KSZ Switch Utilities User Guide

Rev 1.4

July 26, 2017

2180 Fortune Dr., San Jose, CA 95131, USA • (408) 944-0480 • http://www.microchip.com

Table of Contents

1 Revision History

Revision	Date	Summary of Changes
1.0	04/13/11	Initial revision.
1.1	08/04/11	Updated testing features in transmitting PTP messages.
1.2	08/08/12	Updated PTP APIs. Added 802.3 support.
1.3	09/26/14	Updated for KSZ956X switches.
1.4	07/26/17	Renamed to KSZ Switch Utilities User Guide.

2 Introduction

This document describes how to use different available utilities to verify the functions of Microchip KSZ switches. Some utilities can only be run on switches with certain capabilities such as 1588 PTP.

2.1 Glossaries

Following are acronyms and terms used throughout the document:

- Audio/Video Bridge AVB
- End-to-end E2E
- Peer-to-peer P2P
- Precision Time Protocol PTP
- Pulse per second PPS
- Inter-range Instrumentation Group IRIG
- Master Clock MC
- Ordinary Clock OC
- Slave Clock SC
- Transparent Clock TC
- High-availability Seamless Redundancy HSR
- Device Level Ring DLR

3 Using `regs_bin` Utility

The `regs_bin` program uses the standard Linux register access API to access the switch or device registers. As such it only works when such function is implemented in the switch or device driver.

The program requires the device name as input to determine which file to access the Linux register API. For Ethernet device with name like `eth0` the assumed location is `/sys/class/net/eth0`. For SPI device with name like `spi0.0` the location is `/sys/bus/spi/devices/spi0.0`. For I2C the full path has to be specified as the device name is generic. When the program can access the API it will present a prompt for user to enter commands.

The program will try to determine the width of register access by first reading a register with 0 byte size. The driver which supports this function will return the default register size in either 8-bit, 16-bit, or 32-bit. From then on the program knows the size of registers and accesses it in that size. There are commands to access registers in other widths. If this size is somehow not known then the number "`2`" or "`4`" can be added in the command to specify the register size.

The basic read command is "r <reg>" in which <reg> is the register to read and is in hexadecimal format. The program will display the value in the default register size: <reg>: xx in 8-bit, xxxx in 16-bit, or xxxxxxxx in 32-bit. A number can be added after the register to indicate reading the next registers so many times. The program then displays in the register values in a format that fits the screen. This is useful to dump some registers for debug purpose.

The write command is "w <reg> <val>" in which < reg> is the register to write the <val> to. Both are in hexadecimal format. It is possible to continue adding values like "w <reg> <val> <val> .. <val>" so the values are written to registers consecutively.

The "r" and 'w" commands can be appended with "b" to indicate 8-bit; "w", 16-bit; and "d", 32-bit. This is useful if certain registers are defined in that format.

For 16-bit and 32-bit register accesses the program can automatically selects the width if the specified register is not in the correct boundary. So "r 1" will automatically reads the register in 8-bit. For 32-bit access the command "r 3" will read the register in 8-bit; "r 2", 16-bit. For some register accesses that require exact 32-bit size it is not possible to limit the size to less than that like "rb 1 2."

The "q" command is used to quit the program.

4 Using kszsw Utility

This utility is in development and is designed to configure the switch using standard commands. Its functions can also be accessed by a corresponding remote program so that the switch can be configured remotely. The primary function right now for this utility is to configure the Device Level Ring (DLR) function of the switch.

The Ethernet device is required as input to the program: kszsw eth0 or kszsw br0.

The current implemented operations for basic switch functions are

```
l <p> [0,1]    # turn on/off port learning
r <p> [0,1]    # turn on/off port rx
t <p> [0,1]    # turn on/off port tx
s <p> [0,1]    # turn on/off port PHY power
```

The DLR functions can be selected with the "dlr" command if DLR is available.

The HSR functions can be selected with the "hsr" command if HSR is available.

The "sw" command is used to go back to the basic switch function menu.

