AM335x CPSW (Ethernet) Driver's Guide

AM335x CPSW (Ethernet) Driver's Guide

Translate this page to cs - Česky Translate

Google™ Custom Search Search

Navigation

Main Page
All pages
All categories
Popular pages
Popular authors
Popular categories
Category stats
Recent changes
Random page
Help
Google Search

Print/export

Create a book
Download as PDF
Printable version

Toolbox

What links here
Related changes
Special pages
Permanent link
Browse properties

AM335x CPSW (Ethernet) Driver's Guide

Linux PSP

Contents [hide]

1 Introduction
2 Driver Configuration
2.1 Module Build
3 Interrupt Pacing
4 Switch Configuration Interface
4.1 Introduction
4.2 Configuring Kernel with VLAN Support
4.3 Switch Config Commands
4.3.1 CONFIG_SWITCH_ADD_MULTICAST
4.3.2 CONFIG_SWITCH_ADD_UNICAST
4.3.3 CONFIG_SWITCH_ADD_OUI
4.3.4 CONFIG_SWITCH_FIND_ADDR
4.3.5 CONFIG_SWITCH_DEL_MULTICAST
4.3.6 CONFIG_SWITCH_DEL_UNICAST
4.3.7 CONFIG_SWITCH_ADD_VLAN
4.3.8 CONFIG_SWITCH_FIND_VLAN
4.3.9 CONFIG_SWITCH_DEL_VLAN
4.3.10 CONFIG_SWITCH_SET_PORT_VLAN_CONFIG
4.3.11 CONFIG_SWITCH_TIMEOUT
4.3.12 CONFIG_SWITCH_DUMP
4.3.13 CONFIG_SWITCH_SET_FLOW_CONTROL
4.3.14 CONFIG_SWITCH_SET_PRIORITY_MAPPING
4.3.15 CONFIG_SWITCH_PORT_STATISTICS_ENABLE
4.3.16 CONFIG_SWITCH_CONFIG_DUMP
4.3.17 CONFIG_SWITCH_RATELIMIT
4.3.18 CONFIG_SWITCH_VID_INGRESS_CHECK
4.3.19 CONFIG_SWITCH_ADD_UNKNOWN_VLAN_INFO
4.3.20 CONFIG_SWITCH_802_1
4.3.21 CONFIG_SWITCH_MACAUTH
4.3.22 CONFIG_SWITCH_SET_PORT_CONFIG
4.3.23 CONFIG_SWITCH_GET_PORT_CONFIG
4.3.24 CONFIG_SWITCH_PORT_STATE
4.3.25 CONFIG_SWITCH_RESET
5 Switch Configuration Example
5.1 Introduction
5.2 Source
5.3 Compilation
5.4 Switch configuration commands
5.5 Arguments
6 Dual Standalone EMAC mode
6.1 Introduction
6.2 Assumptions
6.3 Constrains
6.4 Compiling kernel
6.5 Bringing Up interfaces
6.5.1 DHCP
6.5.2 Manual IP

Introduction

TI Common Platform Ethernet Switch (CPSW) is a three port switch (one CPU port and two external ports).

Driver Configuration

To enable/disable Networking support, start the *Linux Kernel Configuration* tool:

```
$ make menuconfig
```

Select *Networking support* from the main menu.

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

Then select *Device Drivers* from the main menu.

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

Select *Network device support* as shown below:

```
    ...
    ...
    [*]Network device support --->
    ISDN support --->
    ...
    ...
```

Select *Ethernet driver support* as shown below:

```
    ...
    ...
    *** CAIF transport drivers ***
    [*]   Ethernet driver support  --->
    -*-   PHY Device support and infrastructure  --->
    < >   PPP (point-to-point protocol) support
    ...
    ...
```

Select as shown here:

```
    ...
    [*]   Texas Instruments (TI) devices
    < >     TI DaVinci EMAC Support
    -*-     TI DaVinci MDIO Support
    -*-     TI DaVinci CPDMA Support
    <*>     TI CPSW Switch Support
    <*>       TI TLK110 v1.0 PHY Workaround
```

Module Build

Module build for the cpsw driver is supported. To do this, at all the places mentioned in the section above select module build (short-cut key **M**).

