USER-KERNEL space communication interfaces:
UK_IF=TRUE
Now for Linux we offered an interface to communicate between the user space to the kernel space based on netlink soceket in Linux. If you don't need these interface you can define UK_IF=FALSE. If you want to built shell then UK_IF must been defined as TRUE.

API_LOCK all APIs locker with locker or not
If you want all APIs with locker please define API_LOCK=TRUE. Otherwise please define API_LOCK=FALSE.

Features included in SSDK:
IN_ACL=TRUE
IN_FDB=TRUE
IN_IGMP=TRUE
IN_LEAKY=TRUE
IN_LED=TRUE
IN_MIB=TRUE
IN_MIRROR=TRUE
IN_MISC=TRUE
IN_PORTCONTROL=TRUE
IN_PORTVLAN=TRUE
IN_QOS=TRUE
IN_RATE=TRUE
IN_STP=TRUE
IN_VLAN=TRUE

IN_REDUCED_ACL=FALSE
IN_COSMAP= FALSE
IN_IP= FALSE
IN_NAT= FALSE
IN_TRUNK= FALSE
IN_SEC= FALSE
IN_NAT_HELPER=FALSE

For example, if you don't need the ACL feature you can set value of IN_ACL to FALSE, then the target of building will exclude the ACL related APIs.

3.2.2. Build target

- Building kernel space part library:
  make kslib

you will get ssdk_ks.a locates in PRJ_PATH/build/bin

- Building the user space part library:
  make uslib

you will get ssdk_us.a locates in PRJ_PATH/build/bin

- Building the Shell for userland:
  make shell

you will get ssdk_sh locates in PRJ_PATH/build/bin

And you can build all the targets by *make all.*

Please pay attention on editing the file config correctly before building target. For example you can define *TOOL_PATH=/home/sw/build/gcc-3.4.4-2.16.1/build_mips/bin* , *SYS_PATH=/home/sw/linux/kernels/mips-linux-2.6.15* and other options.

4. Porting

4.1. Initialization

The entry for initializing the SSDK is the function ssdk_init() in \src\init\ssdk_init.c no matter the FAL is included or not. The SSDK is designed for future multi-chip stacking system, so the identification dev_id, which is usually set to 0 for single switch chip system, is necessary for every switch chip. Other parameters for initialization are those SSDK initialization configurations that are described below:

- cpu_mode: It is used to represent connection mode between the switch and external devices via GMII/RGMII/RMII/MII. If there is only one connection which used to connect

the CPU port and the external CPU, the mode should be set to HSL_CPU_1. In the case there are two connections, one for the CPU port to one MAC of external CPU and another for the independent PHY to another MAC of the external CPU, the mode should be HSL_CPU_2. Furthermore, if one connection is for the CPU port to the external CPU and the other is for switch core to external device, the mode should be HSL_CPU_1_PLUS. Please make sure that the setting conforms to hardware design of your system.

- reg_mode: It is used to define the method for the SSDK to access switch register. There are two MDIO and header packets currently. Usually MDIO is recommended.
- reg_func: It is register access function pointers which point to the functions to access switch registers. It includes mdio_reg_set, mdio_reg_get, header_reg_set and header_reg_get.
- nl_prot: It is defined the netlink protocol type to communicate between the user space and the kernel space for Linux.

4.2. Register Access

The SSDK accesses switch registers through MDIO or header which should be provided by SDAPI in SAL. To be ported to various platforms, customers' Board System Package should provide APIs for MDIO and header that will be invoked by SDAPIs. The function pointers of these APIs are registered to the SSDK in initialization phase. They are ssdk_mdio_set, ssdk_mdio_get, ssdk_hdr_reg_set and ssdk_hdr_reg_get in SDAPIs. The user' driver should adapt their own MDIO functions to ssdk_mdio_set and ssdk_mdio_get. For the users who want to use header to configure the registers, they should write their own header configure functions for their specific systems. These four functions are registered to the SSDK by ssdk_init() via structure hsl_reg_func or by user's drivers via setting these variables directly.

5. Shell

5.1. Basics

The SSDK includes a CLI-like switch shell to configure the switch on Linux. If you have built the shell per chapter 3, you can invoke it by executing the file ssdk_sh.

The SSDK provides some useful help mechanisms to facilitate the usage of the Shell. To get help specific to a command mode, a command, a keyword, or an argument, use one of the following commands:

- Entering a question mark (?) at the Shell prompt allows you to obtain a list of commands available for each command mode.
- Entering "*abbreviated-command*?" the shell shows a list of commands that begin with a

particular character string. (No space between command and question mark.)

- Entering "*Command ?*" to get the keywords or arguments that you must enter next on the command line. (Space between command and question mark.)
- Entering "*abbreviated-command<Tab>*", the shell helps you to Completes a partial command name or lists all commands partially matched

Help messages of the SSDK use following conventions:

- Required command arguments are inside angle brackets (< >).
- Optional command arguments are in square brackets ([ ]).
- Alternative keywords are separated by vertical bars (|).
- The minimum and the maximum of a value range are separated by horizontal line (-).

To quit the Shell, you can enter "q" or "quit" at the Shell prompt.

