Manual:Switch Chip Features

From MikroTik Wiki

Applies to RouterOS: v6.0 +

Contents

Introduction

There are several types of switch chips on Routerboards and they have a different set of features. Most of them (from now on "Other") have only basic "Port Switching" feature, but there are few with more features:

Capabilities of switch chips:

Feature	QCA8337	Atheros8327	Atheros8316	Atheros8227	Atheros7240	ICPlus175D	MT7621	RTL8367	Other
Port Switching	yes	yes	yes	yes	yes	yes	yes	yes	yes
Port Mirroring	yes	yes	yes	yes	yes	yes	yes	yes	no
Host table	2048 entries	2048 entries	2048 entries	1024 entries	2048 entries	no	2048 entries	2048 entries	no
Vlan table	4096 entries	4096 entries	4096 entries	4096 entries	16 entries	no	no	no	no
Rule table	92 rules	92 rules	32 rules	no	no	no	no	no	no

RouterBoard	Switch-chip description
RB1100AHx4	RTL8367 (ether1-ether5); RTL8367 (ether6-ether10); RTL8367 (ether11-ether13)
RB750Gr3 (hEX)	MT7621 (ether1-ether5)
RB3011 series	QCA8337 (ether1-ether5); QCA8337 (ether6-ether10)
RB OmniTik ac series	QCA8337 (ether1-ether5)
RB941-2nD (hAP lite)	Atheros8227 (ether1-ether4)
RB951Ui-2nD (hAP); RB952Ui-5ac2nD (hAP ac lite)	Atheros8227 (ether1-ether5)
RB750r2 (hEX lite); RB750UPr2 (hEX PoE lite); RB750P-PBr2 (PowerBox)	Atheros8227 (ether1-ether5)
RB750Gr2 (hEX); RB962UiGS-5HacT2HnT (hAP ac); RB960PGS (hEX PoE); RB960PGS-PB (PowerBox Pro)	QCA8337 (ether1-ether5)
RB750P r2	Atheros8227 (ether1-ether5)
RB953GS	Atheros8327 (ether1-ether3+sfp1)
RB850Gx2	Atheros8327 (ether1-ether5) with ether1 optional [more (http://wiki.mikrotik.com/wiki/Manual:Switch_Chip_Features#switch-all-ports)]
RB2011 series	Atheros8327 (ether1-ether5+sfp1); Atheros8227 (ether6-ether10)
RB750GL	Atheros8327 (ether1-ether5)

RB751G-2HnD	Atheros8327 (ether1-ether5)
RB951G-2HnD	Atheros8327 (ether1-ether5)
RB1100AH	Atheros8327 (ether1-ether5); Atheros8327 (ether6-ether10)
RB1100AHx2	Atheros8327 (ether1-ether5); Atheros8327 (ether6-ether10)
CCR1009 series	Atheros8327 (ether1-ether4)
RB493G	Atheros8316 (ether1+ether6-ether9); Atheros8316 (ether2-ether5)
RB435G	Atheros8316 (ether1-ether3) with ether1 optional [more (http://wiki.mikrotik.com/wiki/Manual:Switch_Chip_Features#switch-all-ports)]
RB450G	Atheros8316 (ether1-ether5) with ether1 optional [more (http://wiki.mikrotik.com/wiki/Manual:Switch_Chip_Features#switch-all-ports)]
RB433GL	Atheros8327 (ether1-ether3)
RB750G	Atheros8316 (ether1-ether5)
RB1200	Atheros8316 (ether1-ether5)
RB1100	Atheros8316 (ether1-ether5); Atheros8316 (ether6-ether10)
RB750	Atheros7240 (ether2-ether5)
RB750UP	Atheros7240 (ether2-ether5)
RB751U-2HnD	Atheros7240 (ether2-ether5)
RB951-2n	Atheros7240 (ether2-ether5)
RB951Ui-2HnD	Atheros8227 (ether1-ether5)
RB433 series	ICPlus175D (ether2-ether3); older models had ICPlus175C
RB450	ICPlus175D (ether2-ether5); older models had ICPlus175C
RB493 series	ICPlus178C (ether2-ether9)
RB816	ICPlus178C (ether1-ether16)

Command line config is under `/interface ethernet switch` menu. This menu contains a list of all switch chips present in system, and some sub-menus as well. `/interface ethernet switch` menu list item represents a switch chip in system:

```
[admin@MikroTik] /interface ethernet switch> print
Flags: I - invalid
 #   NAME     TYPE         MIRROR-SOURCE   MIRROR-TARGET
 0   switch1  Atheros-8316 ether2          none
```

Depending on switch type there might be available or not available some configuration capabilities.

