Application Note

Five-Ethernet-Port Enablement on AM64x and AM243x

Ashwani Goel, Teja Rowthu

Sitara MPU

ABSTRACT

This document provides guidelines for the design, development, implementation, and performance metrics of five Ethernet® ports support using R5F and A53 cores available on Texas Instruments' Arm® based system-on-chip (SoC) AM64x and AM243x devices. Currently, the implementation is only available with RTOS based multicore implementation as both Common Platform Ethernet Switch (CPSW3G) and Industrial Communication Subsystem (PRU-ICSSG) Ethernet ports work simultaneously.

Currently, this implementation is not supported in public MCU+ SDK release. Please follow FAQ for more updated details.

Table of Contents

List of Figures

Trademarks

1 Introduction

1.1 AM64x and AM243x EVMs

AM64x and AM243x are industrial grade devices from the Sitara™ MCU family.

The AM64x device is built for industrial applications, such as motor drives and remote I/O modules, which require a combination of real-time communications and processing.

The AM64x family provides scalable performance with up to two instances of gigabit TSN-enabled PRU_ICSSG in Sitara. For more information, see the *AM64x and AM243x Technical Reference Manual*.

1.2 SoC Architecture

This section describes the SOC-level description of SITARA MPU devices AM64x and AM243x.

1.2.1 AM64x

The AM64x devices are an extension of the Sitara MCU industrial-grade family of heterogeneous Arm processors. AM64x is built for industrial applications, such as motor drives and Programmable Logic Controllers (PLC), which require a unique combination of real-time processing and communications with applications processing. AM64x combines two instances of the gigabit TSN-enabled PRU-ICSSG of the Sitara device with up to two Arm® Cortex®-A53 cores, up to four Cortex-R5F MCUs, and a Cortex-M4F MCU. AM64x is built to provide real-time performance through the high-performance R5Fs, tightly-coupled memory banks, configurable SRAM partitioning, and dedicated low-latency paths to and from peripherals for rapid data movement in and out of the SoC. This deterministic architecture allows for AM64x to handle the tight control loops found in servo drives while the peripherals such as Fast Serial Interface (FSI), General Purpose Memory Controller (GPMC), Pulse-Width Modulation (PWM), sigma-delta decimation filters, and absolute encoder interfaces help enable a number of different architectures found in these systems.

The Cortex-A53s provide the powerful computing elements necessary for Linux applications. Linux, and Real-Time (RT) Linux, is provided through TI's Processor SDK Linux which stays updated to the latest Long Term Support (LTS) Linux kernel, bootloader, and Yocto Project™ file system on an annual basis. AM64x enables isolation between Linux applications and real-time streams through configurable memory partitioning. The Cortex-A53s can be assigned to work strictly out of DDR for Linux, and the internal SRAM can be broken up into various sizes for the Cortex-R5Fs to use together or independently. The AM64x provides flexible industrial communications capability including full protocol stacks for EtherCAT® subdevices, PROFINET® devices, Ethernet, image prompt (IP) adapter, and IO-Link controllers. The PRU-ICSSG further provides capability for gigabit and TSN-based protocols. The PRU-ICSSG also enables additional interfaces in the SoC including sigma-delta decimation filters and absolute encoder interfaces. Functional safety features can be enabled through the integrated Cortex-M4F along with dedicated peripherals that can be isolated from the rest of the SoC. AM64x also supports secure boot.

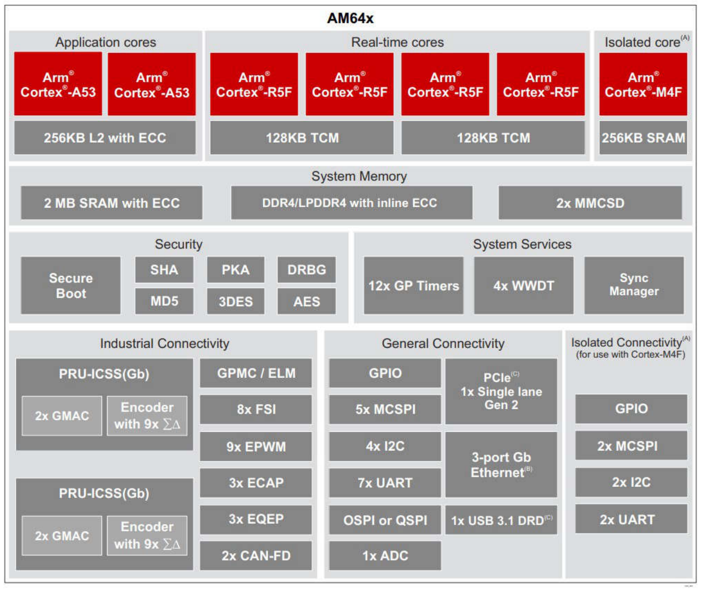


Figure 1-1. AM64x

1.2.2 AM243x

AM243x is an extension of the industrial-grade portfolio from the Sitara technology into high-performance microcontrollers. The AM243x device is built for industrial applications, such as motor drives and remote I/O modules which require a combination of real-time communications and processing. The AM243x family provides scalable performance with up to four Cortex-R5F MCUs, one Cortex-M4F, and two instances of the gigabit TSN-enabled PRU_ICSSG in the Sitara microcontroller. The AM243x SoC architecture was designed to provide best-in-class real-time performance through the high-performance Arm Cortex-R5F cores, Tightly-Coupled Memory (TCM) banks, configurable SRAM partitioning, and dedicated low-latency paths to and from peripherals for rapid data movement in and out of the SoC.

This deterministic architecture allows for AM243x to process the tight control loops found in servo drives, while the peripherals such as FSI, GPMC, Enhanced Capture Modules (eCAP), PWMs, and encoder interfaces help enable a number of different architectures found in these systems. The SoC provides flexible industrial communications capability including full protocol stacks for EtherCAT targets, PROFINET devices, EtherNet or IP adapters, and IO-Link controllers. The PRU_ICSSG further provides capability for gigabit and TSN-based protocols. The PRU_ICSSG enables additional interfaces including a UART interface, sigma-delta decimation filters, and absolute encoder interfaces. Functional safety features can be enabled through the integrated Cortex-M4F along with dedicated peripherals that can be isolated from the rest of the SoC. AM243x also supports secure boot.

