TI81XX PSP ETHERNET Switch User Guide

TI81XX PSP Ethernet Switch User Guide

Linux PSP

Important This document is applicable only for TI814X and TI813X

Introduction

This user guide provides an overview of the Common Platform SWitch (CPSW) driver for the TI814X and TI813X devices. The CPSW or Ethernet Switch driver follows the standard Linux network interface architecture.

The driver supports the following features:

1. 10/100/1000 Mbps mode of operation.
2. Auto negotiation.
3. Support for multicast and broadcast frames.
4. Linux NAPI support
5. Second switch port (from DEV_DM814XPSP_04.01.00.06 release)
6. IEEE 1588/802.1AS PTP Support (Supported from DEV_DM814XPSP_04.01.00.06 release)
7. VLAN (Subscription common for all ports) (Supported from DEV_DM814XPSP_04.01.00.06 release)
8. Ethertool (Supports only Slave 0) (Supported from DEV_DM814XPSP_04.01.00.06 release)
9. Switch mode of operation (Supported from DEV_DM814XPSP_04.01.00.07 release)
10. Dual Standalone EMAC mode (Supported from DEV_TI81XXPSP_04.04.00.01 release)

IEEE 1588/802.1AS PTP Support

Introduction

Precise time information is especially important for distributed systems. With the Precision Time Protocol (PTP) described in IEEE 1588/802.1AS, it is possible to synchronize distributed clocks with an accuracy of less than 1 microsecond via Ethernet networks for the very first time. The PTP protocol relies on establishing the time difference between connected components based on timestamped packets being exchanged between the master and slave. For better accuracy, the time-stamp needs to be generated as close to the wire(in HW) as possible. The PTP spec also provides guidelines about synchronization of clocks once the difference is calculated based on the messages exchanged.

For synchronization of the clocks we need a mechanism for

- Time-stamp generation for PTP messages - as close to the wire for better accuracy
- Mechanism for modifying the local clock once the difference in time is computed

For best possible results, we will need HW assist for both of the steps mentioned above. Relatively simpler systems can rely on SW only implementation where the timestamps on PTP messages can be generated in SW(by the driver, when en-queuing message with HW) based on kernel timer source(gettimeofday) and also use setttimeofday() to modify the local clock rate periodically.

Support for IEEE 802.1AS

The DM814X and DM813X series of devices has support for the IEEE 802.1AS standard. This is a profile of IEEE 1588-2008 and runs on Layer 2 only. This support is implemented in a new subsystem called the Common Platform Time Stamp (CPTS).

Note: Note IEEE 1588 on Layer 3 is not supported.

Note: Note 802.1AS does not currently work with an AVB switch (as of March-2012).

Software Time stamping

Software only implementation will rely on the kernel timer for both time-stamp generation and gettimeofday/settimeofday() calls for clock synchronization. Due to variable latency associated with handled this in Software, this will not yield the best results but is the ideal starting point to put together a working system that can be pruned later(with addition of Hardware timestamps and clock control APIs).

Salient features in this mode are

- Local clock source is got through the clock_gettime with the CLOCK_REALTIME and the clock rate set is done using the settimeofday api for resetting the clock to sync to the master clock.
- Time stamping of the packets are done through the socket option SO_TIMESTAMP through software kernel timer.

Approach

1. Download PTPdv2 source from the source forge
2. PC setup : build for x86 target, no modifications required. Will use SO_TIMESTAMP socket option for generating timestamps in software and use local REALTIME clock source.
3. DM814X EVM setup : needs cross compilation for arm architecture
4. PTPdv2 mods: Applied patch to use new POSIX clock API, PTP hardware clock framework and to use hardware SO_TIMESTAMPING interface for time stamp generation.
5. Kernel modification : Added a syscall "sys_clock_adjtime" for the arm architecture.
6. Test with ptpdv2 in master mode on PC and slave mode on DM814X EVM
7. Measure time difference and re-sync time and accuracy after re-sync operation (results summarized)

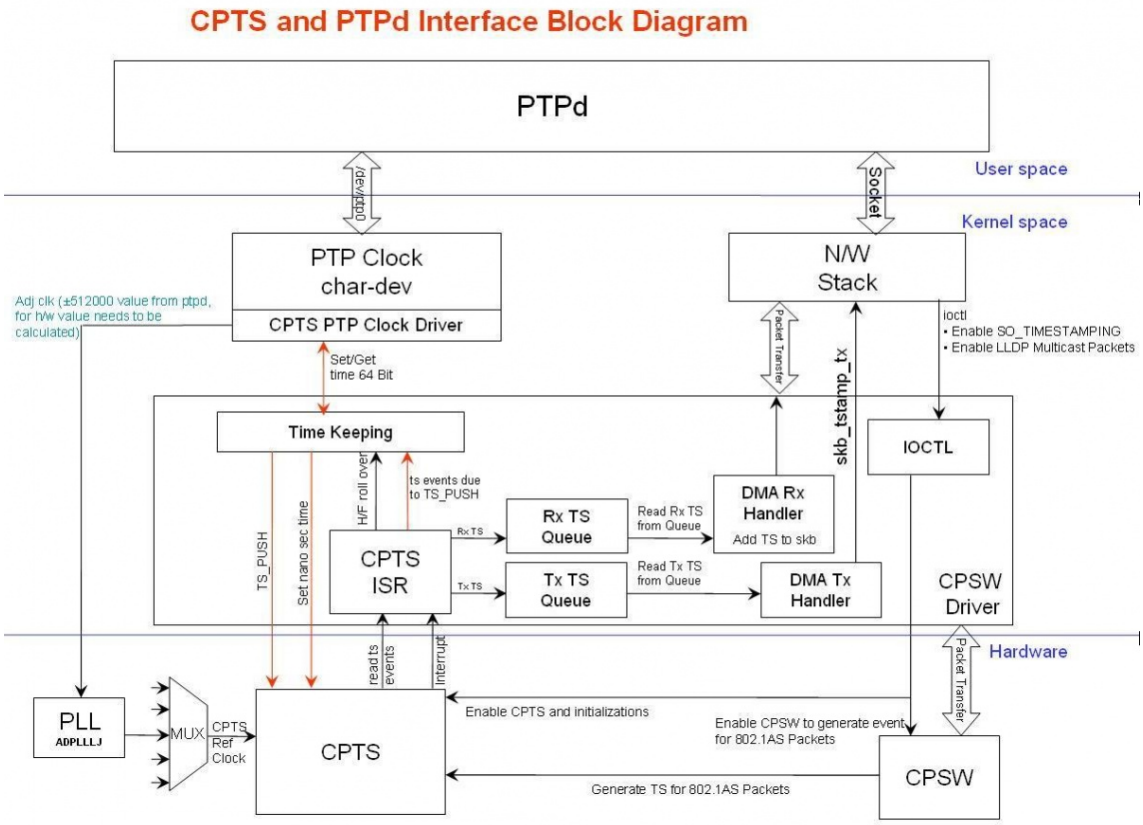
Hardware Time stamping

Fully h/w supported implementation shall exploit h/w assist for timestamp generation and special capabilities exported by the h/w for clock synchronization.

- Time stamping of the packets are done through the hardware by the SO_TIMESTAMPING socket option.
- The local hardware clock source control is achieved by implementing a new driver that fits into the PTP HW clock infrastructure added in Linux.
- Additional functionality can be provided through the custom ioctl calls exported by the clock driver module.
- Modifying the PTPdv2 source to support 802.1AS IEEE 1588v2 packets.

Capabilities supported in Hardware

- 32 Bit Nano Clock is supported, Above 32 bit host needs to maintain in software.
- Generate time-stamp for 802.1AS packets with configurable event type messages and VLAN type.
- 4 hardware triggered time-stamp events to register a hardware event occurance.