Atheros8316 packet flow diagram (http://wiki.mikrotik.com/wiki/Manual:Packet_flow_through_Atheros8316)

Features

Port Switching

Switching feature allows wire speed traffic passing among a group of ports, like the ports were a regular ethernet switch. You configure this feature by setting a "master-port" property to one ore more ports in `/interface ethernet` menu. A 'master' port will be the port through which the RouterOS will communicate to all ports in the group. Interfaces for which the 'master' port is specified become inactive - no traffic is received on them and no traffic can be sent out.

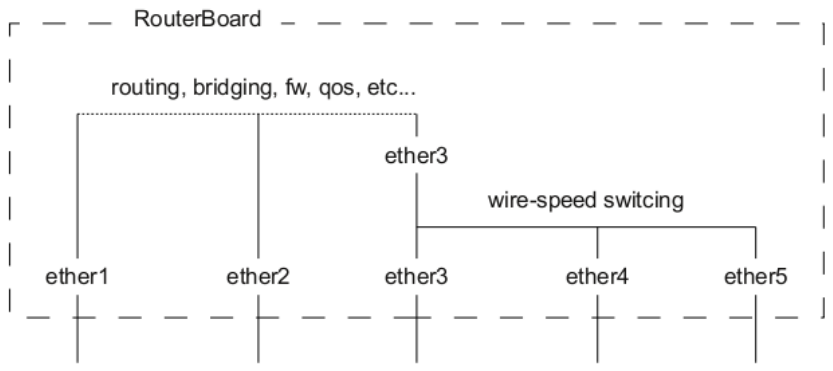
For example consider a router with five ethernet interfaces:

```
[admin@MikroTik] > interface ethernet print
Flags: X - disabled, R - running, S - slave
 #    NAME      MTU   MAC-ADDRESS       ARP        MASTER-PORT      SWITCH
 0 R  ether1    1500  00:0C:42:3E:5D:BB enabled
 1    ether2    1500  00:0C:42:3E:5D:BC enabled    none             switch1
 2    ether3    1500  00:0C:42:3E:5D:BD enabled    none             switch1
 3    ether4    1500  00:0C:42:3E:5D:BE enabled    none             switch1
 4 R  ether5    1500  00:0C:42:3E:5D:BF enabled    none             switch1
```

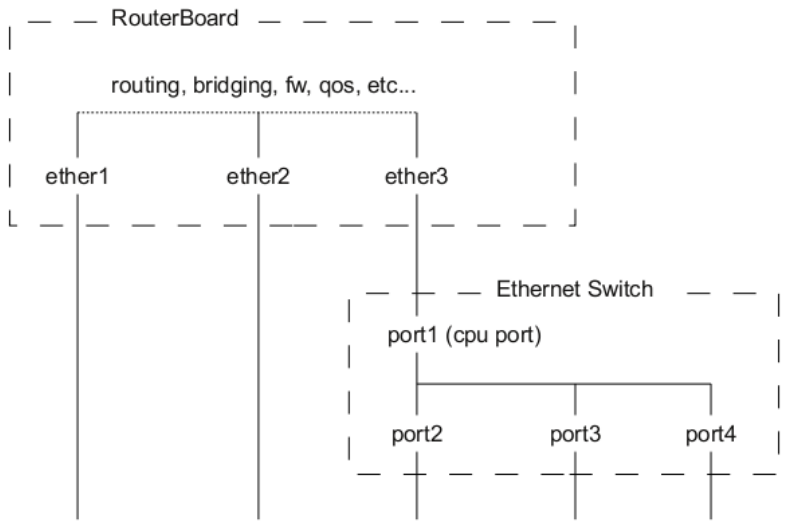
And you configure a switch containing three ports ether3, ether4 and ether5:

```
[admin@MikroTik] /interface ethernet> set ether4,ether5 master-port=ether3
[admin@MikroTik] /interface ethernet> print
Flags: X - disabled, R - running, S - slave
 #    NAME      MTU   MAC-ADDRESS       ARP        MASTER-PORT      SWITCH
 0 R  ether1    1500  00:0C:42:3E:5D:BB enabled
 1    ether2    1500  00:0C:42:3E:5D:BC enabled    none             switch1
 2 R  ether3    1500  00:0C:42:3E:5D:BD enabled    none             switch1
 3  S ether4    1500  00:0C:42:3E:5D:BE enabled    ether3           switch1
 4 RS ether5    1500  00:0C:42:3E:5D:BF enabled    ether3           switch1
```