Select as shown here:

```
    ...
    [*]   Texas Instruments (TI) devices
    < >     TI DaVinci EMAC Support
    -*-     TI DaVinci MDIO Support
    -*-     TI DaVinci CPDMA Support
    <M>     TI CPSW Switch Support
    <M>       TI TLK110 v1.0 PHY Workaround
```

Interrupt Pacing

CPSW interrupt pacing feature limits the number of interrupts that occur during a given period of time. For heavily loaded systems in which interrupts can occur at a very high rate, the performance benefit is significant due to minimizing the overhead associated with servicing each interrupt.

To enable interrupt pacing, please execute below mentioned command using ethtool utility:
ethtool -C eth0 rx-usecs <delayperiod>
To achieve maximum performance set <delayperiod> to 500

CPSW driver uses Timer 5 & 6 for CPSW Interrupt Pacing. Sharing this Timer with any other module will result in Ethernet pacing not working properly.

Switch Configuration Interface

Introduction

The CPSW 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 SIOCSWITCHCONFIG 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, SIOCSWITCHCONFIG, &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	0 - 7

cmd_struct.cmd_data.portcmd.reg_multi_port_mask	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.

Switch Configuration Example

Introduction

The CPSW Ethernet Switch can be configured in various different combination of Ethernet Packet forwarding and blocking. This section provides an user level application interface to configure the switch. Switch-config is user space utility to configure the CPSW 3 Port switch with various combination of ALE entry and Phy configuration

Source

The Switch Configuratio Example can be found in the following location

<PSP Release>/src/examples/switch-config/

Compilation

Kernel Header installation

```
make ARCH=arm CROSS_COMPILE=arm-arago-linux-gnueabi- headers_install
```

Switch Configuration Example

```
make CROSS_COMPILE=arm-arago-linux-gnueabi- KBUILD_OUTPUT=<Kernel Src>
```

Switch configuration commands

All input and output values are in Decimal

switch-config -m,--add-multi <address> -n,--port <Portmask> [-V,--vid <vid>] [-s,--super] [-k,--fwd-state <value 0-3>]

switch-config -u,--add-uni <address> -n,--port <Port number 0-2> [-V,--vid <vid>] [-a,--ageable] [-b,--blocked] [-E,--secure]

switch-config -O,--add-oui <address>

switch-config -f,--find-addr <address> [-V,--vid <vid>]

switch-config -x,--del-multi <address> [-V,--vid <vid>] [-n,--port <Portmask>]

switch-config -X,--del-uni <address> [-V,--vid <vid>]

switch-config -i,--add-vlan <vlan id> -n,--port <Portmask> [-K,--vid-untag <Portmask>] [-M,--reg-multi <Portmask>] [-N,--unreg-multi <Portmask>]

switch-config -F,--find-vlan <vid>

switch-config -y,--del-vlan <vid> [-n,--port <Portmask>]

switch-config -p,--port-vlan <vid> -n,--port <Port No 0-2> [-r,--priority <priority 0-7>] [-C,--cfi]

switch-config -t,--timeout <timeout (msec)>

switch-config -h,--dump [-j,--index <index 0-1023>]

switch-config -c,--flow-control -n,--port <Portmask>

switch-config -P,--prio-map -n,--port <Portmask> [-q,--prio-rx <value 0-7>] [-Q,--prio-tx <value 0-7>] [-W,--prio-swi <value 0-3>]

switch-config -T,--en-stat -n,--port <Portmask>

switch-config -H,--conf-dump

switch-config -l,--rate-limit -n,--port <Portmask> [-e,--enable (Specify to enable) -B,--addr-type <broadcast/multicast> -L,--limit <No of Packet 0-255> [-d,--direction specify for Tx]]