Approach

1. Choose appropriate clock as cpts ref clock in UBoot.
2. Modified the CPSW driver to enable CPTS module and to generate the H/W timestamp for 802.1AS packets.
3. Implemented the PTP class driver for Enabling the CPTS as Hardware clock source.
4. Implemented the Timekeeping module to hold the upper 32 bit of the 64 bit time in Software
5. Implemented the software queue to hold the timestamps and deliver it to the DMA Handlers.
6. Implemented ioctl call to support start and stop hw time stamping.
7. Implemented set_multicast_list receiving LLDP multicast packets.
8. Implemented the adjust clock frequency to change the CPTS ref clock
   - Input ref clock = CLCKINP (20Mhz)
   - Target rate = CLKOUT
   - DCO clock freq = DCOCLK
   - multiplier = M, pre-divider = N, post-divider= M2, fract_mult = Fm
   - CLKOUT = CLKINP*(M+Fm)/(N+1)*M2
   - DCOCLK = CLKOUT*M2
   - REFCLK = CLKINP/(N+1)
9. Modified the PTPd2 source from source forge to support 802.1AS protocol by the reference from the source from code.google.com as it supports 802.1AS but doesn't support POSIX hardware clock.
10. DM814X EVM setup : needs cross compilation for arm architecture
11. Test with ptpdv2 in master mode on one DM814X EVM and slave mode on another DM814X EVM

Source code

EVM

- Kernel : http://arago-project.org/git/projects/linux-omap3.git
- PTPd :

Host PC

- PTPd :

Setup and Testing

Initial testing (Software time stamping) has been carried with DM814X EVM(will act as slave,ordinary clock) connected to Linux PC(acting as master). PTPdv2 is used at both ends with appropriate command switches as described in the "TestResults" section.

Hardware time stamping testing is carried out with two DM814X EVMs, One acting as master and another as slave connected to the same Switch.

Compilation

Linux PC (Software Time stamping only)

Host PC Configuration

- Ubuntu 10.04.1

1. tar xzvf ptpdv2-google.tar.gz
2. cd ptpdv2-google
3. make

DM814X/DM813X EVM

Kernel

1. Defconfig
    1. DM814X - make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- ti8148_evm_defconfig
    2. DM813X - make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- dm385_evm_defconfig
2. make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- menuconfig
    1. Select PPS Support - Device Drivers ---> PPS Support ---> <*> PPS Support
    2. Select PTP Clock Support - Device Drivers ---> PTP Clock Support ---> <*> PTP Clock Support
    3. Select TI CPTS PTP Clock - Device Drivers ---> PTP Clock Support ---> <*> TI CPTS as PTP clock
3. make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- uImage
4. make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- headers_install

Constraint

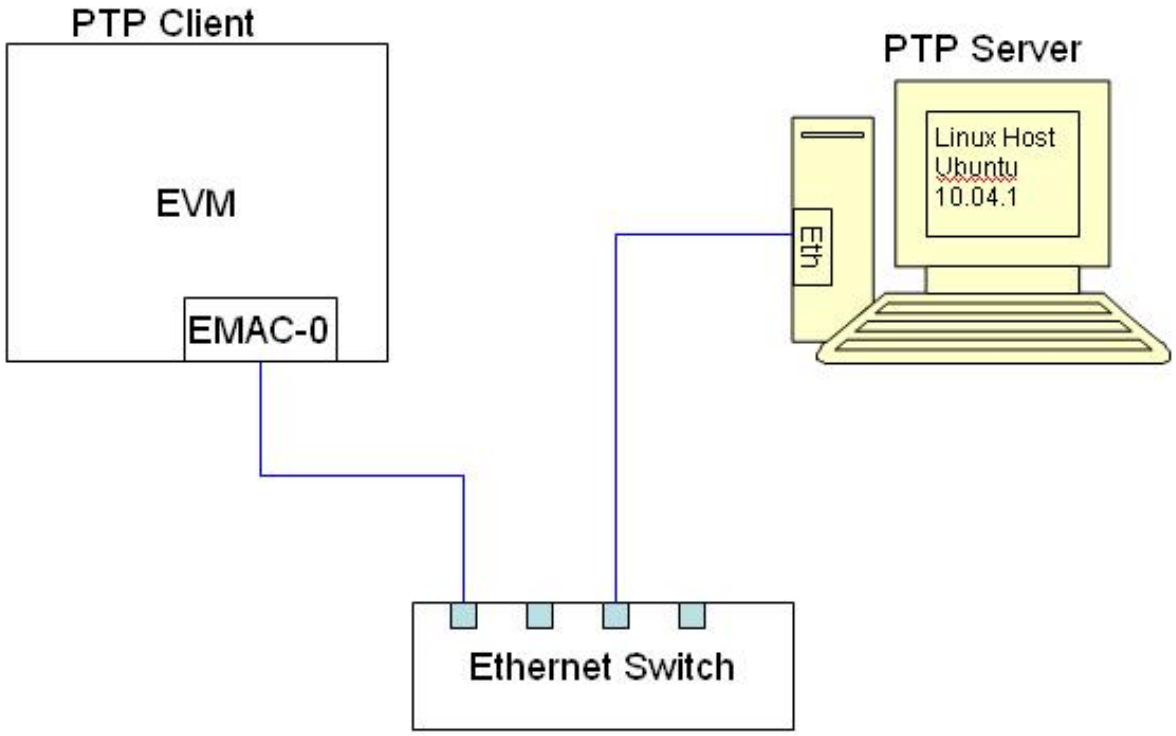
- CPTS ref clock of 250MHz is derived from PLL_AUDIO_OUT ADPLL, so make sure that this clock is not used by any other Subsystem or choose an appropriate clock source from the RMII_REFCLK_SRC register.
- Hardware time-stamping is done only for the Layer 2 Multicast or Unicast PTP packets and not on IP based PTP packets.

PTPdv2

1. tar xzvf ptpdv2.tar.gz
2. cd ptpdv2/src
3. make CC=arm-none-linux-gnueabi-gcc KBUILD_OUTPUT='kernel directory'