5.2. Detailed commands

5.2.1. Port Control

<1> name: port duplex get
function: get duplex mode of a port
usage: port duplex get <port_id>

<2> name: port duplex set
function: set duplex mode of a port
usage: port duplex set <port_id> <half | full>

<3> name: port speed get
function: get speed mode of a port
usage: port speed get <port_id>

<4> name: port speed set
function: set speed mode of a port
usage: port speed set <port_id> <10 | 100 | 1000>

<5> name: port autoAdv get
function: get auto-negotiation advertisement of a port
usage: port autoAdv get <port_id>

<6> name: port autoAdv set
function: set auto-negotiation advertisement of a port
usage: port autoAdv set <port_id> <cap_bitmap>

<7> name: port autoNeg get

function: get auto-negotiation status of a port
usage: port autoNeg get <port_id>

<8> name: port autoNeg enable
function: enable auto-negotiation of a port
usage: port autoNeg enable <port_id>

< 9> name: port autoNeg restart
function: restart auto-negotiation process of a port
usage: port autoNeg restart <port_id>

<10> name: port header set
function: set atheros header/tag status of a port
usage: port header set <port_id> <enable | disable>

<11> name: port header get
function: get atheros header/tag status of a port
usage: port header get <port_id>

<12> name: port txhdr set
function: set atheros header/tag status of a port
usage: port txhdr set <port_id> <noheader|onlymanagement|allframe>

<13> name: port txhdr get
function: get atheros header/tag status of a port
usage: port txhdr get <port_id>

<14> name: port rxhdr set
function: set atheros header/tag status of a port
usage: port rxhdr set <port_id> <noheader|onlymanagement|allframe>

<15> name: port rxhdr get
function: get atheros header/tag status of a port
usage: port rxhdr get <port_id>

<16> name: port flowCtrl set
function: set flow control status of a port
usage: port flowCtrl set <port_id> <enable | disable>

<17> name: port flowCtrl get
function: get flow control status of a port
usage: port flowCtrl get <port_id>

<18> name: port powersave set

function: set power saving status of a port
usage: port powersave set <port_id> <enable | disable>

<19> name: port powersave get
function: get power saving status of a port
usage: port powersave get <port_id>

<20> name: port hibernate set
function: set hibernate status of a port
usage: port hibernate set <port_id> <enable | disable>

<21> name: port hibernate get
function: get hibernate status of a port
usage: port hibernate set <port_id>

<22> name: port cdt run
function: run cable diagnostic test of a port
usage: port cdt run <port_id> <mdi_pair>

5.2.2. VLAN

<1> name: vlan entry create
function: create a vlan entry
usage: vlan entry create <vlan_id>

<2> name: vlan entry del
function: delete a vlan entry
usage: vlan entry del <vlan_id>

<3> name: vlan entry update
function: update port member of a vlan entry
usage: vlan entry update <vlan_id> <member_bitmap> <0>

<4> name: vlan entry find
function: find a vlan entry by vlan id
usage: vlan entry find <vlan_id>

<5> name: vlan entry next
function: find next vlan entry by vlan id
usage: vlan entry next <vlan_id>

<6> name: vlan entry append
function: append a vlan entry

usage: vlan entry append

<7> name: vlan entry show
function: show whole vlan entries
usage: vlan entry show

<8> name: vlan entry flush
function: flush all VLAN entries
usage: vlan entry flush

<9> name: vlan fid set
function: set VLAN entry fid
usage: vlan fid set <vlan_id> <fid>

<10> name: vlan fid get
function: get VLAN entry fid
usage: vlan fid get <vlan_id>

<11> name: vlan member add
function: add VLAN entry member
usage: vlan member add <vlan_id> <port_id> <unmodified|untagged|tagged>

<12> name: vlan member del
function: del VLAN entry member
usage: vlan member del <vlan_id> <port_id>

<9> name: vlan learnsts set
function: set VLAN entry learn status
usage: vlan learnsts set <vlan_id> <enable|disable>

<10> name: vlan learnsts get
function: get VLAN entry learn status
usage: vlan learnsts get <vlan_id>

5.2.3. Port VLAN

<1> name: portVlan ingress get
function: get ingress vlan mode of a port
usage: portVlan ingress get <port_id>

<2> name: portVlan ingress set
function: set ingress vlan mode of a port
usage: portVlan ingress set <port_id> <disable | secure | check | fallback>

<3> name: portVlan egress get
function: get egress vlan mode of a port
usage: portVlan egress get <port_id>

<4> name: portVlan egress set
function: set egress vlan mode of a port
usage: portVlan egress set <port_id> <unmodified | untagged | tagged>

<5> name: portVlan member add
function: add a member to the port based vlan of a port
usage: portVlan member add <port_id> <memport_id>

<6> name: portVlan member del
function: delete a member from the port based vlan of a port
usage: portVlan member del <port_id> <memport_id>

<7> name: portVlan member update
function: update members of the port based vlan of a port
usage: portVlan member update <port_id> <port_bitmap>

<8> name: portVlan member get
function: get members of the port based vlan of a port
usage: portVlan member get <port_id>

<9> name: portVlan defaultVid get
function: get default vlan id of a port
usage: portVlan defaultVid get <port_id>

<10> name: portVlan defaultVid set
function: set default vlan id of a port
usage: portVlan defaultVid set <port_id> <vid>

<11> name: portVlan forceVid set
function: set vlan id enforcement status of a port
usage: portVlan forceVid set <port_id> <enable | disable>

<12> name: portVlan forceVid get
function: get vlan id enforcement status of a port
usage: portVlan forceVid get <port_id>

<13> name: portVlan forceMode set
function: set port based vlan enforcement status of a port
usage: portVlan forceMode set <port_id> <enable | disable>

<14> name: portVlan forceMode get
function: get port based vlan enforcement status of a port
usage: portVlan forceMode get <port_id>

<15> name: portVlan nestVlan set
function: set nest vlan status of a port
usage: portVlan nestVlan set <port_id> <enable | disable>

<16> name: portVlan nestVlan get
function: get nest vlan status of a port
usage: portVlan nestVlan get <port_id>