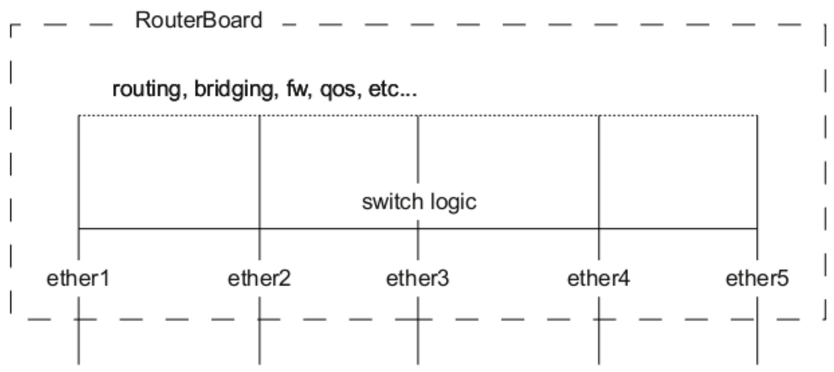
ether3 is now the master port of the group. Note: you can see that previously a link was detected only on ether5, but now as the ether3 is a 'master' the running flag is propagated to master port.



In essence this configuration is the same as if you had a RouterBoard with 3 ethernet interfaces with ether3 connected to ethernet switch that has 4 ports:



A more general diagram of RouterBoard with switch chip that has 5 port switch chip:



Here you can see that, a packet that gets received by one of the ports always passes through the switch logic at first. Switch logic decides to which ports the packet should be going to. Passing packet 'up' or giving it to RouterOS is also called sending it to switch chips 'cpu' port. That means that at the point switch forwards the packet to cpu port the packet starts to get processed by RouterOS as some interfaces incoming packet. While the packet does not have to go to cpu port it is handled entirely by switch logic and does not require any cpu cycles and happen at wire speed for any frame size.

Bridge Hardware Offloading

Since RouterOS v6.41 there are user interface changes which convert RouterBoard master-port configuration into a bridge with hardware offloading. From now on bridges will handle all Layer2 forwarding and the use of switch chip (`hw-offload`) will automatically turn on if appropriate conditions are met. The rest of RouterOS Switch features remain untouched in usual menus. By default all newly created bridge ports have `hw=yes` option and it allows enabling of `hw-offload` when possible. If such functionality is not required, it can be disabled by `hw=no` on bridge port to have completely software operated bridging.

Note: Downgrading to previous RouterOS versions will not restore master-port configuration. The bridge with no hw-offload will appear instead and master-port configuration will have to be redone from the beginning.

Following table states what features currently in v6.40rc keep bridge hardware offloading enabled on certain RouterBoard and switch chip models.