Test Results

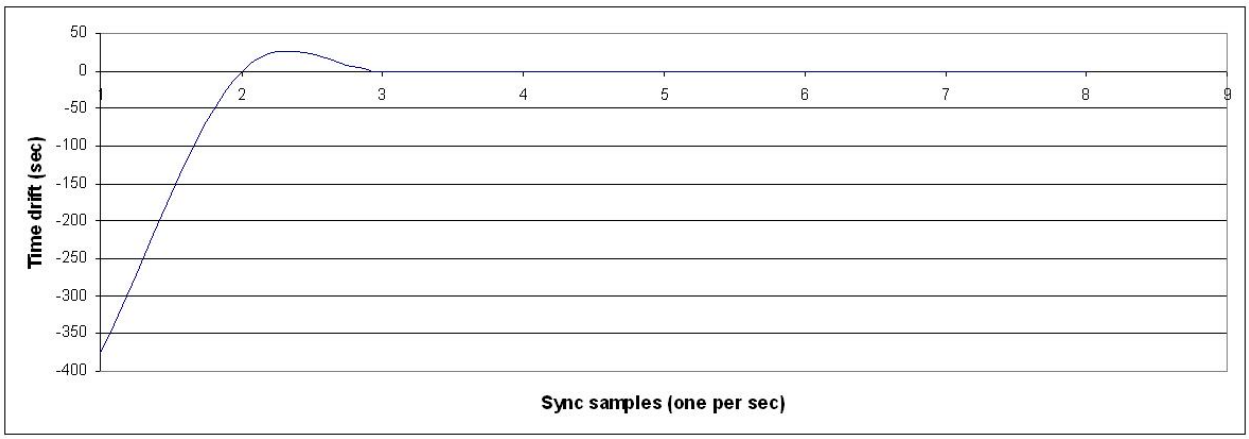
Software Time stamping



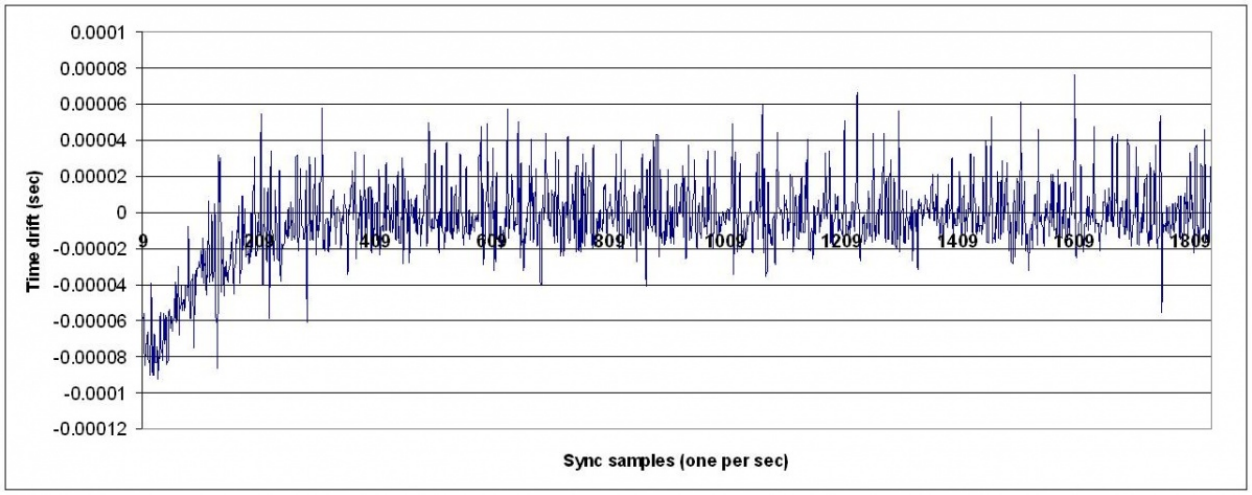
Client with reset clock

Execution steps

- On server side: sudo ./ptpv2d -c -d -t -y 2 -2 -8
- On client side : ./ptpd2 -c -e -g
- PTP Log during clock reset:



- PTP Log after a clock reset:

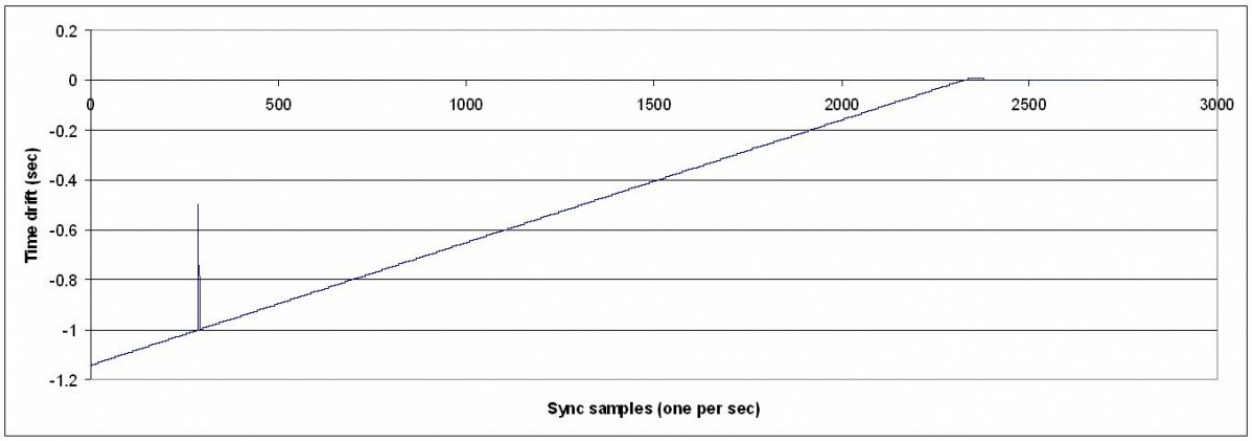


In summary the clocks can be synchronized with 100usecs accuracy. This should be much better if either/both ends start using HW generated timestamps for PTP messages and this will be taken up as the immediate next step.

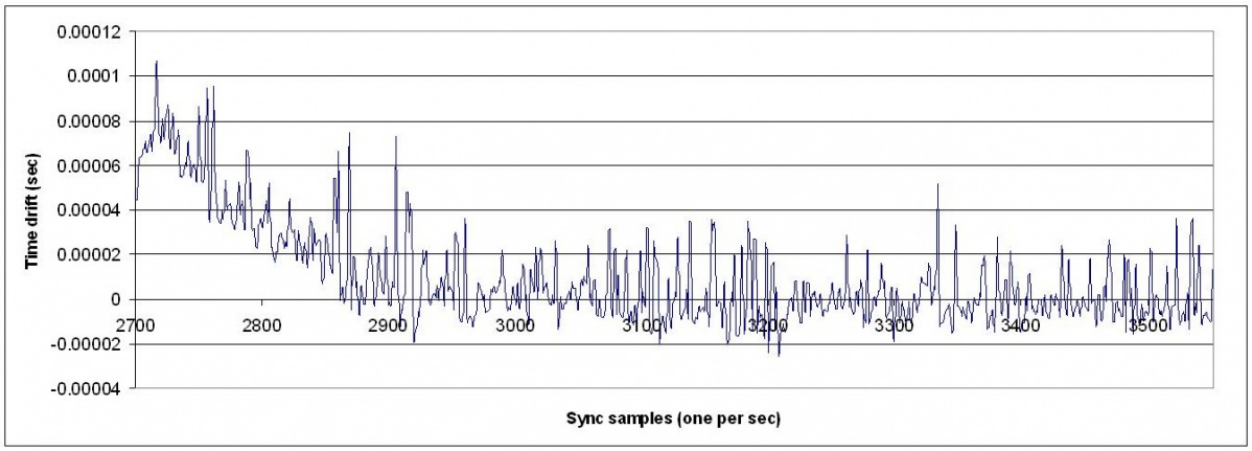
Client without reset clock

Execution steps

- On server side: sudo ./ptpv2d -c -d -t -y 2 -2 -8
- On client side : ./ptpd2 -c -e -g -x
- PTP Log without clock reset