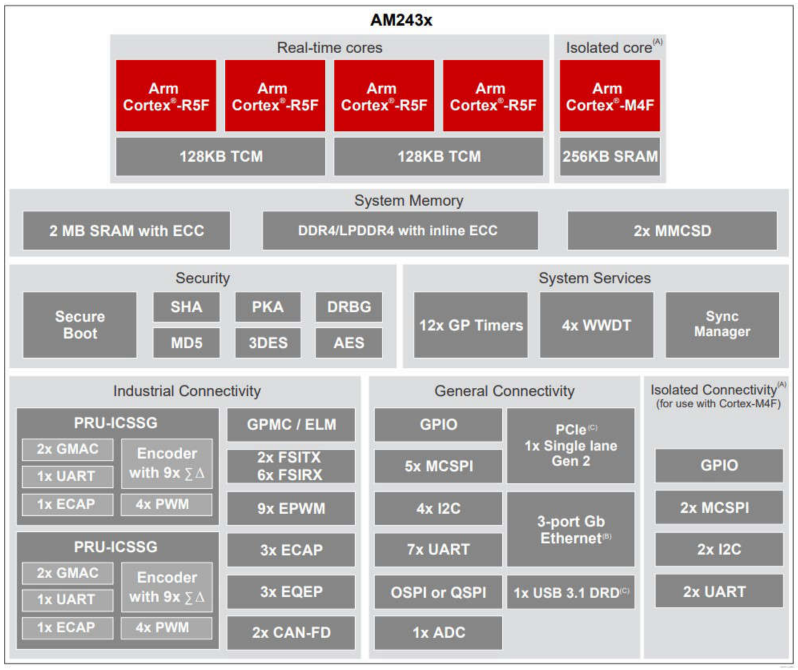


Figure 1-2. AM243x

1.3 Peripherals

A brief description about PRU-ICSSG and CPSW3G peripherals available on AM64x/AM243x is described in this section.

1.3.1 CPSW3G

CPSW (Common Platform Switch): CPSW subsystem provides IEEE 802.3 standard Ethernet gigabit speed packet communication for the device and can also be configured as an Ethernet switch.

CAUTION
CPSW is shown as *3-port Gb Ethernet* in Figure 1-1and Figure 1-2.

CAUTION
CPSW3G on AM64x and AM243x supports RGMII and RMII interfaces.

1.3.2 PRU-ICSSG

Programmable Real-Time Unit and Industrial Communication Subsystem Gigait (PRU-ICSSG): PRU-ICSSG is firmware programmable and can take on various personalities like Industrial Communication Protocol Switch (for protocols like EtherCAT, Profinet, Ethernet/IP), Ethernet Switch, Ethernet MAC, Industrial Drives, and so forth.

CAUTION
PRU-ICSSG on AM64x/AM243x supports RGMII and MII mode only.

CAUTION
PRU-ICSSG is shown as *PRU-ICSSG* in Figure 1-1 and Figure 1-2.

1.4 Ethernet Software Architecture

The software components overview highlighting the components used in the networking software development is shown in Figure 1-3.