The operations for the DLR functions are

```
v          # report revision
c          # report capabilities
t          # report topology
n          # report network status
s          # report node status
r          # report ring participant count
l          # report ring participant list
e <p>      # get active not at port
d          # report active supervisor address
p          # report active supervisor precedence
ib [#]     # get or set beacon interval
tb [#]     # get or set beacon timeout
pb [#]     # get or set precedence
vb [#]     # get or set VLAN id
a          # report all
C [#]      # get or set supervisor configuration
F [#]      # get or set ring fault count
V          # verify fault
R          # clear rapid fault
S          # restart SignOn process
G          # clear gateway fault
```

The operations for the HSR functions are

```
c          # report capabilities
r          # report ring participant count
l          # report ring participant list
```

5 Using `ptp_cli` Utility

The `ptp_cli` and `ptp` programs described below are for use with devices with 1588 PTP capability.

The `ptp_cli` program is a command line utility to control the PTP hardware. It does not have any argument.

As the backspace character does not translate well in some terminal programs, modifying the command line is not possible. As a result the program tries to minimize the length of command for less typing so that fewer errors are produced. If something is wrong in the command, just hit Enter to let the program decide to execute it or not.

For this reason some commands requiring many parameters can be entered with a few required

ones. The rest are stored in variables that can be manipulated independently, as normally those parameters do not vary too often.

The commands will be described in sections.

Commands	
?e ...	Timestamp input commands.
?o ...	Trigger output commands.
?c ...	Clock commands.
?d ...	Delay commands.
mci [id]	Set master clock identity. The format is 01:23:45:FF:FE:67:89:AB. It is used by the programs to filter out messages. Sync and Follow_Up messages are sent using this clock identity.
sci [id]	Set slave clock identity. The format is 01:23:45:FF:FE:67:89:AB. This is the identity of the target device. It is used by the program to filter out messages.
h	Display command help messages.
q	Quit the program.

The flags used in timestamp event and trigger output are defined as followed:

Command Flags	**Values**	**Meaning**
PTP_CMD_INTR_OPER	0x01	Enable interrupt.
PTP_CMD_SILENT_OPER	0x02	Silent output.
PTP_CMD_ON_TIME	0x04	Operate exactly on trigger time.
PTP_CMD_REL_TIME	0x08	Trigger time is relative.
PTP_CMD_CLK_OPT	0x10	Enable clock option.
PTP_CMD_CASCADE_RESET_OPER	0x40	Used for testing only.
PTP_CMD_CANCEL_OPER	0x80	Cancel operation.

The PTP_CMD_INTR_OPER flag is normally on for both timestamp event and trigger output.

The PTP_CMD_SILENT_OPER flag can be used for timestamp event for internal use only. It is used in trigger output to not display anything when the output is done.

The PTP_CMD_ON_TIME and PTP_CMD_REL_TIME flags are only used in trigger output.

The PTP_CMD_CANCEL_OPER flag is used to cancel operation.

The second character of timestamp input detection commands is “e.”

Commands	
`de tsi gpi event [total] [flags] [timeout]`	Enable event detection on GPIO input pin `gpi` starting at event unit `tsi`. Optional parameters include `total`, which is 1, `flags` that is stored in the rx flags variable, and `timeout` that is stored in the rx timeout variable.
`ee tsi`	Cancel event detection with event unit `tsi`.
`ge tsi`	Retrieve events from event unit `tsi`.
`pe tsi`	Poll hardware for events from event unit `tsi`.
`me [timeout]`	Change rx timeout value.
`re [flags]`	Change rx flag value.
`te gpi`	Convenient event detection setup on GPIO input pin `gpi`. Used for testing.
`ae`	Analyze events received from `te`.

Valid parameter for GPIO input pin is less than 12, although not all of them are usable. Valid parameter for event unit is less than 12, but 10 is reserved to use in the driver. Parameter event can be zero for falling edge detection, 1 for rising edge detection, and others for both edges detection.

Specifying parameter total with a number greater than 1 enables timestamp input in cascade mode. The number of units assigned matches the total.

The parameter flags normally is 1 to indicate interrupt is used to report the event as soon as it happens.

The parameter timeout is used to show how long the driver should wait to disable the event unit after the first event happens. This is used to capture the second or more events in the unit. Specifying a value of zero means the detection is repeatable. After the first event the driver automatically setup the unit to detect another event.

If the command is successful the first unit used is displayed. Otherwise an error message is displayed. If the command fails several times, then it is necessary to use the command “`ee`” to cancel the event.

If interrupt is enabled, the program will show the events as they happen. Otherwise, use the “`ge`” command to retrieve the events stored in the unit.

The command “`pe`” is used to poll the hardware for new events. It currently is not necessary to use this command unless the PTP timing interrupt is not running.