<17> name: portVlan sVlanTPID set
function: set service vlan tpid
usage: portVlan sVlanTPID set <tpid>

<18> name: portVlan sVlanTPID get
function: get service vlan tpid
usage: portVlan sVlanTPID get

<19> name: portVlan invlan set
function: set port invlan mode
usage: portVlan invlan set <port_id> <admit_all|admit_tagged|admit_untagged>

<20> name: portVlan invlan get
function: get port invlan mode
usage: portVlan invlan get <port_id>

<21> name: portVlan tlsMode set
function: set TLS mode
usage: portVlan tlsMode set <port_id> <enable|disable>

<22> name: portVlan tlsMode get
function: get TLS mode
usage: portVlan tlsMode get <port_id>

<23> name: portVlan priPropagation set
function: set priority propagation
usage: portVlan priPropagation set <port_id> <enable|disable>

<23> name: portVlan priPropagation get
function: get priority propagation
usage: portVlan priPropagation get <port_id>

<24> name: portVlan defaultSVid set
function: set default SVID
usage: portVlan defaultSVid set <port_id> <vlan_id>

<25> name: portVlan defaultSVid get
function: get default SVID
usage: portVlan defaultSVid get <port_id>

<26> name: portVlan defaultCVid set
function: set default CVID
usage: portVlan defaultCVid set <port_id> <vlan_id>

<27> name: portVlan defaultCVid get
function: get default CVID
usage: portVlan defaultCVid get <port_id>

<28> name: portVlan vlanPropagation set
function: set vlan propagation
usage: portVlan vlanPropagation set <port_id> <disable|clone|replace>

<29> name: portVlan vlanPropagation get
function: get vlan propagation
usage: portVlan vlanPropagation get <port_id>

<30> name: portVlan translation add
function: add vlan translation
usage: portVlan translation add <port_id>

<31> name: portVlan translation del
function: del vlan translation
usage: portVlan translation del <port_id>

<32> name: portVlan translation get
function: get vlan translation
usage: portVlan translation get <port_id>

<33> name: portVlan qinqMode set
function: set qinq mode
usage: portVlan qinqMode set <ctag|stag>

<34> name: portVlan qinqMode get
function: get qinq mode
usage: portVlan qinqMode get

<35> name: portVlan qinqRole set
function: set qinq role
usage: portVlan qinqRole set <port_id> <edge|core>

<36> name: portVlan qinqRole get
function: get qinq role
usage: portVlan qinqMode get <port_id>

<37> name: portVlan translation iterate
function: iterate vlan translation tables
usage: portVlan translation iterate <port_id> <iterator>

5.2.4. FDB

<1> name: fdb entry add
function: add a fdb entry
usage: fdb entry add

<2> name: fdb entry flush
function: flush all fdb entries
usage: fdb entry flush <0:dynamic only|1:dynamic and static>

<3> name: fdb entry show
function: show whole fdb entries
usage: fdb entry show

<4> name: fdb portEntry flush
function: flush all fdb entries by a port
usage: fdb portEntry flush <port_id> <0:dynamic only|1:dynamic and static>

<5> name: fdb Entry del
function: delete a fdb entry
usage: fdb Entry del

<6> name: fdb firstEntry find
function: find the first fdb entry
usage: fdb firstEntry find

<7> name: fdb nextEntry find
function: find next fdb entry
usage: fdb nextEntry find

<8> name: fdb entry find
function: find a fdb entry

usage: fdb entry find

<9> name: fdb portLearn set
function: set fdb entry learning status of a port
usage: fdb portLearn set <port_id> <enable | disable>

<10> name: fdb portLearn get
function: get fdb entry learning status of a port
usage: fdb portLearn get <port_id>

<11> name: fdb ageCtrl set
function: set fdb entry aging status
usage: fdb ageCtrl set <enable | disable>

<12> name: fdb ageCtrl get
function: get fdb entry aging status
usage: fdb ageCtrl get

<13> name: fdb ageTime set
function: set fdb entry aging time
usage: fdb ageTime set <time:s>

<14> name: fdb ageTime get
function: get fdb entry aging time
usage: fdb ageTime get

<15> name: fdb entry iterate
function: iterate all FDB entries
usage: fdb entry iterate <iterator>

<16> name: fdb ptlearnlimit set
function: set port FDB entry learn limit
usage: fdb ptlearnlimit set <port_id> <enable|disable> <limitcounter>

<17> name: fdb ptlearnlimit get
function: get port FDB entry learn limit
usage: fdb ptlearnlimit get <port_id>

<18> name: fdb ptlearnexceedcmd set
function: set port forwarding cmd when exceed learn limit
usage: fdb ptlearnexceedcmd set <port_id> <forward|drop|cpycpu|rdtcpu>

<19> name: fdb ptlearnexceedcmd get
function: get port forwarding cmd when exceed learn limit

usage: fdb ptlearnexceedcmd get <port_id>

<20> name: fdb learnlimit set
function: set FDB entry learn limit
usage: fdb learnlimit set <enable|disable> <limitcounter>

<21> name: fdb learnlimit get
function: get FDB entry learn limit
usage: fdb ptlearnlimit get

<22> name: fdb learnexceedcmd set
function: set forwarding cmd when exceed learn limit
usage: fdb ptlearnexceedcmd set <forward|drop|cpycpu|rdtcpu>

<23> name: fdb learnexceedcmd get
function: get FDB entry learn limit
usage: fdb learnexceedcmd get

<24> name: fdb learnexceedcmd get
function: get FDB entry learn limit
usage: fdb learnexceedcmd get

<25> name: fdb resventry add
function: add a reserve FDB entry
usage: fdb resventry add