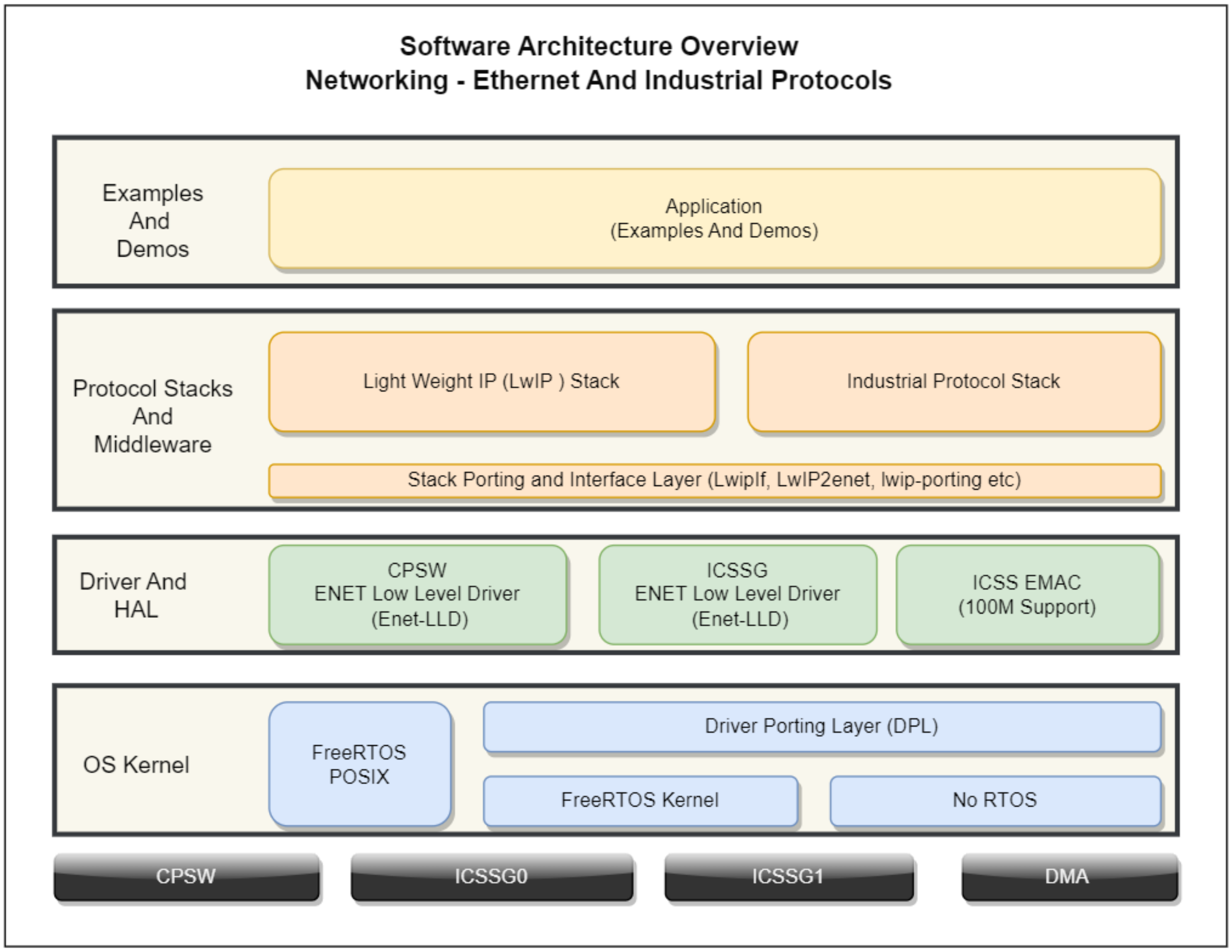

Software Architecture Overview
Networking - Ethernet And Industrial Protocols
Examples And Demos
Application (Examples And Demos)
Protocol Stacks And Middleware
Light Weight IP (LwIP) Stack
Industrial Protocol Stack
Stack Porting and Interface Layer (LwipIf, LwIP2enet, lwip-porting etc)
Driver And HAL
CPSW ENET Low Level Driver (Enet-LLD)
ICSSG ENET Low Level Driver (Enet-LLD)
ICSS EMAC (100M Support)
OS Kernel
FreeRTOS POSIX
Driver Porting Layer (DPL)
FreeRTOS Kernel
No RTOS
CPSW
ICSSG0
ICSSG1
DMA

Figure 1-3. Ethernet Networking Layers

1.5 Prerequisite

These are the prerequisites for enablement of five Ethernet ports using AM64x-EVM and a SEM board.

1.5.1 HW Prerequisite

- AM64x/AM243x-EVM

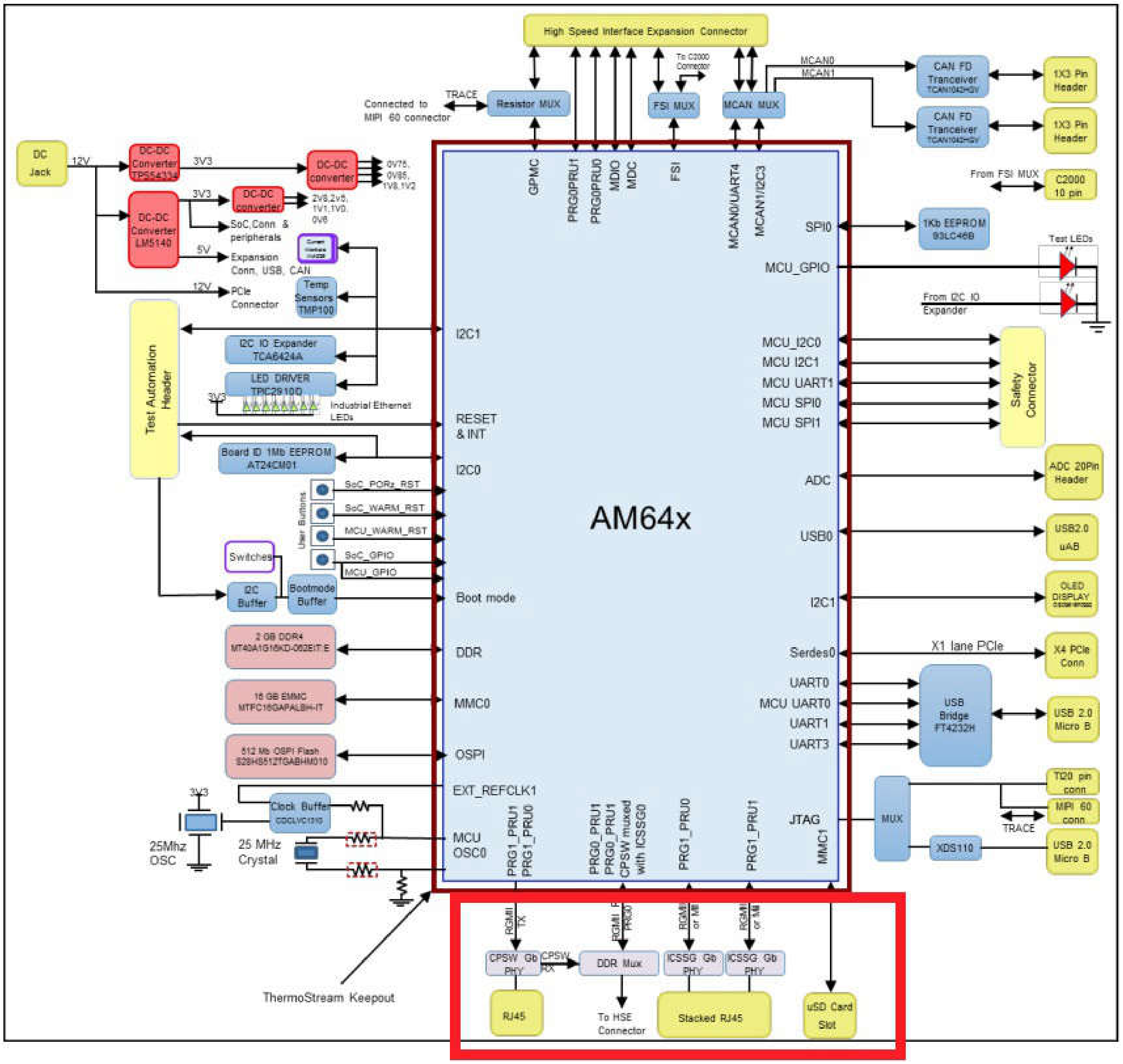


Figure 1-4. EVM Block Diagram

- SEM Board connected over HSE connector
  - DP83826e PHY
  - Only 100M is supported for testing purpose.