Notes:

- Enabling this feature maintains hw-offload: +
- Enabling this feature turns off hw-offload: -

RouterBoard/[Switch Chip] Model	Features in Switch menu	Bridge STP/RSTP	Bridge MSTP	Bridge IGMP Snooping	Bridge VLAN Filtering	Bonding
CRS3xx series	+	+	+	+	+	-
CRS1xx/CRS2xx series	+	+	-	+	-	-
[QCA8337]	+	+	-	-	-	-
[AR8327]	+	+	-	-	-	-
[AR8227]	+	+	-	-	-	-
[AR8316]	+	+	-	-	-	-
[AR7240]	+	+	-	-	-	-
RB750Gr3 [MT7621]	+	-	-	-	-	-
RB1100AHx4 [RTL8367]	+	-	-	-	-	-
[ICPlus175D]	+	-	-	-	-	-

- Port switching with master-port configuration before v6.40rc29

```
[admin@MikroTik] > interface ethernet export
/interface ethernet
set [ find default-name=ether3 ] master-port=ether2
set [ find default-name=ether4 ] master-port=ether2
set [ find default-name=ether5 ] master-port=ether2
[admin@MikroTik] >

[admin@MikroTik] > interface ethernet print
Flags: X - disabled, R - running, S - slave
 #    NAME          MTU MAC-ADDRESS       ARP             MASTER-PORT          SWITCH
 0 R  ether1       1500 D4:CA:6D:E2:64:64 enabled         none                 switch1
 1 R  ether2       1500 D4:CA:6D:E2:64:65 enabled         none                 switch1
 2 RS ether3       1500 D4:CA:6D:E2:64:66 enabled         ether2               switch1
 3 RS ether4       1500 D4:CA:6D:E2:64:67 enabled         ether2               switch1
 4 RS ether5       1500 D4:CA:6D:E2:64:68 enabled         ether2               switch1
[admin@MikroTik] >
```

- Port switching with bridge configuration and enabled hw-offload since v6.40rc29

```
[admin@MikroTik] > interface bridge export
/interface bridge
add name=bridge1 igmp-snooping=no  protocol-mode=none
/interface bridge port
add bridge=bridge1 interface=ether2
add bridge=bridge1 interface=ether3
add bridge=bridge1 interface=ether4
add bridge=bridge1 interface=ether5
[admin@MikroTik] >

[admin@MikroTik] > interface bridge port print
Flags: X - disabled, I - inactive, D - dynamic, H - hw-offload
 #     INTERFACE              BRIDGE              HW  PVID PRIORITY  PATH-COST INTERNAL-PATH-COST    HORIZON
 0   H ether2                 bridge1             yes    1     0x80         10                 10       none
 1   H ether3                 bridge1             yes    1     0x80         10                 10       none
 2   H ether4                 bridge1             yes    1     0x80         10                 10       none
 3   H ether5                 bridge1             yes    1     0x80         10                 10       none
[admin@MikroTik] >
```

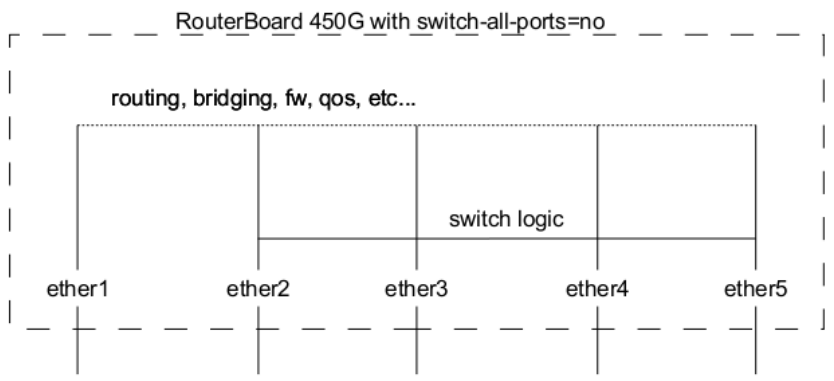
Switch All Ports Feature

Ether1 port on RB450G/RB435G/RB850Gx2 has a feature that allows it to be removed/added to the default switch group. By default ether1 port will be included in the switch group. This configuration can be changed with `/interface ethernet switch set switch1 switch-all-ports=no`

- **switch-all-ports**=yes/no -

"yes" means ether1 is part of switch and supports switch grouping, and all other advanced Atheros8316/Atheros8327 features including extended statistics (`/interface ethernet print stats`).