The "te" command is a convenient way to setup several units for testing. It setup units 11, 0, and 1 to detect falling edge; and 2 to 6 for rising edge.

The format of the event is

```
unit: <tsi>=<event>
<#>=[0,1] <sec>:<nanosec>
<#>=[0,1] <sec>:<nanosec>
```

The "ae" command can be used to analyze the events. Given the events received from the command "te" or a both edge detection the up and down durations of the signal can be calculated.

The second character of trigger output commands is "o."

Commands	
to tso gpo event [pulse] [cycle] [cnt] [iterate] [sec] [nsec] [flags]	Enable trigger output on GPIO output pin gpo using output unit tso. Other parameters not required to be entered are pulse, which comes from either tx_pulse or tx_pattern variable; cycle, tx_cycle; cnt, tx_cnt; iterate, cascade_iterate; sec, tx_sec; nsec, tx_nsec; and flags, tx_flags.
oo tso	Cancel and reset output.
ao tso gpo total [flags]	Begin cascade mode on GPIO output pin gpo using total units that start with unit tso.
bo tso gpo total [cnt] [flags]	Enable cascade mode on GPIO output pin gpo using total units that start with unit tso. Other parameters required are cnt, which comes from cascade_cnt variable; and flags, tx_flags.
do tso gpo total	Cancel cascade mode.
co [tx_cnt]	Change output count.
yo [tx_cycle]	Change output cycle time in nanosecond.
po [tx_pattern]	Change output pattern in hexadecimal format.
uo [pulse]	Change output pulse time in nanosecond.
so [sec]	Change output second target time.
no [nsec]	Change output nanosecond target time.
vo [cascade_iterate]	Change iteratation time in nanosecond. Used only in cascade mode.
wo [cascade_gap]	Change gap time between units. Used only in cascade mode.
xo [cascade_cnt]	Change output count in cascade mode.
ro [tx_flags]	Change tx flags value.

io	Display output variables.
irig code	Output IRIG time code on GPIO 1 for demonstration and testing only.
go gpo	Show output units that have raised level to high.

The following trigger output patterns are supported:

Output Pattern	Value	Output Signal
TRIG_NEG_EDGE	0	Negative edge—a falling edge from high to low
TRIG_POS_EDGE	1	Positive edge—a rising edge from low to high
TRIG_NEG_PULSE	2	Negative pulse—falling edge then rising edge after pulse time
TRIG_POS_PULSE	3	Positive pulse—rising edge then falling edge after pulse time
TRIG_NEG_CYCLE	4	Negative cycle—falling edge then rising edge after pulse time and stay high through cycle time
TRIG_POS_CYCLE	5	Positive cycle—rising edge then falling edge after pulse time and stay low through cycle time
TRIG_REG_OUTPUT	6	Register bit pattern output—0 to indicate low and 1 to indicate high

Pulse time is required for outputs TRIG_NEG_PULSE, TRIG_POS_PULSE, TRIG_NEG_CYCLE, and TRIG_POS_CYCLE. Pulse time should be multiple of 8 ns, as the unit used in the hardware is 8 ns. The maximum pulse time is 524280 ns.

Cycle time is required for outputs TRIG_NEG_CYCLE and TRIG_POS_CYCLE. The cycle time includes the pulse time and so should be longer than the pulse time. It should be at least 56 ns longer because of hardware constraint. The minimum cycle time is 40 ns. For output TRIG_REG_OUTPUT cycle time is used as an interval between the bits in the pattern. The size of the pattern is 16-bit.

For TRIG_NEG_CYCLE, TRIG_POS_CYCLE, and TRIG_REG_OUTPUT outputs that require cycle time the output can be repeated. For the others the output is one-shot. A repeat count of 0 means the output is running infinitely.

To repeat one-shot outputs or generate more complex signals cascade mode can be used. In cascade mode several output units are grouped to perform an output signal in sequence.

The output units in cascade are separated by an iteratation time. When these iteratation times in output units are the same doing the cascaded output repeatedly has no concern. As this iteratation time is configured separately for each output unit and can be different, care must be exercised to not repeat the cascade output too much lest the individual outputs overlap each

other.

The target time specified by second and nanosecond is absolute. However, for ease of use the second can be used relatively. So instead of specifying a very big second value, a very small value can be used to indicate a few seconds after the current second. This is done by setting the `PTP_CMD_REL_TIME` flag in the tx flags variable. The `PTP_CMD_INTR_OPER` flag is again used to enable interrupt so that the output unit can be stopped and released right after the operation is completed.