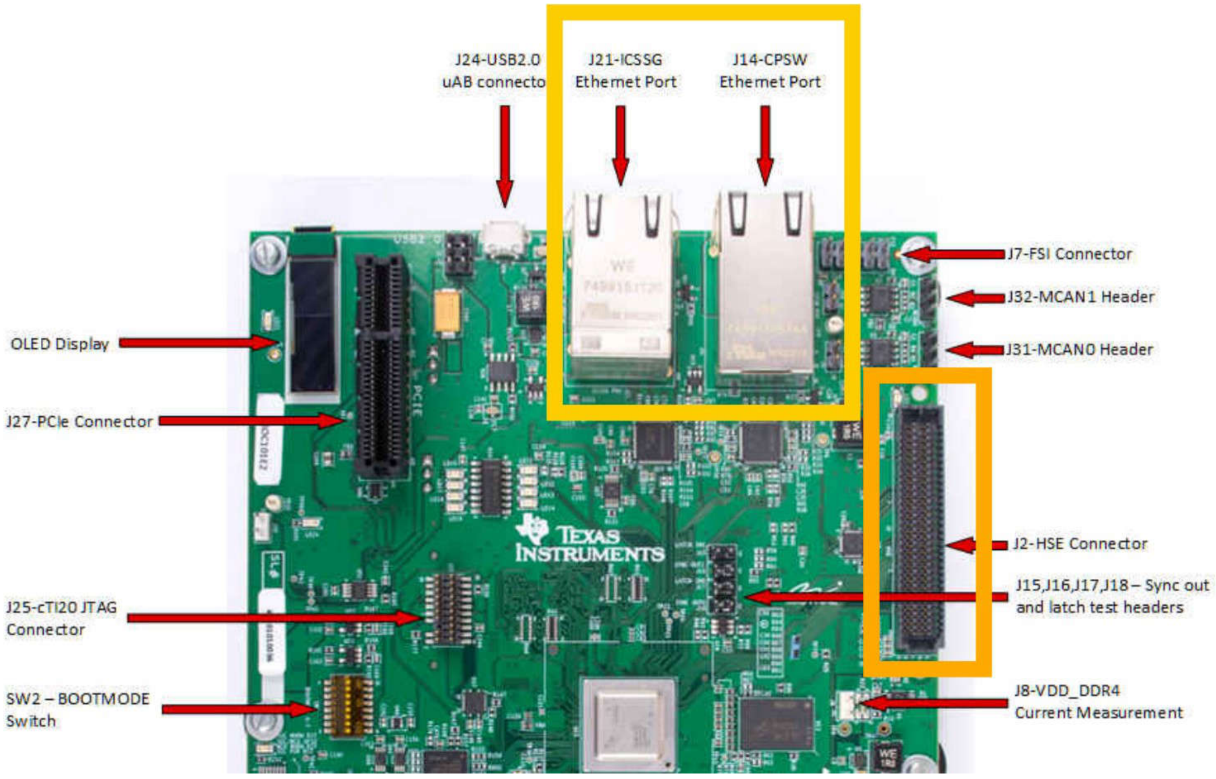


Figure 1-5. HSE Connector: ICSSG0 pin out

1.5.2 SW Prerequisite

- Install MCU-PLUS-SDK for AM64x or AM243x and dependent tools. For more information, see *Getting Started.*
- SYSCONFIG

CAUTION

Make sure to make SBL updates before running this example on the board. The default SBL does not allow CPSW to run on R5FSS_0-1.