- PTP Log without clock reset



Hardware Time stamping

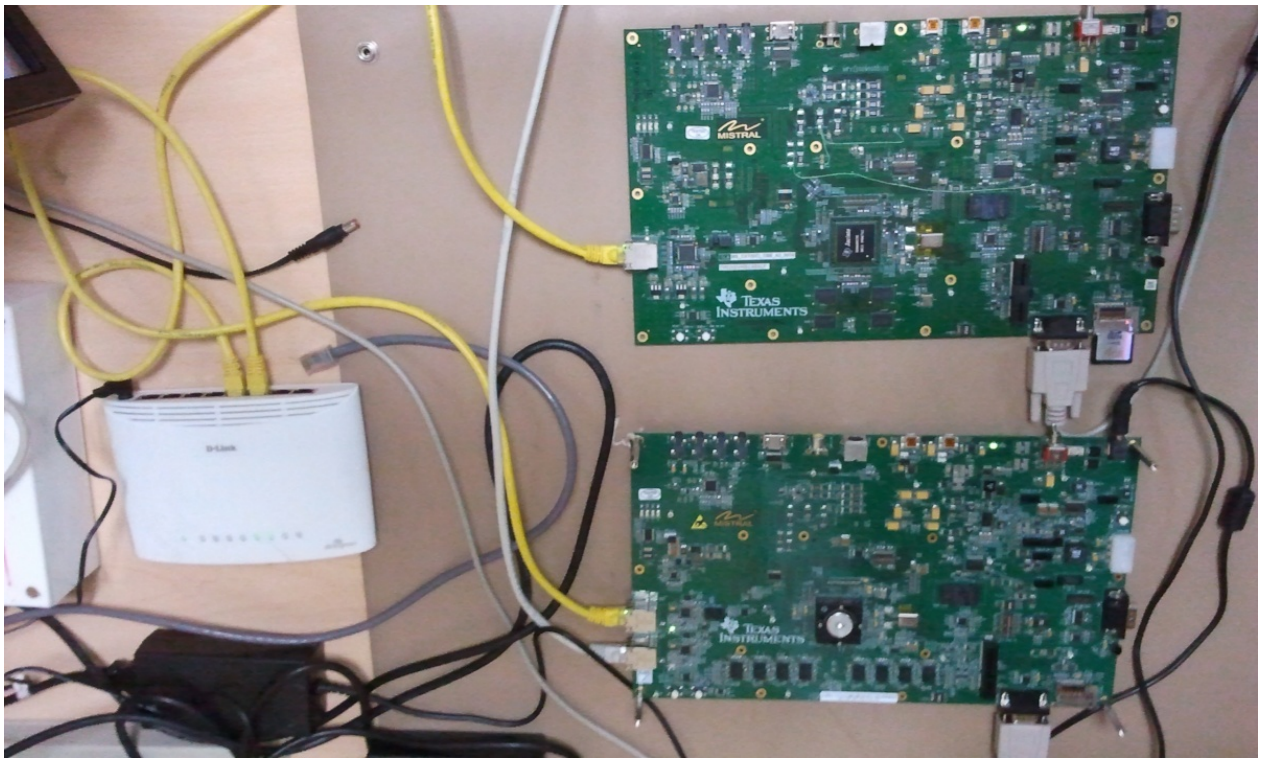
Execution steps

- On server side: ./ptpd2 -ce
- On client side: ./ptpd2 -ceg

Board Details

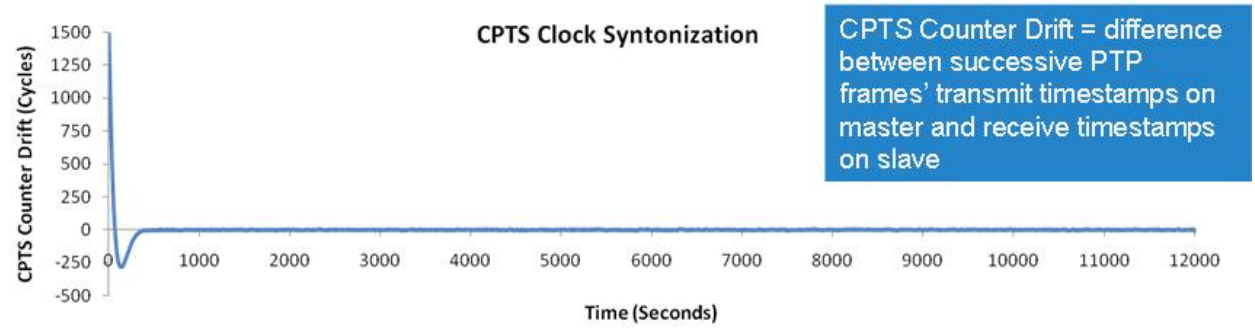
- PTP Server - DM814X 1.0 EVM
- PTP Client - DM814X 2.1 EVM