<26> name: fdb resventry del
function: delete a reserve FDB entry
usage: fdb resventry del

<27> name: fdb resventry find
function: find a reserve FDB entry
usage: fdb resventry find

<28> name: fdb resventry iterate
function: iterate all reserve FDB entries
usage: fdb resventry iterate <iterator>

<29> name: fdb resventry show
function: show whole resv FDB entries
usage: fdb resventry show

<30> name: fdb ptLearnstatic set
function: set FDB entry learning static status of a port

usage: fdb ptLearnstatic set <port_id> <enable|disable>

<31> name: fdb ptLearnstatic get
function: get FDB entry learning static status of a port
usage: fdb ptLearnstatic get <port_id>

5.2.5. ACL

<1> name: acl list create
function: create an acl list
usage: acl list create <list_id> <priority>

<2> name: acl list destroy
function: destroy an acl list
usage: acl list destroy <list_id>

<3> name: acl list bind
function: bind an acl list to a port
usage: acl list bind <list_id> <0-0:direction> <0-0:objtype> <objindex>

<4> name: acl list unbind
function: unbind an acl list from a port
usage: acl list unbind <list_id> <0-0:direction> <0-0:objtype> <objindex>

<5> name: acl rule add
function: add acl rules to an acl list
usage: acl rule add <list_id> <rule_id> <rule_nr>

<6> name: acl rule delete
function: delete acl rules from an acl list
usage: acl rule delete <list_id> <rule_id> <rule_nr>

< 7> name: acl rule query
function: query a acl rule
usage: acl rule query <list_id> <rule_id>

<8> name: acl status set
function: set status of ACL engine
usage: acl status set <enable | disable>

<9> name: acl status get
function: get status of ACL engine
usage: acl status get

<10> name: acl udfprofile set
function: set port udf profile
usage: acl udfprofile set <port_id> <l2/l2snap/l3/l3plus/l4> <offset> <length>

<11> name: acl udfprofile get
function: get port udf profile
usage: acl udfprofile get <port_id> <l2/l2snap/l3/l3plus/l4>

5.2.6. QoS

<1> name: qos schMode set
function: set traffic scheduling mode
usage: qos schMode set <sp | wrr | mix> <q0,q1,q3,q4>

<2> name: qos schMode get
function: get traffic scheduling mode
usage: qos schMode get

<3> name: qos qTxBufSts set
function: set queue tx buffer counting status of a port
usage: qos qTxBufSts set <port_id> <enable | disable>

<4> name: qos qTxBufSts get
function: get qos queue tx buffer counting status of a port
usage: qos qTxBufSts get <port_id>

<5> name: qos qTxBufNr set
function: set queue tx buffer number
usage: qos qTxBuf set <port_id> <queueid:0-3> <number>

<6> name: qos qTxBufNr get
function: get queue tx buffer number
usage: qos qTxBuf get <port_id> <queueid:0-3>

<7> name: qos ptTxBufSts set
function: set port tx buffer counting status of a port
usage: qos ptTxBufSts set <port_id> <enable | disable>

<8> name: qos ptTxBufSts get
function: set port tx buffer counting status of a port
usage: qos ptTxBufSts get <port_id>

<9> name: qos ptTxBufNr set
function: set port tx buffer number
usage: qos ptTxBufNr set <port_id> <number>

<10> name: qos ptTxBufNr get
function: get port tx buffer number
usage: qos ptTxBufNr get <port_id>

<11> name: qos up2q set
function: set user priority to queue mapping
usage: qos up2q set <up:0-7> <queueid:0-3>

<12> name: qos up2q get
function: get user priority to queue mapping
usage: qos up2q get <up:0-7>

<13> name: qos dscp2q set
function: set dscp to queue mapping
usage: qos dscp2q set <dscp:0-63> <queueid:0-3>

<14> name: qos dscp2q get
function: get dscp to queue mapping
usage: qos dscp2q get <dscp:0-63>

<15> name: qos ptMode set
function: set qos mode of a port
usage: qos ptMode set <port_id> <da | up | dscp | port> <enable | disable>

<16> name: qos ptMode get
function: get qos mode of a port
usage: qos ptMode get <port_id> <da | up | dscp | port>

<17> name: qos ptModePri set
function: set the priority of qos modes of a port
usage: qos ptModePri set <port_id> <da | up | dscp | port> <priority:0-3>

<18> name: qos ptModePri get
function: get the priority of qos modes of a port
usage: qos ptModePri get <port_id> <da | up | dscp | port>

<19> name: qos ptDefaultUp set
function: set default user priority for received frames of a port
usage: qos ptDefaultUp set <port_id> <up:0-7>

<20> name: qos ptDefaultUp get
function: get default user priority for received frames of a port
usage: qos ptDefaultUp get <port_id>

<21> name: qos ptschMode set
function: set port traffic scheduling mode
usage: qos ptschMode set <port_id> <sp|wrr|mix|mixplus> <q0,q1,q2,q3>

<22> name: qos ptschMode get
function: get port traffic scheduling mode
usage: qos ptschMode get <port_id> <sp|wrr|mix|mixplus> <q0,q1,q2,q3>

<23> name: qos ptRxBufNr set
function: set port rx buffer number
usage: qos ptRxBufNr set <port_id> <number:0-60>

<24> name: qos ptRxBufNr get
function: get port rx buffer number
usage: qos ptRxBufNr get <port_id>

<23> name: qos ptDefaultSpri set
function: set default stag priority for received frames of a port
usage: qos ptDefaultSpri set <port_id> <spri:0-7>

<24> name: qos ptDefaultSpri get
function: get default stag priority for received frames of a port
usage: qos ptDefaultSpri get <port_id>