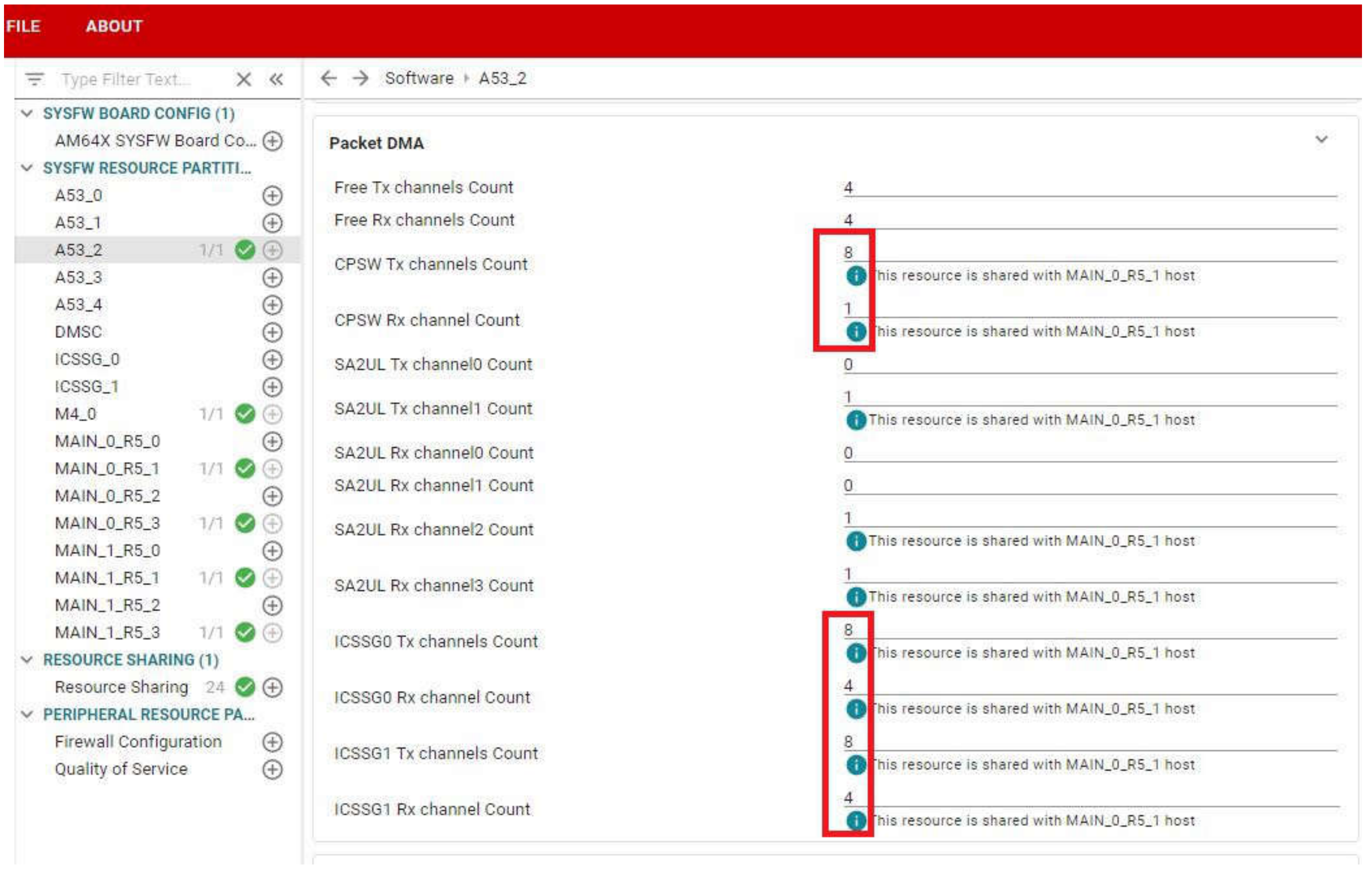
1.5.2.1 Resource Allocation - AM64x

Resource manager must be asked first to allocate any peripheral to any core. Users can configure ICSSG0, ICSSG1 and CPSW Ethernet ports with different modes (MII, RMII, RGMII) using sysconfig tool and attach to the core as per need.

> **CAUTION**
>
> These are the examples given for AM64x which has A53 core:
>
> AM243x does not have A53 core.
>
> Use resource allocation as per use case.

Use the following steps to enable this feature:

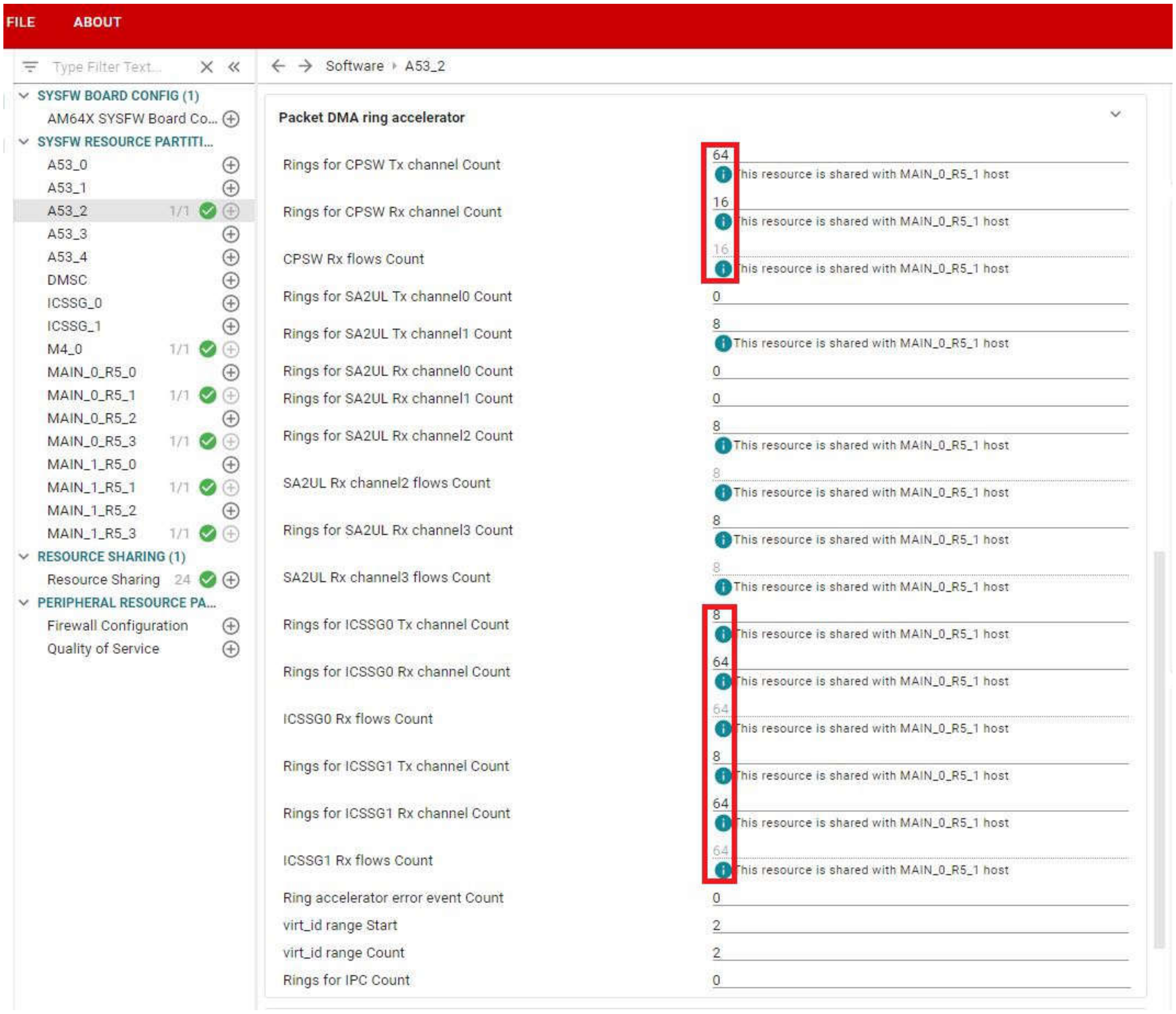
1. Navigate to the MCU-PLUS-SDK installation directory (In this document, this is referred to as MCU_SDK_HOME).
   For example: `C:\ti\mcu_plus_sdk_am64x_09_01_00_41`
2. Open the command prompt and run the following command:
   `gmake -s -C tools/sysfw/boardcfg configure SOC=am64x`
3. Change the Tx and Rx channel count for CPSW, ICSSG0, and ICSSG1 to *0* in the A53 core.