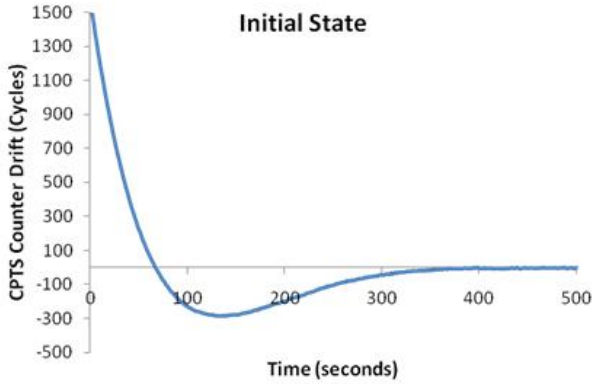
PTP Test Setup

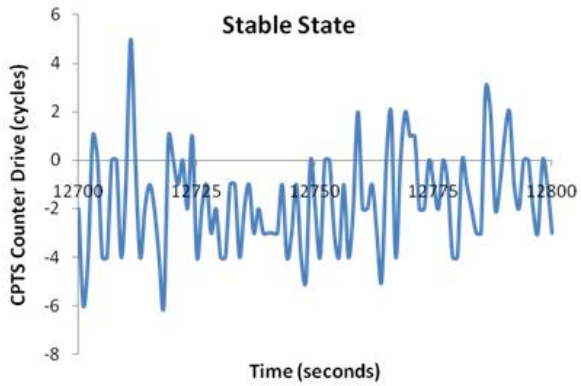


PTP Logs

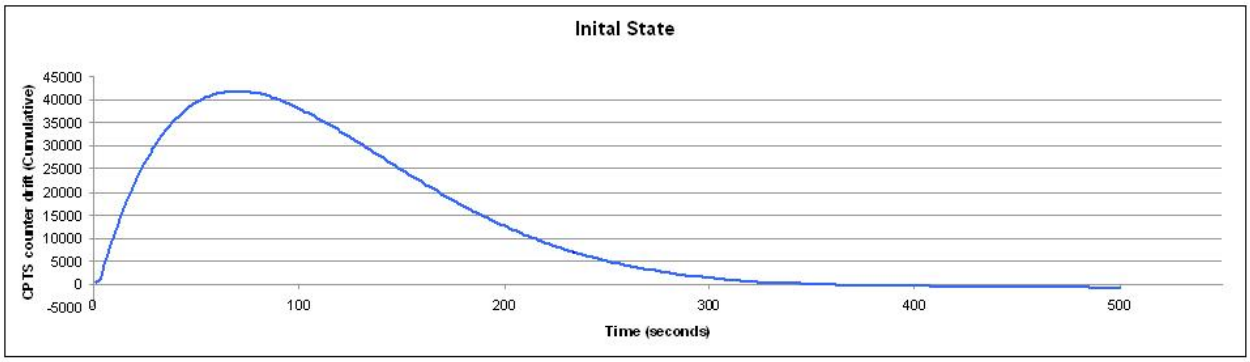
PTP Log Counter/Frequency Drift

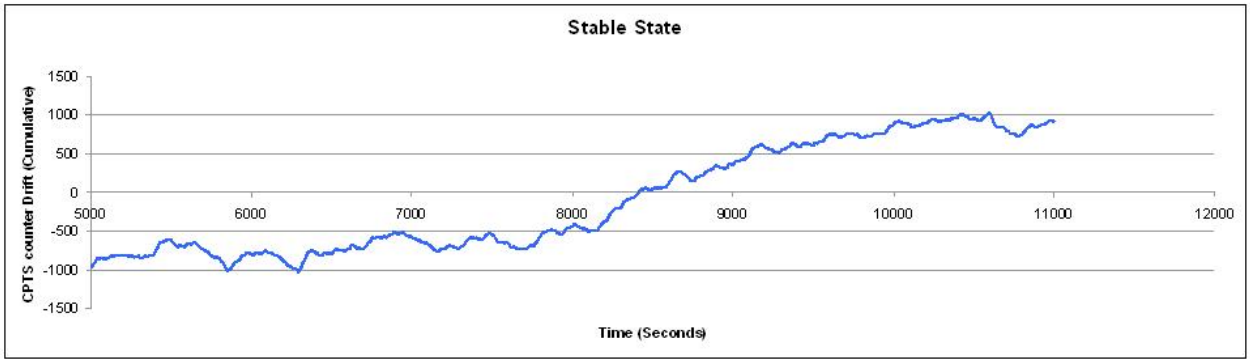






PTP Log Cumulative Counter Drift

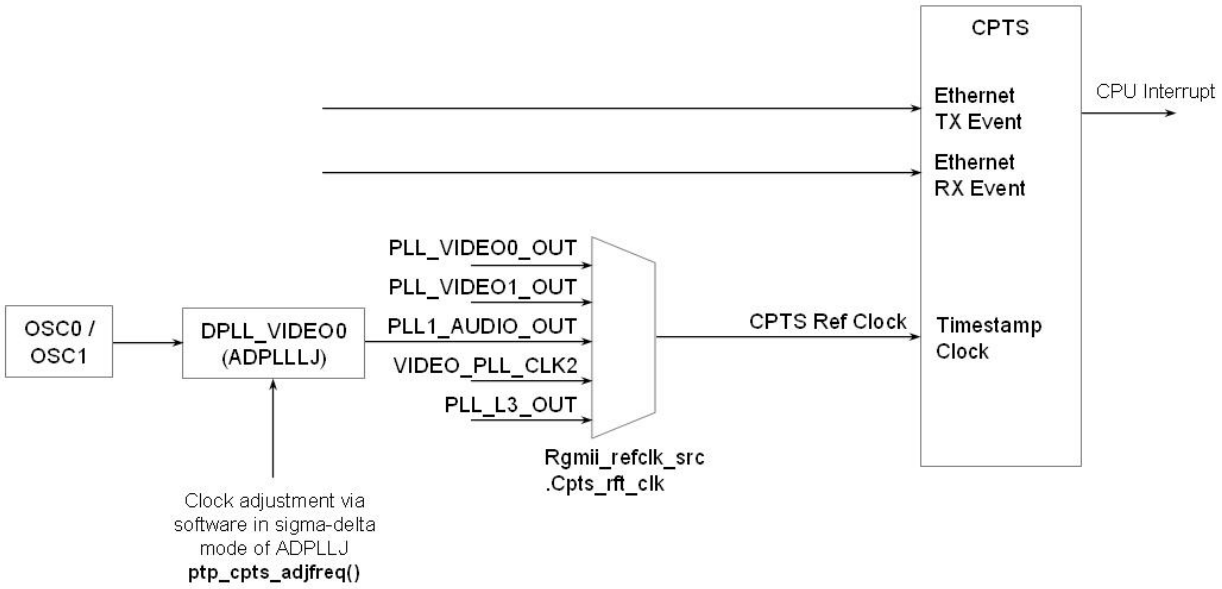




Procedure for Implementing Custom Clock using CPTS

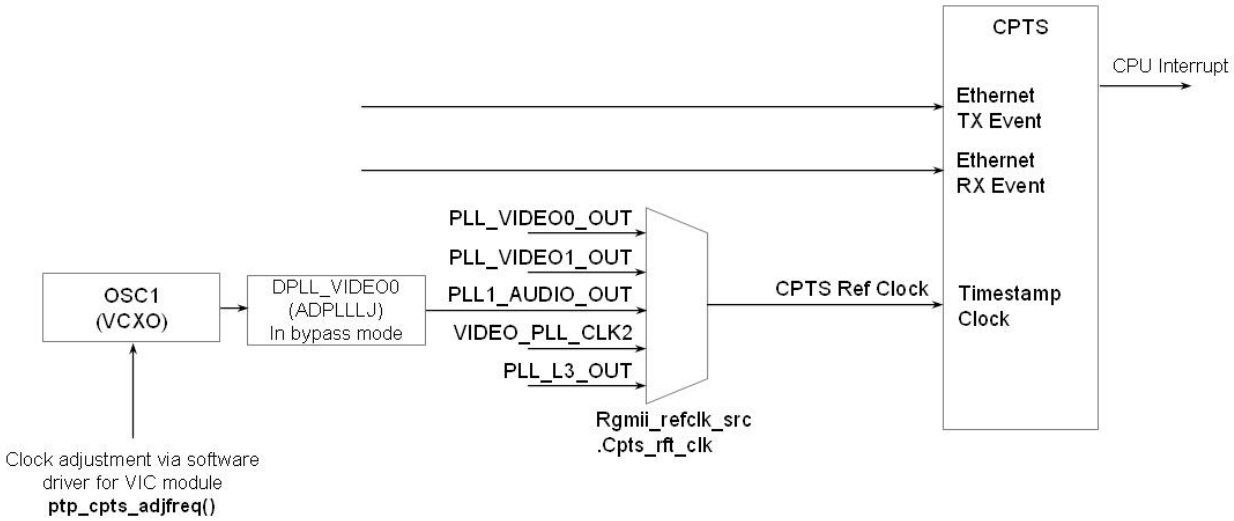
TI Implementation

1. Select PLL_AUDIO_OUT as CPTS ref clock in RMII_REFCLK_SRC in UBoot.
2. In ptp_cpts_init() get the PLL_AUDIO_OUT PLL clock data structure from clock framework.
3. In ptp_cpts_adjfreq()
   1. Convert the ppb (No of parts per billion to be drifted from current clock frequency) to frequency to set to the clock.
      f' = f + f * ppb/1000000000
   2. Set f' to the PLL_AUDIO_OUT by cpts_ref_clk->set_rate()



Customer Implementation

1. Select PLL_AUDIO_OUT as CPTS ref clock in Rgmii_refclk_src.Cpts_rft_clk in UBoot.
2. Make the PLL_AUDIO_OUT to be in Bypass Mode, so that the Clock from the OSC1 is fed directly to the CPTS ref clock input.
3. In ptp_cpts_init() initialize the VCXO clock data structure to cpts_clock data structure.
4. In ptp_cpts_adjfreq()
   1. Convert the ppb (No of parts per billion to be drifted from current clock frequency) to the VCXO value where it provides 1 ppb precision.
   2. Set the VCXO value, so that the desired frequency from the below formula is generated by OSC1.
      f' = f + f * ppb/1000000000



Switch Mode of Operation

Introduction

The DM814X Ethernet Switch can be configured in various different combination of Ethernet Packet forwarding and blocking. There is no such standard interface in Linux to configure a switch. This user guide provides an interface to configure the switch using Socket IOCTL through SIOCDEVPRIVATE command.

Configuring Kernel with VLAN Support

```
    Userspace binary formats  --->
    Power management options  --->
[*] Networking support  --->
    Device Drivers  --->
    File systems  --->
    Kernel hacking  --->
```

```
--- Networking support
      Networking options  --->
[ ]   Amateur Radio support  --->
<*>   CAN bus subsystem support  --->
< >   IrDA (infrared) subsystem support  --->
< >   Bluetooth subsystem support  --->
< >   RxRPC session sockets
```

```
< > The RDS Protocol (EXPERIMENTAL)
< > The TIPC Protocol (EXPERIMENTAL)  --->
< > Asynchronous Transfer Mode (ATM)
< > Layer Two Tunneling Protocol (L2TP)  --->
< > 802.1d Ethernet Bridging
[ ] Distributed Switch Architecture support  --->
<*> 802.1Q VLAN Support
[*]   GVRP (GARP VLAN Registration Protocol) support
< > DECnet Support
```

```
< > ANSI/IEEE 802.2 LLC type 2 Support
< > The IPX protocol
```

Switch Config Commands

Following is sample code for configuring the switch.

```
include <stdio.h> ... include <linux/net_switch_config.h> int main(void) { struct net_switch_config cmd_struct; struct ifreq ifr; int sockfd; strncpy(ifr.ifr_name, "eth0", IFNAMSIZ); ifr.ifr_data = (char*)&cmd_struct; if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) { printf("Can't open the socket\n"); return -1; } memset(&cmd_struct, 0, sizeof(struct net_switch_config));
```

...//initialise cmd_struct with switch commands

if (ioctl(sockfd, SIOCDEVPRIVATE, &ifr) < 0) { printf("Command failed\n"); close(sockfd); return -1; } printf("command success\n"); close(sockfd); return 0; }

CONFIG_SWITCH_ADD_MULTICAST

CONFIG_SWITCH_ADD_MULTICAST is used to add a LLDP Multicast address and forward the multicast packet to the subscribed ports. If VLAN ID is greater than zero then VLAN LLDP/Multicast is added.

cmd_struct.cmd = CONFIG_SWITCH_ADD_MULTICAST

Parameter	Description	Range
cmd_struct.cmd_data.switchcmd.addr	LLDP/Multicast Address	MAC Address
cmd_struct.cmd_data.switchcmd.mem_port	Member port Bit 0 – Host port/Port 0 Bit 1 – Slave 0/Port 1 Bit 2 – Slave 1/Port 2	0 – 7
cmd_struct.cmd_data.switchcmd.vid	VLAN ID	0 – 4095
cmd_struct.cmd_data.switchcmd.flag	Super	0/1
cmd_struct.cmd_data.switchcmd.untag_port	Multicast forward state	0 – 3

Result

ioctl call returns success or failure.