<23> name: qos ptDefaultCpri set
function: set default ctag priority for received frames of a port
usage: qos ptDefaultCpri set <port_id> <spri:0-7>

<24> name: qos ptDefaultCpri get
function: get default ctag priority for received frames of a port
usage: qos ptDefaultCpri get <port_id>

<23> name: qos ptQuRemark set
function: set egress queue based remark
usage: qos ptQuRemark set <port_id> <queue_id> <table_id> <enable|disable>

<24> name: qos ptQuRemark get
function: get egress queue based remark
usage: qos ptQuRemark get <port_id> <queue_id>

5.2.7. IGMP/MLD

<1> name: igmp mode set
function: set igmp/mld snooping status of a port
usage: igmp mode set <port_id> <enable | disable>

< 2> name: igmp mode get
function: get port igmp/mld snooping status
usage: igmp mode get <port_id>

<3> name: igmp cmd set
function: set igmp/mld frames forwarding command
usage: igmp cmd set <forward | drop | cpycpu | rdtcpu>

< 4> name: igmp cmd get
function: get igmp/mld frames forwarding command
usage: igmp cmd get

<5> name: igmp portJoin set
function: set igmp/mld hardware joining status
usage: igmp portJoin set <port_id> <enable | disable>

< 6> name: igmp portJoin get
function: get igmp/mld hardware joining status
usage: igmp portJoin get <port_id>

<7> name: igmp portLeave set
function: set igmp/mld hardware leaving status
usage: igmp portLeave set <port_id> <enable | disable>

< 8> name: igmp portLeave get
function: get igmp/mld hardware leaving status
usage: igmp portLeave get <port_id>

<9> name: igmp rp set
function: set igmp/mld router ports
usage: igmp rp set port_bit_map

<10> name: igmp rp get
function: get igmp/mld router ports
usage: igmp rp get

<11> name: igmp createStatus set
function: set igmp/mld ability for creating entry
usage: igmp createStatus set <enable | disable>

<12> name: igmp createStatus get
function: get igmp/mld ability for creating entry
usage: igmp createStatus get

<13> name: igmp static set
function: set IGMP/MLD static status for creating entry
usage: igmp static set <enable|disable>

<14> name: igmp static get
function: get IGMP/MLD static status for creating entry
usage: igmp static get

<15> name: igmp leaky set
function: set IGMP/MLD leaky status for creating entry
usage: igmp leaky set <enable|disable>

<16> name: igmp leaky get
function: get IGMP/MLD leaky status for creating entry
usage: igmp leaky get

<17> name: igmp version3 set
function: set IGMP v3/MLD v2 status for creating entry
usage: igmp version3 set <enable|disable>

<18> name: igmp version3 get
function: get IGMP v3/MLD v2 status for creating entry
usage: igmp version3 get

<19> name: igmp queue set
function: set IGMP/MLD queue status for creating entry
usage: igmp queue set <enable|disable>

<20> name: igmp queue get
function: get IGMP/MLD queue status for creating entry
usage: igmp queue get

<21> name: igmp ptlearnlimit set
function: set port Multicast entry learn limit
usage: igmp ptlearnlimit set <port_id> <enable|disable> <limitcounter>

<22> name: igmp ptlearnlimit get
function: get port Multicast entry learn limit
usage: igmp ptlearnlimit get <port_id>

<23> name: igmp ptlearnexceedcmd set
function: set port forwarding cmd when exceed multicast learn limit
usage: igmp ptlearnexceedcmd set <port_id> <forward|drop|cpycpu|rdtcpu>

<24> name: igmp ptlearnexceedcmd get
function: get port forwarding cmd when exceed multicast learn limit
usage: igmp ptlearnexceedcmd get <port_id>

5.2.8. Leaky

<1> name: leaky ucMode set
function: set unicast packets leaky mode
usage: leaky ucMode set <port | fdb>

<2> name: leaky ucMode get
function: get unicast packets leaky mode
usage: leaky ucMode get

<3> name: leaky mcMode set
function: set multicast packets leaky mode
usage: leaky mcMode set <port | fdb>

<4> name: leaky mcMode get
function: get multicast packets leaky mode
usage: leaky mcMode get

<5> name: leaky arp set
function: set arp packets leaky mode
usage: leaky arp set <port_id> < enable | disable>

<6> name: leaky arp get
function: get arp packets leaky mode
usage: leaky arp get <port_id>

<7> name: leaky ptUcMode set
function: set unicast packets leaky status of a port
usage: leaky ptUcMode set <port_id> <enable | disable>

<8> name: leaky ptUcMode get

function: get unicast packets leaky status of a port
usage: leaky ptUcMode get <port_id>

<9> name: leaky ptMcMode set
function: set multicast packets leaky status of a port
usage: leaky ptMcMode set <port_id> <enable | disable>

<10> name: leaky ptMcMode get
function: get multicast packets leaky status of a port
usage: leaky ptMcMode get <port_id>

5.2.9. Mirror

<1> name: mirror analyPt set
function: set mirror analysis port
usage: mirror analyPt set <port_id>

<2> name: mirror analyPt get
function: get mirror analysis port
usage: mirror analyPt get

<3> name: mirror ptIngress set
function: set ingress mirror status of a port
usage: mirror ptIngress set <port_id> <enable | disable>

<4> name: mirror ptIngress get
function: get ingress mirror status of a port
usage: mirror ptIngress get <port_id>

<5> name: mirror ptEgress set
function: set egress mirror status of a port
usage: mirror ptEgress set <port_id> <enable | disable>

<6> name: mirror ptEgress get
function: get egress mirror status of a port
usage: mirror ptEgress get <port_id>