Free Tx channels Count	4
Free Rx channels Count	4
CPSW Tx channels Count	8 — This resource is shared with MAIN_0_R5_1 host
CPSW Rx channel Count	1 — This resource is shared with MAIN_0_R5_1 host
SA2UL Tx channel0 Count	0
SA2UL Tx channel1 Count	1 — This resource is shared with MAIN_0_R5_1 host
SA2UL Rx channel0 Count	0
SA2UL Rx channel1 Count	0
SA2UL Rx channel2 Count	1 — This resource is shared with MAIN_0_R5_1 host
SA2UL Rx channel3 Count	1 — This resource is shared with MAIN_0_R5_1 host
ICSSG0 Tx channels Count	8 — This resource is shared with MAIN_0_R5_1 host
ICSSG0 Rx channel Count	4 — This resource is shared with MAIN_0_R5_1 host
ICSSG1 Tx channels Count	8 — This resource is shared with MAIN_0_R5_1 host
ICSSG1 Rx channel Count	4 — This resource is shared with MAIN_0_R5_1 host

Figure 1-6. SysConfig: Packet DMA

4. After the channel counts are updated, the values are as follows:
    - CPSW Tx channel count = 0
    - CPSW Rx channel count = 0
    - ICSSG0 Tx channel count = 0
    - ICSSG0 Rx channel count = 0
    - ICSSG1 Tx channel count = 0
    - ICSSG1 Rx channel count = 0



Packet DMA ring accelerator

Rings for CPSW Tx channel Count	64	This resource is shared with MAIN_0_R5_1 host
Rings for CPSW Rx channel Count	16	This resource is shared with MAIN_0_R5_1 host
CPSW Rx flows Count	16	This resource is shared with MAIN_0_R5_1 host
Rings for SA2UL Tx channel0 Count	0	
Rings for SA2UL Tx channel1 Count	8	This resource is shared with MAIN_0_R5_1 host
Rings for SA2UL Rx channel0 Count	0	
Rings for SA2UL Rx channel1 Count	0	
Rings for SA2UL Rx channel2 Count	8	This resource is shared with MAIN_0_R5_1 host
SA2UL Rx channel2 flows Count	8	This resource is shared with MAIN_0_R5_1 host
Rings for SA2UL Rx channel3 Count	8	This resource is shared with MAIN_0_R5_1 host
SA2UL Rx channel3 flows Count	8	This resource is shared with MAIN_0_R5_1 host
Rings for ICSSG0 Tx channel Count	8	This resource is shared with MAIN_0_R5_1 host
Rings for ICSSG0 Rx channel Count	64	This resource is shared with MAIN_0_R5_1 host
ICSSG0 Rx flows Count	64	This resource is shared with MAIN_0_R5_1 host
Rings for ICSSG1 Tx channel Count	8	This resource is shared with MAIN_0_R5_1 host
Rings for ICSSG1 Rx channel Count	64	This resource is shared with MAIN_0_R5_1 host
ICSSG1 Rx flows Count	64	This resource is shared with MAIN_0_R5_1 host
Ring accelerator error event Count	0	
virt_id range Start	2	
virt_id range Count	2	
Rings for IPC Count	0	

Figure 1-7. SysConfig: Packet DMA Ring Accelerator

  - Rings for CPSW Tx channel count = 0
  - Rings for CPSW Rx channel count = 0
  - Rings for CPSW Rx flows count = 0
  - Rings for ICSSG0 Tx channel count = 0
  - Rings for ICSSG0 Rx channel count = 0
  - Rings for ICSSG0 Rx flows count = 0
  - Rings for ICSSG1 Tx channel count = 0
  - Rings for ICSSG1 Rx channel count = 0
  - Rings for ICSSG1 Rx flows count = 0
5. From (0,0) to (8,1) in R5-non secure 0-1 core (main-0-r5-0-1).
6. Change resource sharing from A52 -> r5_ss_0-1 to r5_ss_0-1 -> r5_ss_0-3.
7. `gmake -s -C tools/sysfw/boardcfg configure-gen SOC=am64x`
8. Navigate to the following file. `source/drivers/sciclient/sciclient_default_boardcfg/am64x/sciclient_defaultBoardcfg_rm.c`
9. Change the core allocations from A53 -> r5_ss_0-1 to r5_ss_0-1 -> r5_ss_0-3 for CPSW-related changes. There are three A53 changes and three r5-0-1 changes.

1.5.2.2 SBL update

CAUTION
This part is applicable only in the case of sbl boot is used.

Follow the steps below:

- For compiling SBL:

```
gmake -s -C tools/sysfw/boardcfg sciclient_boardcfg SOC=am64x
gmake -j -s -f makefile.am64x sbl-clean
gmake -j -s -f makefile.am64x sbl
```

- For flashing SBL:

```
cd tools/boot/
python3 uart_uniflash.py -p /dev/ttyUSB0 --cfg=sbl_prebuilt/am64x-evm/default_sbl_null.cfg
```

2 Multicore 5-Ethernet Ports Realization

Multicore 5-Ethernet ports can be realized by one of the two methods shown below:

- 2-Ethernet ports from ICSSG0, 2-Ethernet ports from ICSSG1, and 1-Ethernet port from CPSW.

or

- 2-Ethernet ports from ICSSG0, 1-Ethernet port from ICSSG1, and 2-Ethernet ports from CPSW.

These are the details on the method that can be realized using a SEM extension card with an HSE port, which adds two additional Ethernet ports for ICSSG0.

3 Supported Configurations on PRU-ICSSG

Here is a link to show the supported Ethernet configurations on ICSSG0/1: What are the possible Ethernet configurations on ICSSG0/1?.