CONFIG_SWITCH_ADD_UNICAST

CONFIG_SWITCH_ADD_UNICAST is used to add a Unicast address and forward the unicast packet to that port. If VLAN ID is greater than zero then VLAN Unicast is added.

cmd_struct.cmd = CONFIG_SWITCH_ADD_UNICAST

Parameter	Description	Range
cmd_struct.cmd_data.switchcmd.addr	Unicast Address	MAC Address
cmd_struct.cmd_data.switchcmd.mem_port	Port Number	0 – 2
cmd_struct.cmd_data.switchcmd.vid	VLAN ID	0 – 4095
cmd_struct.cmd_data.switchcmd.blocked	Blocked	0/1
cmd_struct.cmd_data.switchcmd.secure	Secure Bit	0/1
cmd_struct.cmd_data.switchcmd.ageable	Ageable	0/1

Result

ioctl call returns success or failure.

CONFIG_SWITCH_ADD_OUI

CONFIG_SWITCH_ADD_OUI is used to add a OUI address.

cmd_struct.cmd = CONFIG_SWITCH_ADD_OUI

Parameter	Description	Range
cmd_struct.cmd_data.switchcmd.addr	Unicast Address	MAC Address
cmd_struct.cmd_data.switchcmd.mem_port	Member port Bit 0 – Host port/Port 0 Bit 1 – Slave 0/Port 1 Bit 2 – Slave 1/Port 2	0 – 7

Result

ioctl call returns success or failure.

CONFIG_SWITCH_FIND_ADDR

CONFIG_SWITCH_FIND_ADDR is used to find a address with or without VLAN ID.

cmd_struct.cmd = CONFIG_SWITCH_FIND_ADDR

Parameter	Description	Range
cmd_struct.cmd_data.switchcmd.addr	Unicast Address	MAC Address
cmd_struct.cmd_data.switchcmd.vid	VLAN ID	0 – 4095

Result

ioctl call returns success or failure.

On success cmd_struct.ret_type will hold the ALE table index

CONFIG_SWITCH_DEL_MULTICAST

CONFIG_SWITCH_DEL_MULTICAST is used to Delete a LLDP/Multicast address with or without VLAN ID.

cmd_struct.cmd = CONFIG_SWITCH_DEL_MULTICAST

Parameter	Description	Range
cmd_struct.cmd_data.switchcmd.addr	Unicast Address	MAC Address
cmd_struct.cmd_data.switchcmd.vid	VLAN ID	0 – 4095
cmd_struct.cmd_data.switchcmd.mem_port	Member port Bit 0 – Host port/Port 0 Bit 1 – Slave 0/Port 1 Bit 2 – Slave 1/Port 2	0 – 7

Result

ioctl call returns success or failure.

CONFIG_SWITCH_DEL_UNICAST

CONFIG_SWITCH_DEL_UNICAST is used to Delete a Unicast address with or without VLAN ID.

cmd_struct.cmd = CONFIG_SWITCH_DEL_UNICAST

Parameter	Description	Range
cmd_struct.cmd_data.switchcmd.addr	Unicast Address	MAC Address
cmd_struct.cmd_data.switchcmd.vid	VLAN ID	0 – 4095

Result

ioctl call returns success or failure.

CONFIG_SWITCH_ADD_VLAN

CONFIG_SWITCH_ADD_VLAN is used to add VLAN ID.

cmd_struct.cmd = CONFIG_SWITCH_ADD_VLAN

Parameter	Description	Range
cmd_struct.cmd_data.switchcmd.vid	VLAN ID	0 – 4095
cmd_struct.cmd_data.switchcmd.mem_port	Member port Bit 0 – Host port/Port 0 Bit 1 – Slave 0/Port 1 Bit 2 – Slave 1/Port 2	0 – 7
cmd_struct.cmd_data.switchcmd.untag_port	Untagged Egress port mask Bit 0 – Host port/Port 0 Bit 1 – Slave 0/Port 1 Bit 2 – Slave 1/Port 2	0 – 7
cmd_struct.cmd_data.switchcmd.reg_multi	Registered Multicast flood port mask Bit 0 – Host port/Port 0 Bit 1 – Slave 0/Port 1 Bit 2 – Slave 1/Port 2	0 – 7

cmd_struct.cmd_data.switchcmd.unreg_multi	Unknown Multicast flood port mask Bit 0 – Host port/Port 0 Bit 1 – Slave 0/Port 1 Bit 2 – Slave 1/Port 2	0 – 7

Result

ioctl call returns success or failure.

CONFIG_SWITCH_FIND_VLAN

CONFIG_SWITCH_ADD_VLAN is used to add VLAN ID.

cmd_struct.cmd = CONFIG_SWITCH_ADD_VLAN

Parameter	Description	Range
cmd_struct.cmd_data.switchcmd.vid	VLAN ID	0 – 4095

Result

ioctl call returns success or failure.

On success cmd_struct.ret_type will hold the ALE table index

CONFIG_SWITCH_DEL_VLAN

CONFIG_SWITCH_DEL_VLAN is used to delete VLAN ID.

cmd_struct.cmd = CONFIG_SWITCH_DEL_VLAN

Parameter	Description	Range
cmd_struct.cmd_data.switchcmd.vid	VLAN ID	0 – 4095

Result

ioctl call returns success or failure.

CONFIG_SWITCH_SET_PORT_VLAN_CONFIG

CONFIG_SWITCH_SET_PORT_VLAN_CONFIG is used to set port VLAN ID.

cmd_struct.cmd = CONFIG_SWITCH_SET_PORT_VLAN_CONFIG

Parameter	Description	Range
cmd_struct.cmd_data.switchcmd.port	Port number	0 - 2
cmd_struct.cmd_data.switchcmd.vid	VLAN ID	0 – 4095
cmd_struct.cmd_data.switchcmd.prio_port	VLAN Priority	0 – 7
cmd_struct.cmd_data.switchcmd.CFI_port	VLAN CFI	0/1

Result

ioctl call returns success or failure.

CONFIG_SWITCH_TIMEOUT

CONFIG_SWITCH_TIMEOUT is used to set ALE aging timeout.

cmd_struct.cmd = CONFIG_SWITCH_TIMEOUT

Parameter	Description	Range
cmd_struct.cmd_data.switchcmd.ale_timeout	ALE age out time	Timeout in Milli second

Result

ioctl call returns success or failure.

CONFIG_SWITCH_DUMP

CONFIG_SWITCH_DUMP is used to dump ALE table.

cmd_struct.cmd = CONFIG_SWITCH_DUMP

Parameter	Description	Range
cmd_struct.cmd_data.switchcmd.aledump	ALE index	0 - 1023

Result

ioctl call returns success or failure.

On success "cmd_struct.cmd_data.buf" holds ALE dump text.

CONFIG_SWITCH_SET_FLOW_CONTROL

CONFIG_SWITCH_SET_FLOW_CONTROL is used to set flow control of the ports.

cmd_struct.cmd = CONFIG_SWITCH_SET_FLOW_CONTROL

Parameter	Description	Range
cmd_struct.cmd_data.portcmd.port	Port Mask Bit 0 – Host port/Port 0 Bit 1 – Slave 0/Port 1 Bit 2 – Slave 1/Port 2	0 - 7

Result

ioctl call returns success or failure.