5.2.10. Rate

<1> name: rate qEgress set
function: set egress rate limit of a queue
usage: rate qEgress set <port_id> <queueid:0-3> <speed:(kbps)> <enable | disable>

<2> name: rate qEgress get
function: get egress rate limit of a queue
usage: rate qEgress get <port_id> <queueid:0-3>

<3> name: rate ptEgress set
function: set egress rate limit of a port
usage: rate ptEgress set <port_id> <speed:(kbps)> <enable | disable>

<4> name: rate ptEgress get
function: get egress rate limit of a port
usage: rate ptEgress get <port_id>

<5> name: rate ptInress set
function: set ingress rate limit of a port
usage: rate ptInress set <port_id> <speed:(kbps)> <enable | disable>

< 6> name: rate ptInress get
function: get ingress rate limit of a port
usage: rate ptInress get <port_id>

<7> name: rate stormCtrl set
function: set storm control status of a particular frame type
usage: rate stormCtrl set <port_id> <unicast | multicast | broadcast> <enable | disable>

< 8> name: rate stormCtrl get
function: get storm control status of a particular frame type
usage: rate stormCtrl get <port_id> <unicast | multicast | broadcast>

<9> name: rate stormCtrlRate set
function: set storm ctrl rate
usage: rate stormCtrlRate set <port_id> <rate:(packets/s)>

<10> name: rate stormCtrlRate get
function: get storm ctrl rate
usage: rate stormCtrlRate get <port_id>

<11> name: rate portpolicer set
function: set port policer
usage: rate portpolicer set <port_id>

<12> name: rate portpolicer get
function: get storm ctrl rate
usage: rate portpolicer get <port_id>

<13> name: rate portshaper set
function: set port egress shaper
usage: rate portshaper set <port_id> <enable|disable>

<14> name: rate portshaper get
function: get port egress shaper
usage: rate portshaper get <port_id>

<15> name: rate queueshaper set
function: set queue egress shaper
usage: rate queueshaper set <port_id> <queue_id> <enable|disable>

<16> name: rate queueshaper get
function: get queue egress shaper
usage: rate queueshaper get <port_id> <queue_id>

<17> name: rate aclpolicer set
function: set acl policer
usage: rate aclpolicer set <policer_id>

<18> name: rate aclpolicer get
function: get acl policer
usage: rate aclpolicer get <policer_id>

5.2.11. STP

<1> name: stp ptState set
function: set stp state of a port
usage: stp ptState set st_id <port_id> <disable | block | listen | learn | forward>

<2> name: stp portState get
function: get stp state of a port
usage: stp ptState get st_id <port_id>

5.2.12. MIB

<1> name: mib statistics get
function: get statistics information of a port
usage: mib statistics get <port_id>

<2> name: mib status set

function: get mib status
usage: mib status set <enable | disable>

<3> name: mib status get
function: get mib status
usage: mib status set

5.2.13. LED

<1> name: led ctrlpattern set
function: set led control pattern
usage: led ctrlpattern set <group_id> <led_id>

<2> name: led ctrlpattern get
function: get led control pattern
usage: led ctrlpattern get <group_id> <led_id>

5.2.14. CoSMap

< 1> name: cosmap dscp2pri set
function: set dscp to priority map table
usage: cosmap dscp2pri set <dscp> <priority>

<2> name: cosmap dscp2pri get
function: get dscp to priority map table
usage: cosmap dscp2pri get <dscp>

< 3> name: cosmap dscp2dp set
function: set dscp to dp map table
usage: cosmap dscp2dp set <dscp> <dp>

<4> name: cosmap dscp2dp get
function: get dscp to dp map table
usage: cosmap dscp2dp get <dscp>

< 5> name: cosmap up2pri set
function: set dot1p to priority map table
usage: cosmap up2pri set <up> <priority>

<6> name: cosmap up2pri get
function: set dot1p to priority map table

usage: cosmap up2pri get <up>

< 7> name: cosmap up2dp set
function: set dot1p to dp map table
usage: cosmap up2dp set <up> <dp>

<8> name: cosmap up2dp get
function: set dot1p to dp map table
usage: cosmap up2dp get <up>

< 9> name: cosmap pri2q set
function: set priority to queue mapping
usage: cosmap pri2q set <priority> <queueid>

<10>name: cosmap pri2q get
function: get priority to queue mapping
usage: cosmap pri2q get <priority>

< 11> name: cosmap pri2ehq set
function: set priority to enhanced queue mapping
usage: cosmap pri2ehq set <priority> <queueid>

<12> name: cosmap pri2ehq get
function: get priority to enhanced queue mapping
usage: cosmap pri2ehq get <priority>

5.2.15. Misc

< 1> name: misc arp set
function: set arp packets hardware identification status
usage: misc arp set <enable | disable>

<2> name: misc arp get
function: get arp packets hardware identification status
usage: misc arp get

<3> name: misc frameMaxSize set
function: set the maximal received frame size of the device
usage: misc frameMaxSize set <size:byte>

<4> name: misc frameMaxSize get
function: get the maximal received frame size of the device
usage: misc frameMaxSize get

<5> name: misc ptUnkSaCmd set
function: set forwarding command for frames with unknown source address
usage: misc ptUnkSaCmd set <port_id> <forward | drop | cpycpu | rdtcpu>

<6> name: misc ptUnkSaCmd get
function: get forwarding command for frames with unknown source address
usage: misc ptUnkSaCmd get <port_id>

< 7> name: misc ptUnkUcFilter set
function: set flooding status of unknown unicast frames
usage: misc ptUnkUcFilter set <port_id> <enable | disable>