"no" means ether1 is not part of switch, effectively making it as stand alone ethernet port, this way increasing its throughput to other ports in bridged, and routed mode, but removing the switching possibility on this port.


RouterBoard 450G with switch-all-ports=no
routing, bridging, fw, qos, etc...
switch logic
ether1
ether2
ether3
ether4
ether5

Port Mirroring

Port mirroring lets switch 'sniff' all traffic that is going in and out of one port (mirror-source) and send a copy of those packets out of some other port (mirror-target). This feature can be used to easily set up a 'tap' device that receives all traffic that goes in/out of some specific port. Note that mirror-source and mirror-target ports have to belong to same switch. (See which port belong to which switch in `/interface ethernet` menu). Also mirror-target can have a special 'cpu' value, which means that 'sniffed' packets should be sent out of switch chips cpu port. Port mirroring happens independently of switching groups that have or have not been set up.

- Port mirroring configuration example:

```
/interface ethernet switch
set switch1 mirror-source=ether2 mirror-target=ether3
```

Warning: If you set mirror-source as a Ethernet port for a device with at least two switch chips and these mirror-source ports are in a single bridge while mirror-target for both switch chips are set to send the packets to the CPU, then this will result in a loop, which can make your device inaccessible.

Host Table

Basically the table represents switch chips internal mac address to port mapping. It can contain two kinds of entries: dynamic and static. Dynamic entries get added automatically, this is also called a learning process: when switch chip receives a packet from certain port, it adds the packets source mac address X and port it received the packet from to host table, so when a packet comes in with destination mac address X it knows to which port it should forward the packet. If the destination mac address is not present in host table then it forwards the packet to all ports in the group. Dynamic entries take about 5 minutes to time out. Learning is enabled only on ports that are configured as part of switch group. So you won't see dynamic entries if you have not specified some 'master-ports'. Also you can add static entries that take over dynamic if dynamic entry with same mac-address already exists. Also by adding a static entry you get access to some more functionality that is controlled via following params:

- **copy-to-cpu**=yes/no - a packet can be cloned and sent to cpu port
- **redirect-to-cpu**=yes/no - a packet can be redirected to cpu port
- **mirror**=yes/no - a packet can be cloned and sent to mirror-target port configured in "/interface ethernet switch"
- **drop**=yes/no - a packet with certain mac address coming from certain ports can be dropped

copy-to-cpu, redirect-to-cpu, mirror actions are performed for packets which destination mac matches mac address specified in entry drop action is performed for packets which source mac address matches mac address specified in entry

Another possibility for static entries is that mac address can be mapped to more that one port, including 'cpu' port.

Vlan Table

Vlan table specifies certain forwarding rules for packets that have specific 802.1q tag. Those rules are of higher priority than switch groups configured using 'master-port' property. Basically the table contains entries that map specific vlan tag ids to a group of one or more ports. Packets with vlan tags leave switch chip through one or more ports that are set in corresponding table entry. The exact logic that controls how packets with vlan tags are treated is controlled by vlan-mode parameter that is changeable per switch port in `/interface ethernet switch port` menu. Vlan-mode can take following values:

- **disabled** - ignore vlan table, treat packet with vlan tags just as if they did not contain a vlan tag;
- **fallback** - the default mode - handle packets with vlan tag that is not present in vlan table just like packets without vlan tag. Packets with vlan tags that are present in vlan table, but incoming port does not match any port in vlan table entry does not get dropped.
- **check** - drop packets with vlan tag that is not present in vlan table. Packets with vlan tags that are present in vlan table, but incoming port does not match any port in vlan table entry does not get dropped.
- **secure** - drop packets with vlan tag that is not present in vlan table. Packets with vlan tags that are present in vlan table, but incoming port does not match any port in vlan table entry get dropped.

Vlan tag id based forwarding takes into account the MAC addresses dynamically learned or manually added in the host table. QCA8337 and AR8327 switch-chips also support Independent VLAN learning (IVL) which does the learning based on both MAC addresses and VLAN IDs thus allowing the same MAC to be used in multiple VLANs. The option "independent-learning" in VLAN table entries enables this feature.