The cascade output needs to be preceded by a cascade init command, followed by several trigger output commands, then finally a cascade enable command.

Example:

```
ao 1 0 3
to 1 0 5
to 2 0 3
to 3 0 5
bo 1 0 3
```

The cascade init command tells the driver how many units are used in the cascade mode. The following trigger output commands configures the output units but leave the target time unset. The cascade enable command finally links the units together and setup the target time for each unit and starts the command.

For convenience the target time is automatically increased for each trigger output, the amount being the value stored in the cascade gap variable. As a result the trigger output commands need to be entered in order. Otherwise the command will fail.

All outputs not run in cascade mode are run in combined mode. That means all output are ORed. Any unit that holds the level high will prevent the other units from changing the signal. Sometimes user may forget which unit has put the level to high. The command "`go`" is useful in this case as the driver reports back what it thinks the units executing output that leaves the level high. User then can use the command "`oo`" to reset the unit. The register read command "`r 680`" can be used to report the actual GPIO status.

The second character of clock commands is "c."

Commands	
`gc`	Get current clock.
`sc sec [nsec]`	Set clock with seconds and optional nanoseconds.
`ic nsec [sec]`	Increment clock by nanoseconds and optional seconds.
`dc nsec [sec]`	Decrement clock by nanoseconds and optional seconds.
`ac drift [interval]`	Set the continual clock adjustment register with drift value. The drift

	value can be negative. Default interval is 1 second.

The second character of delay commands is “d.”

Commands	
`gd port`	Get port delays.
`sd port rx tx asym`	Set port delays of rx latency, tx latency, and asymmetric delay.
`pd port [delay]`	Get port peer delay or set it with delay value.

The hardware setting commands are mostly related to PTP message filtering.

Commands	
`d [#]`	Change hardware domain number.
`a [0,1]`	Change alternate master setting.
`c [0,1]`	Change domain check setting.
`e [0,1]`	Change P2P clock setting.
`m [0,1]`	Change master clock setting.
`p [0,1]`	Change 2-step clock setting.
`u [0,1]`	Change unicast setting.
`has [0,1]`	Change 802.1AS setting.
`hds [0,1]`	Change drop Sync setting.
`huc [0,1]`	Change UDP checksum setting.
`hda [0,1]`	Change Delay_Req association setting.
`hpda [0,1]`	Change Pdelay_Req association setting.
`hsa [0,1]`	Change Sync association setting.
`hp [0,1]`	Change message priority setting.
`r reg`	Read from register. Number is in hexadecimal format.
`w reg val`	Write value to register. Numbers are in hexadecimal format.

Entering a command without a parameter will return the current value of that setting. Note the value is the one maintained by the program, not read from hardware. So if some other programs change hardware settings in the background the value will not be accurate. The only program that runs in background is the PTP stack. In this case this program should not change any setting

that affects PTP message processing. The program reads all hardware settings when started.

6 Using `ptp` Utility

The `ptp` program is a simple utility to allow sending PTP messages for hardware testing. It sends PTP messages in IPv4 or IPv6 packets, and also in 802.3 frames. Normally it sends multicast packets, but it can also send unicast packets.

It can send the following PTP messages: Sync, Follow_Up, Delay_Req, Delay_Resp, Pdelay_Req, Pdelay_Resp, Pdelay_Resp_Follow_Up, and Announce.

It can send some sample Management and Signaling messages just for testing purpose.

It has a simple response mechanism that upon receiving a Delay_Req message it will send a Delay_Resp message. When it receives a Pdelay_Req message it will send Pdelay_Resp message and Pdelay_Resp_Follow_Up message if specified.

The program accepts the following parameters when started.

<local_if> is required. Normally it will be `eth0` for the default interface. Other network interfaces such as `eth1` can be used.

The parameter "`-6`" is used to specify IPv6.

The parameter "`-p`" is used to specify 802.3.

The parameter "`-u`" is used to specify unicast packet. The IP address followed should specify the destination address. For 802.3 the destination address is a MAC address. PTP messages will be sent in unicast packets.

The parameter "`-d[#]`" is used to debug received messages. A number after it indicates the debug level. A number of 5 will show all debug statements.

The parameter "`-l`" is used to enable multicast loop. The sending program will receive the same multicast packet looped back to the system. This is used for testing.

The program will automatically determine the IPv4 and IPv6 address used by the network interface.