<8> name: misc ptUnkUcFilter get
function: get flooding status of unknown unicast frames
usage: misc ptUnkUcFilter get <port_id>

<9> name: misc ptUnkMcFilter set
function: set flooding status of unknown multicast frames
usage: misc ptUnkMcFilter set <port_id> <enable | disable>

<10> name: misc ptUnkMcFilter get
function: get flooding status of unknown multicast frames
usage: misc ptUnkMcFilter get <port_id>

<11> name: misc cpuPort set
function: set cpuport status
usage: misc cpuPort set <enable | disable>

<12> name: misc cpuPort get
function: get cpu port status
usage: misc cpuPort get

<13> name: misc bctoCpu set
function: set broadcast frames to Cpu port status
usage: misc bctoCpu set <enable | disable>

<14> name: misc bctoCpu get
function: get broadcast frames to Cpu port status
usage: misc bctoCpu get

<15> name: misc PppoeCmd set
function: set pppoe frames forwarding command
usage: misc PppoeCmd set <forward | drop | cpycpu | rdtcpu>

<16> name: misc PppoeCmd get
function: get pppoe frames forwarding command
usage: misc PppoeCmd get

<17> name: misc Pppoe set
function: set pppoe frames hardware identification status
usage: misc Pppoe set <enable | disable>

<18> name: misc Pppoe get
function: get pppoe frames hardware identification status
usage: misc Pppoe get

<19> name: misc ptDhcp set
function: set dhcp frames hardware identification status
usage: misc ptDhcp set <port_id> <enable | disable>

<20> name: misc ptDhcp get
function: get dhcp frames hardware identification status
usage: misc ptDhcp get <port_id>

<21> name: misc arpcmd set
function: set arp packets forwarding command
usage: misc arpcmd set <forward|drop|cpycpu|rdtcpu>

<22> name: misc arpcmd get
function: get arp packets forwarding command
usage: misc arpcmd get

<23> name: misc eapolcmd set
function: set eapol packets forwarding command
usage: misc eapolcmd set <forward|drop|cpycpu|rdtcpu>

<24> name: misc eapolcmd get
function: set eapol packets forwarding command
usage: misc eapolcmd get

<25> name: misc pppoesession add
function: add a pppoe session entry
usage: misc pppoesession add <session_id> <enable|disable>

<26> name: misc pppoesession del
function: del a pppoe session entry
usage: misc pppoesession del <session_id>

<27> name: misc pppoesession get
function: get a pppoe session entry
usage: misc pppoesession get <session_id>

<28> name: misc eapolstatus set
function: set eapol frames hardware identification status
usage: misc eapolstatus set <port_id> <enable|disable>

<29> name: misc eapolstatus get
function: get eapol frames hardware identification status
usage: misc eapolstatus get <port_id>

<30> name: misc rip set
function: set rip packets hardware identification status
usage: misc rip set <enable|disable>

<31> name: misc rip get
function: get rip packets hardware identification status
usage: misc rip get

<32> name: misc ptarpreq set
function: set arp request packets hardware identification status
usage: misc ptarpreq set <port_id> <enable|disable>

<33> name: misc ptarpreq get
function: get arp request packets hardware identification status
usage: misc ptarpreq get <port_id>

<34> name: misc ptarpack set
function: set arp ack packets hardware identification status
usage: misc ptarpack set <port_id> <enable|disable>

<35> name: misc ptarpack get
function: get arp ack packets hardware identification status
usage: misc ptarpack get <port_id>

<36> name: misc extendpppoe add
function: add a pppoe session entry
usage: misc extendpppoe add

<37> name: misc extendpppoe del
function: del a pppoe session entry
usage: misc extendpppoe del

<38> name: misc extendpppoe get
function: get a pppoe session entry
usage: misc extendpppoe get

5.2.16. IP

<1> name: ip hostentry add
function: add host entry
usage: ip hostentry add

<2> name: ip hostentry del
function: del host entry
usage: ip hostentry del <del_mode>

<3> name: ip hostentry get
function: get host entry
usage: ip hostentry get <get_mode>

<4> name: ip hostentry next
function: next host entry
usage: ip hostentry next <next_mode>

<5> name: ip hostentry show
function: show whole host entries
usage: ip hostentry show

<6> name: ip hostentry bindcnt
function: bind counter to host entry
usage: ip hostentry bindcnt <host entry id> <cnt id> <enable|disable>

<7> name: ip hostentry bindpppoe
function: bind pppoe to host entry
usage: ip hostentry bindpppoe <host entry id> <pppoe id> <enable|disable>

<8> name: ip ptarplearn set
function: set port arp learn flag, bit0 req bit1 ack
usage: ip ptarplearn set <port_id> <flag>

<9> name: ip ptarplearn get
function: get port arp learn flag, bit0 req bit1 ack
usage: ip ptarplearn get <port_id>

<10> name: ip arplearn set
function: set arp learn mode
usage: ip arplearn set <learnlocal|learnall>

<11> name: ip arplearn get
function: get arp learn mode
usage: ip arplearn get <port_id>

<12> name: ip ptipsrcguard set
function: set ip source guard mode
usage: ip ptipsrcguard set <port_id> <mac_ip|mac_ip_port|mac_ip_vlan|
mac_ip_port_vlan|no_guard>

<13> name: ip ptipsrcguard get
function: get ip source guard mode
usage: ip ptipsrcguard get <port_id>

<14> name: ip ptarpsrcguard set
function: set arp source guard mode
usage: ip ptarpsrcguard set <port_id> <mac_ip|mac_ip_port|mac_ip_vlan|
mac_ip_port_vlan|no_guard>