Packets without vlan tag are treated just like if they had a vlan tag with port `default-vlan-id`. This means that if "vlan-mode=check or secure" to be able to forward packets without vlan tags you have to add a special entry to vlan table with the same vlan id set according to `default-vlan-id`.

Vlan-header option (configured in `/interface ethernet switch port`) sets the VLAN tag mode on egress port. Starting from RouterOS version 6 this option works with QCA8337, AR8316, AR8327, AR8227 and AR7240 switch chips and takes the following values:

- **leave-as-is** - packet remains unchanged on egress port;
- **always-strip** - if VLAN header is present it is removed from the packet;
- **add-if-missing** - if VLAN header is not present it is added to the packet.

Rule Table

Rule table is very powerful tool allowing wire speed packet filtering, forwarding and vlan tagging based on L2,L3,L4 protocol header field condition.

Each rule contains a conditions part and an action part. Action part is controlled by following parameters:

- **copy-to-cpu**=yes/no - clones matching packets and sends them to cpu port;
- **redirect-to-cpu**=yes/no - redirects matching packets to cpu port;
- **mirror**=yes/no - clones matching packets and send them to mirror-target port;
- **new-dst-ports** - if set forces the destination port to be as specified, multiple ports allowed, including cpu port. Non obvious feature of this parameter is to pass empty list of ports to drop matching packets;
- **new-vlan-id** ***(only applies to Atheros8316)*** - if specified changes the vlan tag id, or add new vlan tag if one was not present;
- **new-vlan-priority** - if specified changes the vlan tag priority bits;
- **rate** ***(only applies to Atheros8327/QCA8337)*** - Sets limitation (bits per second) for all matched traffic. Can only be applied to first 32 rule slots.

Conditions part is controlled by rest of parameters:

- **ports** - match port that packet came in from (multiple ports allowed);
- **mac layer conditions**
  - *dst-mac-address* - match by destination mac address and mask;
  - *src-mac-address* - ...;
  - *vlan-header* - match by vlan header presence;
  - *vlan-id* ***(only applies to Atheros8316)*** - match by vlan tag id;
  - *vlan-priority* ***(only applies to Atheros8316)*** - match by priority in vlan tag;
  - *mac-protocol* - match by mac protocol (skips vlan tags if any);
- **ip conditions**
  - *dst-address* - match by destination ip and mask;
  - *src-address* - match by source ip and mask;
  - *dscp* - match by ip dscp field;
  - *protocol* - match by ip protocol;
- **ipv6 conditions**
  - *dst-address6* - match by destination ip and mask;
  - *src-address6* - match by source ip and mask;
  - *flow-label* - match by ipv6 flow label;
  - *traffic-class* - match by ipv6 traffic class;
  - *protocol* - match by ip protocol;
- **L4 conditions**
  - *src-port* - match by tcp/udp source port range;
  - *dst-port* - match by tcp/udp destination port range;

IPv4 and IPv6 specific conditions cannot be present in same rule. Menu contains ordered list of rules just like in `/ip firewall filter`. Due to the fact that the rule table is processed entirely in switch chips hardware there is limitation to how many rules you may have. Depending on the amount of conditions (MAC layer, IP layer, IPv6, L4 layer) you use in your rules the amount of active rules may vary from 8 to 32 for Atheros8316 switch chip and from 24 to 96 for Atheros8327/QCA8337 switch chip. You can always do `/interface ethernet switch rule print` after modifying your rule set to see that no rules at the end of the list are 'invalid' which means those rules did not fit into the switch chip.

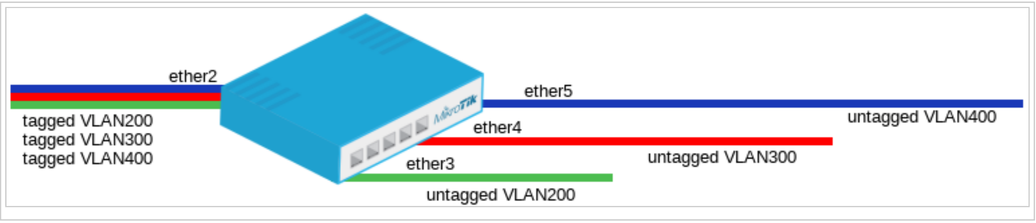
Setup Examples

Note: Make sure you have added all needed interfaces to the VLAN table when using secure vlan-mode. For routing functions to work properly on the same device through ports that use secure vlan-mode, you will need to allow access to the CPU from those ports, this can be done by adding the switchX-cpu interface itself to the VLAN table. Examples can be found at the Management port section.