CONFIG_SWITCH_SET_PRIORITY_MAPPING

CONFIG_SWITCH_SET_PRIORITY_MAPPING is used to set priority mapping of the ports.

cmd_struct.cmd = CONFIG_SWITCH_SET_PRIORITY_MAPPING

Parameter	Description	Range
cmd_struct.cmd_data.portcmd.port	Port Numnber	0 - 2
cmd_struct.cmd_data.priocmd.prio_rx	Receive priority	0 - 7
cmd_struct.cmd_data.priocmd.prio_tx	Transmit priority	0 - 7
cmd_struct.cmd_data.priocmd.prio_switch	Switch priority	0 - 3

Result

ioctl call returns success or failure.

CONFIG_SWITCH_PORT_STATISTICS_ENABLE

CONFIG_SWITCH_PORT_STATISTICS_ENABLE is used to enable hardware statics of the ports.

cmd_struct.cmd = CONFIG_SWITCH_PORT_STATISTICS_ENABLE

Parameter	Description	Range
switch_config.cmd_data.switchcmd.mem_port	Port Mask Bit 0 – Host port/Port 0 Bit 1 – Slave 0/Port 1 Bit 2 – Slave 1/Port 2	0 - 7

Result

ioctl call returns success or failure.

CONFIG_SWITCH_CONFIG_DUMP

CONFIG_SWITCH_CONFIG_DUMP is used to dump the switch configuration.

cmd_struct.cmd = CONFIG_SWITCH_CONFIG_DUMP

Parameter	Description	Range
None	-	-

Result

ioctl call returns success or failure.

On success "cmd_struct.cmd_data.buf" holds Switch dump text.

CONFIG_SWITCH_RATELIMIT

CONFIG_SWITCH_RATELIMIT is used to enable/disable rate limit of the ports.

cmd_struct.cmd = CONFIG_SWITCH_RATELIMIT

Parameter	Description	Range
cmd_struct.cmd_data.portcmd.enable	Enable/Disable	Enable - 1 Disable - 0
cmd_struct.cmd_data.portcmd.direction	Transmit/Receive	Transmit - 0 Receive - 1
cmd_struct.cmd_data.portcmd.port	Port number	0 - 2
cmd_struct.cmd_data.portcmd.addr_type	Broadcast/Multicast	ADDR_TYPE_BROADCAST / ADDR_TYPE_MULTICAST
cmd_struct.cmd_data.portcmd.limit	No of Packet	0 - 255

Result

ioctl call returns success or failure.

CONFIG_SWITCH_VID_INGRESS_CHECK

CONFIG_SWITCH_VID_INGRESS_CHECK is used to set VLAN Ingress Check.

cmd_struct.cmd = CONFIG_SWITCH_VID_INGRESS_CHECK

Parameter	Description	Range
cmd_struct.cmd_data.portcmd.port	Port number	0 - 2
cmd_struct.cmd_data.portcmd.vlan_ingress_check	Ingress enable/disable	Enable - 1 Disable - 0
cmd_struct.cmd_data.portcmd.drop_untagged	Drop untagged enable/disabe	Enable - 1 Disable - 0

Result

ioctl call returns success or failure.

CONFIG_SWITCH_ADD_UNKNOWN_VLAN_INFO

CONFIG_SWITCH_ADD_UNKNOWN_VLAN_INFO is used to set unknown VLAN Info.

cmd_struct.cmd = CONFIG_SWITCH_ADD_UNKNOWN_VLAN_INFO

Parameter	Description	Range
cmd_struct.cmd_data.portcmd.port	Port mask Bit 0 – Host port/Port 0 Bit 1 – Slave 0/Port 1 Bit 2 – Slave 1/Port 2	0 - 7
cmd_struct.cmd_data.portcmd.reg_multi_port_mask	Registered Multicast flood port mask Bit 0 – Host port/Port 0 Bit 1 – Slave 0/Port 1 Bit 2 – Slave 1/Port 2	0 - 7
cmd_struct.cmd_data.portcmd.unknown_reg_multi_port_mask	Unknown Multicast flood port mask Bit 0 – Host port/Port 0 Bit 1 – Slave 0/Port 1 Bit 2 – Slave 1/Port 2	0 - 7

cmd_struct.cmd_data.portcmd.unknown_vlan_member	Unknown Vlan Member port mask Bit 0 – Host port/Port 0 Bit 1 – Slave 0/Port 1 Bit 2 – Slave 1/Port 2	0 - 7

Result

ioctl call returns success or failure.

CONFIG_SWITCH_802_1

CONFIG_SWITCH_802_1 is used to enable 802.1 packet forwarding.

cmd_struct.cmd = CONFIG_SWITCH_802_1

Parameter	Description	Range
cmd_struct.cmd_data.portcmd.enable	Drop untagged enable/disabe	Enable - 1 Disable - 0

Result

ioctl call returns success or failure.

CONFIG_SWITCH_MACAUTH

CONFIG_SWITCH_MACAUTH is used to enable 802.1 packet forwarding.

cmd_struct.cmd = CONFIG_SWITCH_MACAUTH

Parameter	Description	Range
cmd_struct.cmd_data.portcmd.enable	Drop untagged enable/disabe	Enable - 1 Disable - 0

Result

ioctl call returns success or failure.

CONFIG_SWITCH_SET_PORT_CONFIG

CONFIG_SWITCH_SET_PORT_CONFIG is used to set Phy Config.

cmd_struct.cmd = CONFIG_SWITCH_SET_PORT_CONFIG

Parameter	Description	Range
cmd_struct.cmd_data.portcmd.port	Port number	0 - 2
cmd_struct.cmd_data.portcmd.limit	Speed	0 - Auto/ 10/100/1000
cmd_struct.cmd_data.portcmd.direction	Duplexity	Full - 1 Half - 0

Result

ioctl call returns success or failure.

CONFIG_SWITCH_GET_PORT_CONFIG

CONFIG_SWITCH_GET_PORT_CONFIG is used to get Phy Config.

cmd_struct.cmd = CONFIG_SWITCH_GET_PORT_CONFIG

Parameter	Description	Range
cmd_struct.cmd_data.portcmd.port	Port number	0 - 2

Result

ioctl call returns success or failure.

On success "cmd_struct.cmd_data.portcmd.limit" holds port speed (0 - auto/10/100/1000) and "cmd_struct.cmd_data.portcmd.direction" holds duplexity (1 - Full Duplex / 0 - Half Duplex)

CONFIG_SWITCH_PORT_STATE

CONFIG_SWITCH_PORT_STATE is used to set port status.

cmd_struct.cmd = CONFIG_SWITCH_PORT_STATE

Parameter	Description	Range
cmd_struct.cmd_data.portcmd.port	Port number	0 - 2
cmd_struct.cmd_data.portcmd.port_state	Port state	PORT_STATE_DISABLED/ PORT_STATE_BLOCKED/ PORT_STATE_LEARN/ PORT_STATE_FORWARD

Result

ioctl call returns success or failure.

CONFIG_SWITCH_RESET

CONFIG_SWITCH_RESET is used to reset the switch.

cmd_struct.cmd = CONFIG_SWITCH_RESET

Parameter	Description	Range
None	-	-

Result

ioctl call returns success or failure.