4 Implementation

A SEM extension card on HSE port was used for adding two additional Ethernet ports for ICSSG0 and two Ethernet ports for the existing ICSSG1 Ethernet port. These four ICSSG Ethernet ports are controlled by R5FSS_0-0. This is enabled by ICSSG0 and ICSSG1 implementation. For more information, see the *Ethernet PRU_ICSSG Instance-0 (PRU_ICSSG0) Usage Guide.*

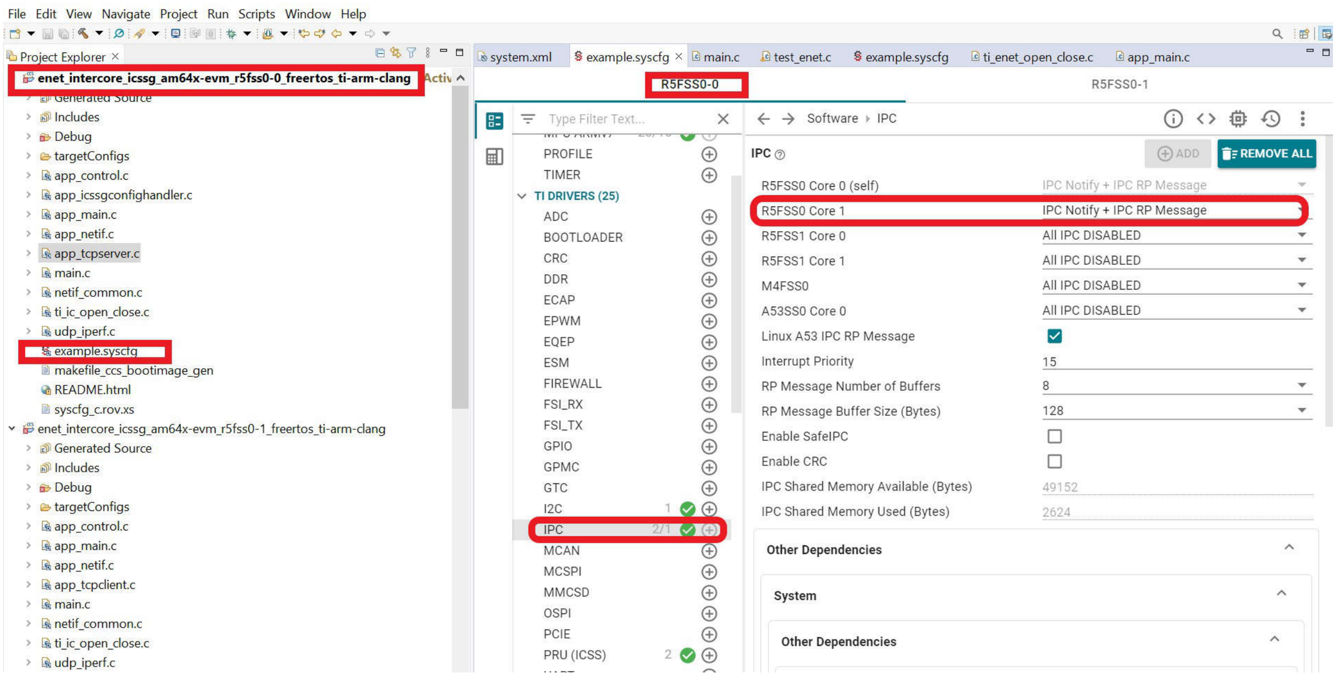
Users can release up to four Ethernet ports using ICSSG0 and ICSSG1. This can be either:

- 2-pairs of (2-port Ethernet switch mode),
- or four Ethernet ports in MAC mode,
- or any other combination.

This particular implementation uses all four Ethernet ports of ICSSG. A user can configure two Ethernet ports from ICSSG0 and one to ISSG1.

Driver that controls CPSW shall run on R5FSS_0-1 CPU core controls a single Ethernet port in MAC mode. This implementation can be changed based on requirements and can operate two on-board RGMII Ethernet ports.

Both the cores communicate over the IPC.



R5FSS0 Core 0 (self)	IPC Notify + IPC RP Message
R5FSS0 Core 1	IPC Notify + IPC RP Message
R5FSS1 Core 0	All IPC DISABLED
R5FSS1 Core 1	All IPC DISABLED
M4FSS0	All IPC DISABLED
A53SS0 Core 0	All IPC DISABLED
Linux A53 IPC RP Message	✓
Interrupt Priority	15
RP Message Number of Buffers	8
RP Message Buffer Size (Bytes)	128
Enable SafeIPC	☐
Enable CRC	☐
IPC Shared Memory Available (Bytes)	49152
IPC Shared Memory Used (Bytes)	2624

Figure 4-1. SysConfig: IPC settings

CAUTION

Note that the system project is required to run and trigger the IPC-related code inclusions.

IPC Examples must be build as System Projects. Single core build fails since IPC code generation depends on all core contexts in the application.

4.1 System Example

A system example is available at this link: TI Test. This example is applicable for AM243x and AM64x devices. The procedure was tested on the AM64x series. AM64x is configured as 4 × (MAC mode) + 1 × (MAC mode) Ethernet ports.

CAUTION
During testing, AM243x was configured as 2 × (2-port Ethernet switch) mode + 1 × (MAC mode).

Use the system make command to build the examples. Load the .out files to the respective cores or flash the combined application image in either UART or OSPI mode.

```
Example:
make -s -C test/networking/lwip/enet_icssg_tcpserver/am64x-evm/system_cpsw_icssg clean

make -s -C test/networking/lwip/enet_icssg_tcpserver/am64x-evm/system_cpsw_icssg

make -s -C test/networking/lwip/enet_icssg_tcpserver/am64x-evm/system_cpsw_icssg all
```

4.1.1 Software Architecture

An overall software architecture for 5-ethernet port allocations is shown in Figure 4-2.