After the program starts the following commands can be used:

Commands	
`m [h#]`	Send Sync message.
`n [h#]`	Send Follow_Up message.

a [h#]	Send Delay_Req message.
b [h#]	Send Delay_Resp message.
x [h#]	Send Pdelay_Req message.
y [h#]	Send Pdelay_Resp message.
z [h#]	Send Pdelay_Resp_Follow_Up message.
e [h#]	Send Announce message.
mg	Send Management test messages.
sg	Send Signaling test messages.
t [0,1]	Set 2-step flag: 0 for 1-step; 1 for 2-step. It is used by the program to decide whether to send Pdelay_Resp_Follow_Up message upon receiving Pdelay_Req message.
v [0,1]	Set alternate master flag.
c [#]	Change correction field. Used for testing.
d [#]	Change domain number.
s [#]	Change source port field. Used for testing.
u [#]	Change reply count. Used for testing.
p [#]	Change destination port. This only applies when running in the the target system. Number 1 indicates port 1; number 2, port 2; and number 3, both ports. For KSZ956X switches with more than 2 ports the number can be in hexadecimal like 0x3f to indicate 6 ports.
w [0,1]	Send TLV in Follow_Up and Announce messages.
mci [id]	Set master clock identity. The format is 01:23:45:FF:FE:67:89:AB. It is used by the programs to filter out messages. Sync and Follow_Up messages are sent using this clock identity.
sci [id]	Set slave clock identity. The format is 01:23:45:FF:FE:67:89:AB. This is the identity of the target device. It is used by the program to filter out messages.
h	Display command help messages.
q	Quit the program.

The commands m, n, a, b, x, y, z, and e are used to send regular PTP messages. The sequence id numbers are updated appropriately. Entering a hexadecimal number following the command will manually increase the sequence number by that number. This is used for testing.

The command u indicates how many replies the program sends back after receiving a request message. The default is 1. A count of 2 will send two replies with the first one's sequence id one less than the sequence id of the request. A count of more than 2 will have the sequence id increased by one after the second reply. Set it to 0 to disable responding.

The command mg sends some sample Management messages for testing.

The command `sg` sends some sample Signaling messages for testing.

The command `w` adds a TLV after the Follow_Up and Announce messages.

The other commands affect the contents of the messages.

For ease of changing hardware to filter PTP messages a set of commands from the `ptp_cli` utility is duplicated in the `ptp` utility. To access them type the command "`hw`" to switch to those commands. Type "`sw`" to switch back to original commands.

These hardware commands involve the hardware clock:

Commands	
`gc`	Get current clock.
`sc sec [nsec]`	Set clock with seconds and optional nanoseconds.
`ic nsec [sec]`	Increment clock by nanoseconds and optional seconds.
`dc nsec [sec]`	Decrement clock by nanoseconds and optional seconds.
`ac drift [interval]`	Set the continual clock adjustment register with drift value. The drift value can be negative. Default interval is 1 second.

These hardware commands involve timestamp delays:

Commands	
`gd port`	Get port delays.
`sd port rx tx asym`	Set port delays of rx latency, tx latency, and asymmetric delay.
`pd port [delay]`	Get port peer delay or set it with delay value.

The hardware setting commands are mostly related to PTP message filtering.

Commands	
`d [#]`	Change hardware domain number.
`a [0,1]`	Change alternate master setting.
`c [0,1]`	Change domain check setting.
`e [0,1]`	Change P2P clock setting.

`m [0,1]`	Change master clock setting.
`p [0,1]`	Change 2-step clock setting.
`u [0,1]`	Change unicast setting.
`has [0,1]`	Change 802.1AS setting.
`hds [0,1]`	Change drop Sync setting.
`huc [0,1]`	Change UDP checksum setting.
`hda [0,1]`	Change Delay_Req association setting.
`hpda [0,1]`	Change Pdelay_Req association setting.
`hsa [0,1]`	Change Sync association setting.
`hp [0,1]`	Change message priority setting.
`r reg`	Read from register. Number is in hexadecimal format.
`w reg val`	Write value to register. Numbers are in hexadecimal format.

Entering a command without a parameter will return the current value of that setting. Note the value is the one maintained by the program, not read from hardware. So if some other programs change hardware settings in the background the value will not be accurate. The only program that runs in background is the PTP stack. In this case this program should not change any setting that affects PTP message processing. The program reads all hardware settings when started.