Warning: When allowing access to the CPU, you are allowing access from a certain port to the actual router/switch, this is not always desirable. Make sure you implement proper firewall filter rules to secure your device when access to the CPU is allowed from a certain VLAN ID and port, use firewall filter rules to allow access to only certain services.

VLAN Example 1 (Trunk and Access Ports)

Routerboards with Atheros switch chips can be used for 802.1Q Trunking. This feature in RouterOS version 6 is supported by **QCA8337, AR8316, AR8327, AR8227** and **AR7240** switch chips.
In this example ether3,ether4 and ether5 interfaces are access ports, while ether2 is a trunk port. VLAN IDs for each access port: ether3 - 200, ether4 - 300, ether5 - 400.



- Create a group of switched ports by selecting one master-port and setting it for other ports.

```
# pre-v6.41 master-port configuration

/interface ethernet
set ether3 master-port=ether2
set ether4 master-port=ether2
set ether5 master-port=ether2

# post-v6.41 bridge hw-offload configuration
```

```
/interface bridge
add name=bridge1 igmp-snooping=no protocol-mode=none
/interface bridge port
add bridge=bridge1 interface=ether2 hw=yes
add bridge=bridge1 interface=ether3 hw=yes
add bridge=bridge1 interface=ether4 hw=yes
add bridge=bridge1 interface=ether5 hw=yes
```

- Add VLAN table entries to allow frames with specific VLAN IDs between ports.

```
/interface ethernet switch vlan
add ports=ether2,ether3 switch=switch1 vlan-id=200
add ports=ether2,ether4 switch=switch1 vlan-id=300
add ports=ether2,ether5 switch=switch1 vlan-id=400
```

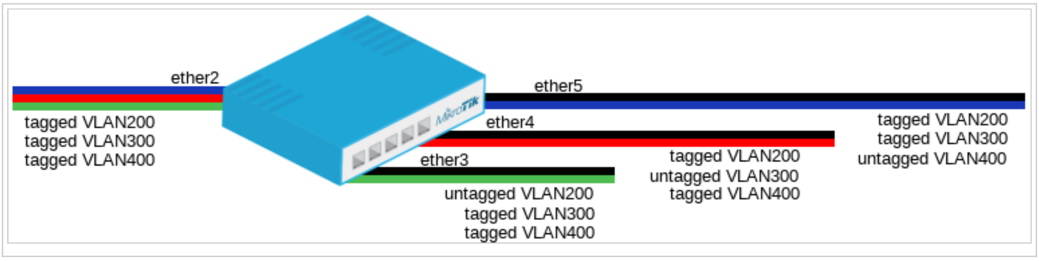
- Assign "vlan-mode" and "vlan-header" mode for each port and also "default-vlan-id" on ingress for each access port.

Setting "vlan-mode=secure" ensures strict use of VLAN table.
Setting "vlan-header=always-strip" for access ports removes VLAN header from frame when it leaves the switch chip.
Setting "vlan-header=add-if-missing" for trunk port adds VLAN header to untagged frames.
"Default-vlan-id" specifies what VLAN ID is added for untagged ingress traffic of the access port.

```
/interface ethernet switch port
set ether2 vlan-mode=secure vlan-header=add-if-missing
set ether3 vlan-mode=secure vlan-header=always-strip default-vlan-id=200
set ether4 vlan-mode=secure vlan-header=always-strip default-vlan-id=300
set ether5 vlan-mode=secure vlan-header=always-strip default-vlan-id=400
```

VLAN Example 2 (Trunk and Hybrid Ports)

VLAN Hybrid ports which can forward both tagged and untagged traffic are supported only by some Gigabit switch chips (**QCA8337, AR8327**)