switch-config -I,--vid-ingress-check -n,--port <Portmask> [-e,--enable (Specify to enable)] [-w,--drop-untagged]

switch-config -U,--add-unknown-vlan-info [-Y,--vid-untag-port-mask <0-7>] [-z,--reg-multi-port-mask <0-7>] [-Z,--unreg-multi-port-mask <0-7>] [-J,--unknown-vlan-mem <0-7>]

switch-config -8,--802-1 [-e,--enable (Specify to enable)]

switch-config -A,--mac-auth [-e,--enable (Specify to enable)]

switch-config -S,--set-port-config <0(auto)/10/100/1000> -n,--port <PortNo> [-D,--duplex <full/half>]

switch-config -G,--get-port-config -n,--port <PortNo>

switch-config -g,--port-state <disabled/blocked/learn/forward> -n,--port <PortNo>

switch-config -R,--reset

Arguments

-m,--add-multi Add multicast or VLAN multicast address entry to the ALE

-u,--add-uni Add Unicast or VLAN unicast addrey entry to ALE

-O,--add-oui Add OUI address entry to ALE

-f,--find-addr Find an address from ALE with or without VLAN

-x,--del-multi Delete a multicast entry with or without VLAN from ALE

-X,--del-uni Delete a unicast entry with or without VLAN from ALE

-i,--add-vlan Add a VLAN entry to ALE

-F,--find-vlan Find a VLAN entry from ALE

-y,--del-vlan Delete a VLAN entry from ALE also to change Port membership

-p,--port-vlan Specify port VLAN for a particular CPSW port

-t,--timeout Specify ALE timeout for ALE flush learned-touched entry

-h,--dump Dump ALE entry to console also to dump particular ALE entry

-c,--flow-control Specify flow control of each port either for tx or rx

-P,--prio-map Specify priority Mapping of the ports

-T,--en-stat Enable hardware statistics

-H,--conf-dump CPSW configuration dump

-l,--rate-limit Specify rate limit for each port for Multicast and Broadcast packets

-I,--vid-ingress-check Specify VLAN ingress check

-U,--add-unknown-vlan-info Add unknown VLAN information untag/drop/forward to port

-8,--802-1 Enable or disable 802.1 forwarding

-A,--mac-auth Enable or disable Mac Authentication

-S,--set-port-config Set Phy Configuration

-G,--get-port-config Display phy configuration

-g,--port-state Set port state

-R,--reset Reset the switch

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
- Both the interfaces should not be connected to the same subnet unless only configuring bridging, and not doing IP routing, then you can configure the two interfaces on the same subnet.

Compiling kernel

Userspace binary formats --->

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

```
[ ] Multiple devices driver support (RAID and LVM)  --->
< > Generic Target Core Mod (TCM) and ConfigFS Infrastructure  --->
[*] Network device support  --->
[ ] ISDN support  --->
< > Telephony support  --->
    Input device support  --->
    Character devices  --->
```

```
< >     Universal TUN/TAP device driver support
< >     Virtual ethernet pair device
      *** CAIF transport drivers ***
[*]   Ethernet driver support  --->
-*-   PHY Device support and infrastructure  --->
< >   PPP (point-to-point protocol) support
< >   SLIP (serial line) support
```

```
[*]   Texas Instruments (TI) devices
< >     TI DaVinci EMAC Support
-*-     TI DaVinci MDIO Support
-*-     TI DaVinci CPDMA Support
<*>     TI CPSW Switch Support
[*]       TI CPSW Switch as Dual EMAC
```

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
```

	Ultra Low Power DSP	8 bit Microcontroller MCU	16 bit Microcontroller MCU	32 bit Microcontroller MCU

Categories: AM335x | Linux | PSP

Leave a Comment