<15> name: ip ptarpsrcguard get
function: get arp source guard mode
usage: ip ptarpsrcguard get <port_id>

<16> name: ip routestatus set
function: set ip route status
usage: ip routestatus set <enable|disable>"

<17> name: ip routestatus get
function: get ip route status
usage: ip routestatus get

<18> name: ip intfentry add
function: add interface mac address
usage: ip intfentry add <enable|disable>"

<19> name: ip intfentry del
function: del interface mac address
usage: ip intfentry del

<20> name: ip intfentry add

function: add interface mac address
usage: ip intfentry add <enable|disable>

<21> name: ip intfentry show
function: show whole interface mac entries
usage: ip intfentry show

<22> name: ip ipunksrc set
function: set ip unkown source command
usage: ip ipunksrc set <forward|drop|cpycpu|rdtcpu>

<23> name: ip ipunksrc get
function: get ip unkown source command
usage: ip ipunksrc get

<24> name: ip arpunksrc set
function: set arp unkown source command
usage: ip arpunksrc set <forward|drop|cpycpu|rdtcpu>

<25> name: ip arpunksrc get
function: get arp unkown source command
usage: ip arpunksrc get

<26> name: ip ip6baseaddr set
function: set ip6 base address
usage: ip ip6baseaddr set <forward|drop|cpycpu|rdtcpu>

<27> name: ip ip6baseaddr get
function: get ip6 base address
usage: ip ip6baseaddr get

5.2.17. NAT

<1> name: nat natentry add
function: add nat entry
usage: nat natentry add

<2> name: nat natentry del
function: del nat entry
usage: nat natentry del <del_mode>

<3> name: nat natentry get
function: get nat entry

usage: nat natentry get <get_mode>

<4> name: nat natentry show
function: show whole nat entries
usage: nat natentry show

<5> name: nat natentry bindcnt
function: bind counter to nat entry
usage: nat natentry bindcnt <nat entry id> <cnt id> <enable|disable>

<6> name: nat naptentry add
function: add napt entry
usage: nat naptentry add

<7> name: nat naptentry del
function: del napt entry
usage: nat naptentry del <del_mode>

<8> name: nat naptentry get
function: get napt entry
usage: nat naptentry get <get_mode>

<9> name: nat naptentry show
function: show whole napt entries
usage: nat naptentry show

<10> name: nat naptentry bindcnt
function: bind counter to napt entry
usage: nat naptentry bindcnt <napt entry id> <cnt id> <enable|disable>

<11> name: nat natstatus set
function: set nat status
usage: nat natstatus set <enable|disable>

<12> name: nat natstatus get
function: get nat status
usage: nat natstatus get

<13> name: nat naptstatus set
function: set napt status
usage: nat naptstatus set <enable|disable>

<14> name: nat naptstatus get
function: get napt status

usage: nat naptstatus get

<15> name: nat nathash set
function: set nat hash mode
usage: nat nathash set <flag>

<16> name: nat nathash get
function: get nat hash mode
usage: nat nathash get

<17> name: nat naptmode set
function: set napt mode
usage: nat naptmode set <fullcone|strictcone|portstrict|synmatric>

<18> name: nat naptmode get
function: get napt mode
usage: nat naptmode get

<19> name: nat prvbaseaddr set
function: set nat prv base address
usage: nat prvbaseaddr set <ip4 addr>

<20> name: nat prvbaseaddr get
function: get nat prv base address
usage: nat prvbaseaddr get

<21> name: nat prvaddrmode set
function: set nat prv address map mode
usage: nat prvaddrmode set <enable|disable>

<22> name: nat prvaddrmode get
function: get nat prv address map mode
usage: nat prvaddrmode get

<23> name: nat pubaddr add
function: add pub address
usage: nat pubaddr add <enable|disable>

<24> name: nat pubaddr del
function: del pub address
usage: nat pubaddr del <del_mode>

<25> name: nat pubaddr show
function: show whole pub address entries

usage: nat pubaddr show

<26> name: nat natunksess set
function: set nat unkown session command
usage: nat natunksess set <forward|drop|cpycpu|rdtcpu>

<27> name: nat natunksess get
function: get nat unkown session command
usage: nat natunksess get

5.2.18. Trunk

<1> name: trunk group set
function: set trunk group member info
usage: trunk group set <trunk_id> <disable|enable> <port_bitmap>

<2> name: trunk group get
function: get trunk group member info
usage: trunk group get <trunk_id>

<3> name: trunk hashmode set
function: set trunk hash mode
usage: trunk hashmode set <hash_mode>

<4> name: trunk hashmode get
function: get trunk hash mode
usage: trunk hashmode get <trunk_id>

5.2.19. Register Access and Debug

<1> name: debug phy get
function: read phy register
usage: debug phy get <ph_id> <reg_addr>

<2> name: debug phy set
function: write phy register
usage: debug phy set <ph_id> <reg_addr> <value>

<3> name: debug reg get
function: read switch register
usage: debug reg get <reg_addr>

< 4> name: debug reg set
function: write switch register
usage: debug reg set <reg_addr> <value>

<5> name: debug entry get
function: read switch register entry
usage: debug entry get <entry_name>

<6> name: debug entry set
function: write switch register entry
usage: debug entry set <entry_name>

< 7> name: debug field get
function: read switch register field
usage: debug field get <field_name>

<8>. name: debug field set
function: write switch register field
usage: debug field set <field_name>

<9>. name: debug aclList dump
function: dump all acl lists
usage: debug aclList dump

<10> name: debug aclRule dump
function: dump all acl rules
usage: debug aclRule dump

5.2.20. Set Device ID

<1> name: device id set
function: set device id
usage: device id set <dev_id>