Dual Standalone EMAC mode

Introduction

This section provides the user guide for Dual Emac mode implementation. Following are the assumptions made for Dual Emac mode implementation

Assumptions

- Interrupt source is common for both eth interfaces
- CPDMA and skb buffers are common for both eth interfaces
- If eth0 is up, then eth0 napi is used. eth1 napi is used when eth0 interface is down
- CPSW and ALE will be in VLAN aware mode irrespective of enabling of 802.1Q module in Linux network stack for adding port VLAN.
- Interrupt pacing is common for both interfaces
- Hardware statistics is common for all the ports
- Switch config will not be available in dual emac interface mode

Constrains

The following are the constrains for Dual Emac mode implementation

- VLAN id 2 and 3 are reserved for EMAC 0 and 1 respectively for port segregation
- While adding VLAN id to the eth interfaces, same VLAN id should not be added in both interfaces which will lead to VLAN forwarding and act as switch
- While adding Multicast MAC ids to the eth interfaces, same Multicast MAC should not be added in both interfaces which will lead to Multicast forwarding and act as switch
- Sysfs ALE table and control interfaces are available in eth0 interface only
- Manual ip for eth1 is not supported from Linux kernel arguments

Compiling kernel

```
      Userspace binary formats  --->
      Power management options  --->
  [*] Networking support  --->
      Device Drivers  --->
      File systems  --->
      Kernel hacking  --->
```

```
      Generic Driver Options  --->
  < > Connector - unified userspace <-> kernelspace linker  --->
  <*> Memory Technology Device (MTD) support  --->
  < > Parallel port support  --->
  [*] Block devices  --->
  [*] Misc devices  --->
  < > ATA/ATAPI/MFM/RLL support (DEPRECATED)  --->
      SCSI device support  --->
  <*> Serial ATA and Parallel ATA drivers  --->
  [ ] Multiple devices driver support (RAID and LVM)  --->
  [ ] Fusion MPT device support  --->
```

```
    IEEE 1394 (FireWire) support  --->
< > I2O device support  --->
[*] Network device support  --->
[ ] ISDN support  --->
< > Telephony support  --->
    Input device support  --->
```

```
--- Network device support
< >   Dummy net driver support
< >   Bonding driver support
< >   MAC-VLAN support (EXPERIMENTAL)
< >   EQL (serial line load balancing) support
< >   Universal TUN/TAP device driver support
< >   Virtual ethernet pair device
< >   ARCnet support  --->
-*-   Generic Media Independent Interface device support
-*-   PHY Device support and infrastructure  --->
[ ]   Ethernet (10 or 100Mbit)  --->
[*]   Ethernet (1000 Mbit)  --->
[ ]   Ethernet (10000 Mbit)  --->
< >   Token Ring driver support  --->
[ ]   Wireless LAN  --->
```

```
--- Ethernet (1000 Mbit)
< >   Alteon AceNIC/3Com 3C985/NetGear GA620 Gigabit support
< >   DL2000/TC902x-based Gigabit Ethernet support
-*-   TI DaVinci MDIO Support
-*-   TI DaVinci CPDMA Support
<*>   TI CPSW Switch Support
[*]     TI CPSW Switch as Dual EMAC
< >   Intel(R) PRO/1000 Gigabit Ethernet support
< >   Intel(R) PRO/1000 PCI-Express Gigabit Ethernet support
< >   IP1000 Gigabit Ethernet support
```

Bringing Up interfaces

Eth0 will be up by-default. Eth1 interface has to be brought up manually using either of the folloing command or through init scripts

DHCP

```
ifup eth1
```

Manual IP

```
ifconfig eth1 <ip> netmask <mask> up
```

Article Sources and Contributors

TI81XX PSP ETHERNET Switch User Guide *Source*: http://processors.wiki.ti.com/index.php?oldid=119701 *Contributors*: Mugunthanvnm, Parth.saxena, RK, Sgturner

Image Sources, Licenses and Contributors

Image:TIBanner.png *Source*: http://processors.wiki.ti.com/index.php?title=File:TIBanner.png *License*: unknown *Contributors*: Nsnehaprabha

Image:Driver_Implementation.JPG *Source*: http://processors.wiki.ti.com/index.php?title=File:Driver_Implementation.JPG *License*: unknown *Contributors*: Mugunthanvnm

Image:Ptp-host-evm-setup.jpg *Source*: http://processors.wiki.ti.com/index.php?title=File:Ptp-host-evm-setup.jpg *License*: unknown *Contributors*: Mugunthanvnm

Image:Ptp-log-with-reset-1.JPG *Source*: http://processors.wiki.ti.com/index.php?title=File:Ptp-log-with-reset-1.JPG *License*: unknown *Contributors*: Mugunthanvnm

Image:Ptp-log-with-reset-2.JPG *Source*: http://processors.wiki.ti.com/index.php?title=File:Ptp-log-with-reset-2.JPG *License*: unknown *Contributors*: Mugunthanvnm

Image:Ptp-log-without-reset-1.JPG *Source*: http://processors.wiki.ti.com/index.php?title=File:Ptp-log-without-reset-1.JPG *License*: unknown *Contributors*: Mugunthanvnm

Image:Ptp-log-without-reset-2.JPG *Source*: http://processors.wiki.ti.com/index.php?title=File:Ptp-log-without-reset-2.JPG *License*: unknown *Contributors*: Mugunthanvnm

Image:Ptp-test-setup.jpg *Source*: http://processors.wiki.ti.com/index.php?title=File:Ptp-test-setup.jpg *License*: unknown *Contributors*: Mugunthanvnm

Image:PTP-log.JPG *Source*: http://processors.wiki.ti.com/index.php?title=File:PTP-log.JPG *License*: unknown *Contributors*: Mugunthanvnm

Image:PTP-log-cumulative.JPG *Source*: http://processors.wiki.ti.com/index.php?title=File:PTP-log-cumulative.JPG *License*: unknown *Contributors*: Mugunthanvnm

Image:Cpts-ti-implimentation.JPG *Source*: http://processors.wiki.ti.com/index.php?title=File:Cpts-ti-implimentation.JPG *License*: unknown *Contributors*: Mugunthanvnm

Image:Cpts-customer-implimentation.JPG *Source*: http://processors.wiki.ti.com/index.php?title=File:Cpts-customer-implimentation.JPG *License*: unknown *Contributors*: Mugunthanvnm

License

d. Except as otherwise agreed in writing by the Licensor or as may be otherwise permitted by applicable law, if You Reproduce, Distribute or Publicly Perform the Work either by itself or as part of any Adaptations or Collections, You must not distort, mutilate, modify or take other derogatory action in relation to the Work which would be prejudicial to the Original Author's honor or reputation. Licensor agrees that in those jurisdictions (e.g. Japan), in which any exercise of the right granted in Section 3(b) of this License (the right to make Adaptations) would be deemed to be a distortion, mutilation, modification or other derogatory action prejudicial to the Original Author's honor and reputation, the Licensor will waive or not assert, as appropriate, this Section, to the fullest extent permitted by the applicable national law, to enable You to reasonably exercise Your right under Section 3(b) of this License (right to make Adaptations) but not otherwise.

5. Representations, Warranties and Disclaimer

UNLESS OTHERWISE MUTUALLY AGREED TO BY THE PARTIES IN WRITING, LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF TITLE, MERCHANTIBILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS, ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

6. Limitation on Liability

EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. Termination

a. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Adaptations or Collections from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.
b. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

a. Each time You Distribute or Publicly Perform the Work or a Collection, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.
b. Each time You Distribute or Publicly Perform an Adaptation, Licensor offers to the recipient a license to the original Work on the same terms and conditions as the license granted to You under this License.
c. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.
d. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.
e. This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.
f. The rights granted under, and the subject matter referenced, in this License were drafted utilizing the terminology of the Berne Convention for the Protection of Literary and Artistic Works (as amended on September 28, 1979), the Rome Convention of 1961, the WIPO Copyright Treaty of 1996, the WIPO Performances and Phonograms Treaty of 1996 and the Universal Copyright Convention (as revised on July 24, 1971). These rights and subject matter take effect in the relevant jurisdiction in which the License terms are sought to be enforced according to the corresponding provisions of the implementation of those treaty provisions in the applicable national law. If the standard suite of rights granted under applicable copyright law includes additional rights not granted under this License, such additional rights are deemed to be included in the License; this License is not intended to restrict the license of any rights under applicable law.