- Create a group of switched ports.

```
# pre-v6.41 master-port configuration

/interface ethernet
set ether3 master-port=ether2
set ether4 master-port=ether2
set ether5 master-port=ether2

# post-v6.41 bridge hw-offload configuration

/interface bridge
add name=bridge1 igmp-snooping=no protocol-mode=none
/interface bridge port
add bridge=bridge1 interface=ether2 hw=yes
add bridge=bridge1 interface=ether3 hw=yes
add bridge=bridge1 interface=ether4 hw=yes
add bridge=bridge1 interface=ether5 hw=yes
```

- Add VLAN table entries to allow frames with specific VLAN IDs between ports.

```
/interface ethernet switch vlan
add ports=ether2,ether3,ether4,ether5 switch=switch1 vlan-id=200
add ports=ether2,ether3,ether4,ether5 switch=switch1 vlan-id=300
add ports=ether2,ether3,ether4,ether5 switch=switch1 vlan-id=400
```

- In switch port menu set "vlan-mode" on all ports and also "default-vlan-id" on planned hybrid ports.

"Vlan-mode=secure" will ensure strict use of VLAN table.
"Default-vlan-id" will define VLAN for untagged ingress traffic on port.
In Gigabit switch chips when "vlan-mode=secure", it ignores switch port "vlan-header" options. VLAN table entries handle all the egress tagging/untagging and works as "vlan-header=leave-as-is" on all ports.
It means what comes in tagged, goes out tagged as well, only "default-vlan-id" frames are untagged at the egress of port.

```
/interface ethernet switch port
set ether2 vlan-mode=secure
set ether3 vlan-mode=secure default-vlan-id=200
set ether4 vlan-mode=secure default-vlan-id=300
set ether5 vlan-mode=secure default-vlan-id=400
```

Management IP Configuration

This example will show one of the possible management IP address configurations. Management IP will be accessible only through trunk port and it will have a separate VLAN with ID 99.

- Add VLAN table entry to allow management traffic through switch-cpu port and the trunk port.

```
/interface ethernet switch vlan
add ports=ether2,switch1-cpu switch=switch1 vlan-id=99
```

- Configure the port which connects switch-chip with CPU, set "vlan-header=leave-as-is" because management traffic already should be tagged.

```
/interface ethernet switch port
set switch1-cpu vlan-mode=secure vlan-header=leave-as-is
```

- Add VLAN 99 and assign IP address to it. Since the master-port receives all the traffic coming from switch-cpu port, VLAN has to be configured on master-port, in this case "ether2" port.

```
/interface vlan
add name=vlan99 vlan-id=99 interface=ether2
/ip address
add address=192.168.88.1/24 interface=vlan99 network=192.168.88.0
```

Spanning Tree Protocol

Starting from RouterOS v6.38 RouterBoards support Spanning Tree Protocols on ports configured for switching. This feature is available on following switch chips: QCA8337; Atheros8327; Atheros8316; Atheros8227; Atheros7240. To enable this feature create RouterOS bridge interface and add the master-port to it.

- Create a group of switched ports

```
/interface ethernet
set ether2 master-port=ether1
set ether3 master-port=ether1
set ether4 master-port=ether1
```

- Create a bridge interface and add the master-port to it

```
/interface bridge add name=bridge1 protocol=rstp

/interface bridge port add bridge=bridge1 interface=ether1
```

- Slave ports are dynamically added to the bridge only to show STP status. Forwarding through switched ports still are handled by hardware switch chip.

```
[admin@MikroTik] > /interface bridge port print
Flags: X - disabled, I - inactive, D - dynamic
 #    INTERFACE           BRIDGE           PRIORITY  PATH-COST    HORIZON
 0    ether1              bridge1              0x80         10       none
 1 ID ether2              bridge1              0x80         10       none
 2  D ether3              bridge1              0x80         10       none
 3  D ether4              bridge1              0x80         10       none
```

Retrieved from "https://wiki.mikrotik.com/index.php?title=Manual:Switch_Chip_Features&oldid=30535"

Categories: Manual | Interface | Case Studies | Routerboard

- This page was last edited on 26 January 2018, at 17:13.