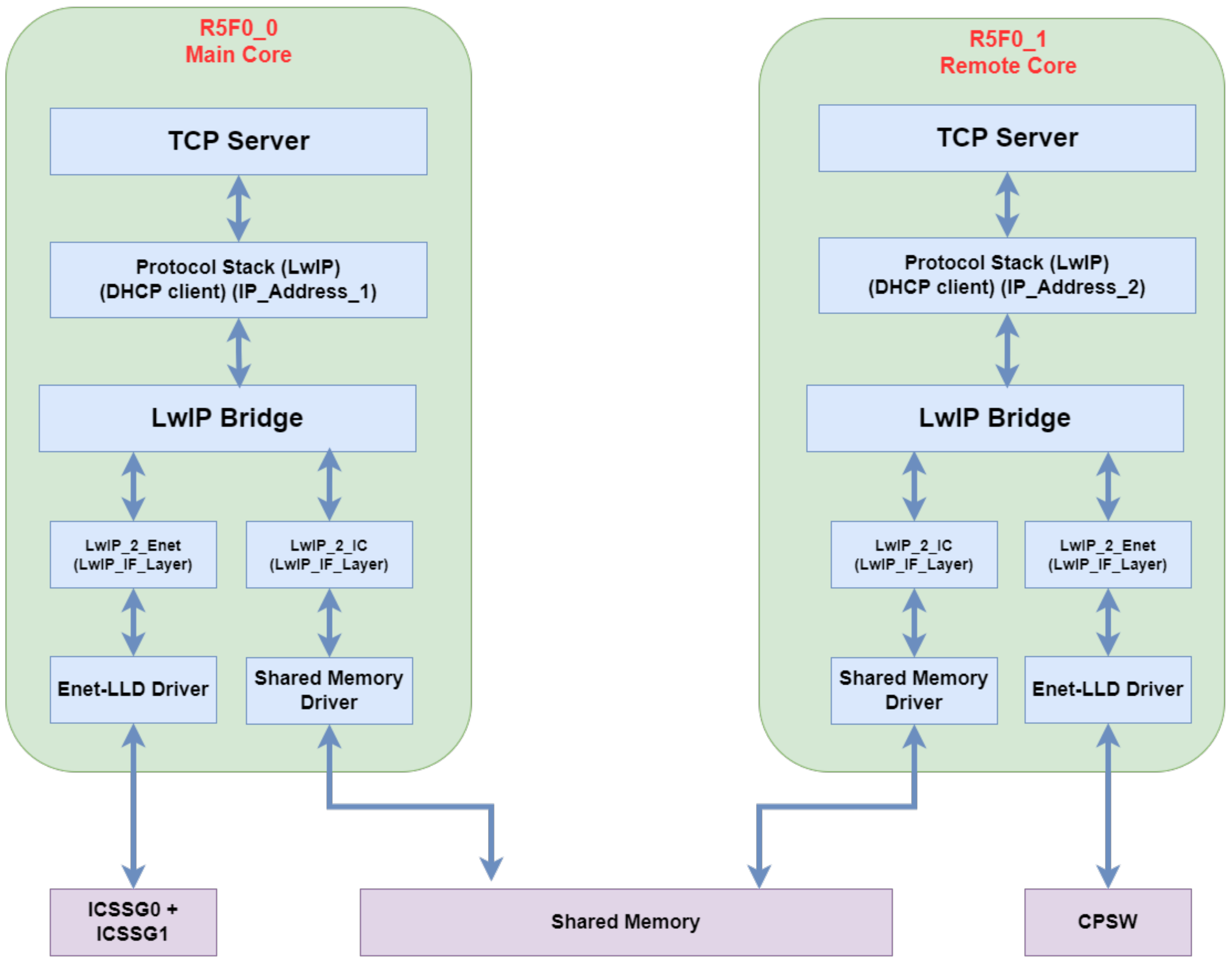


Figure 4-2. ICSSG and CPSW Allocation

4.1.2 5-Ethernet Port Example

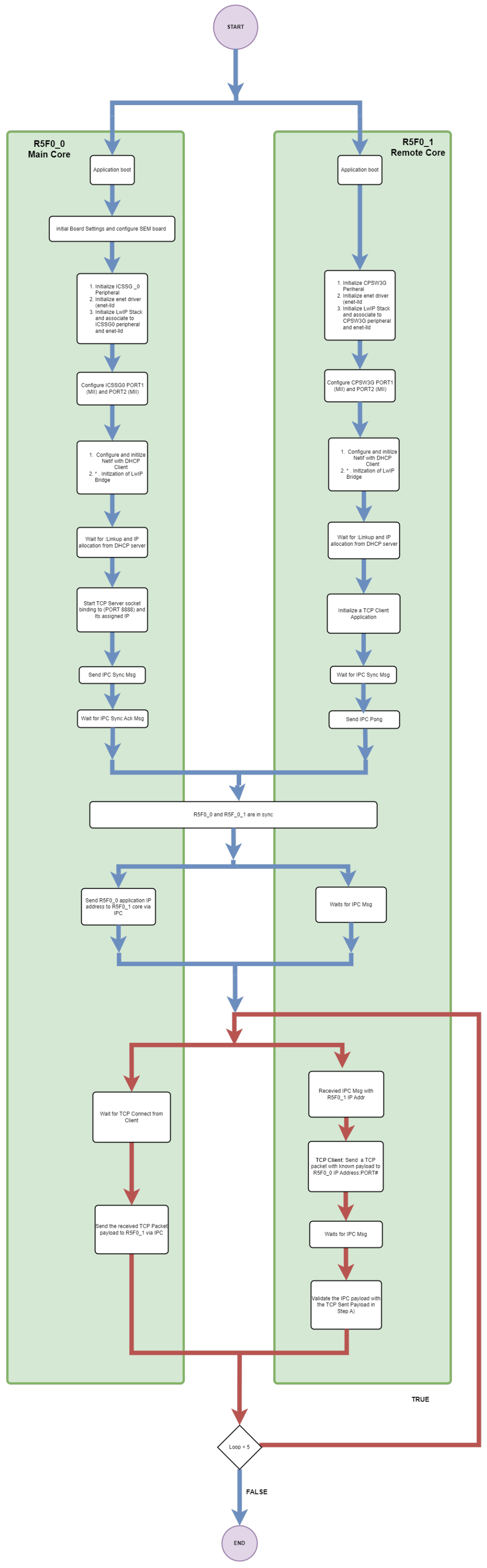
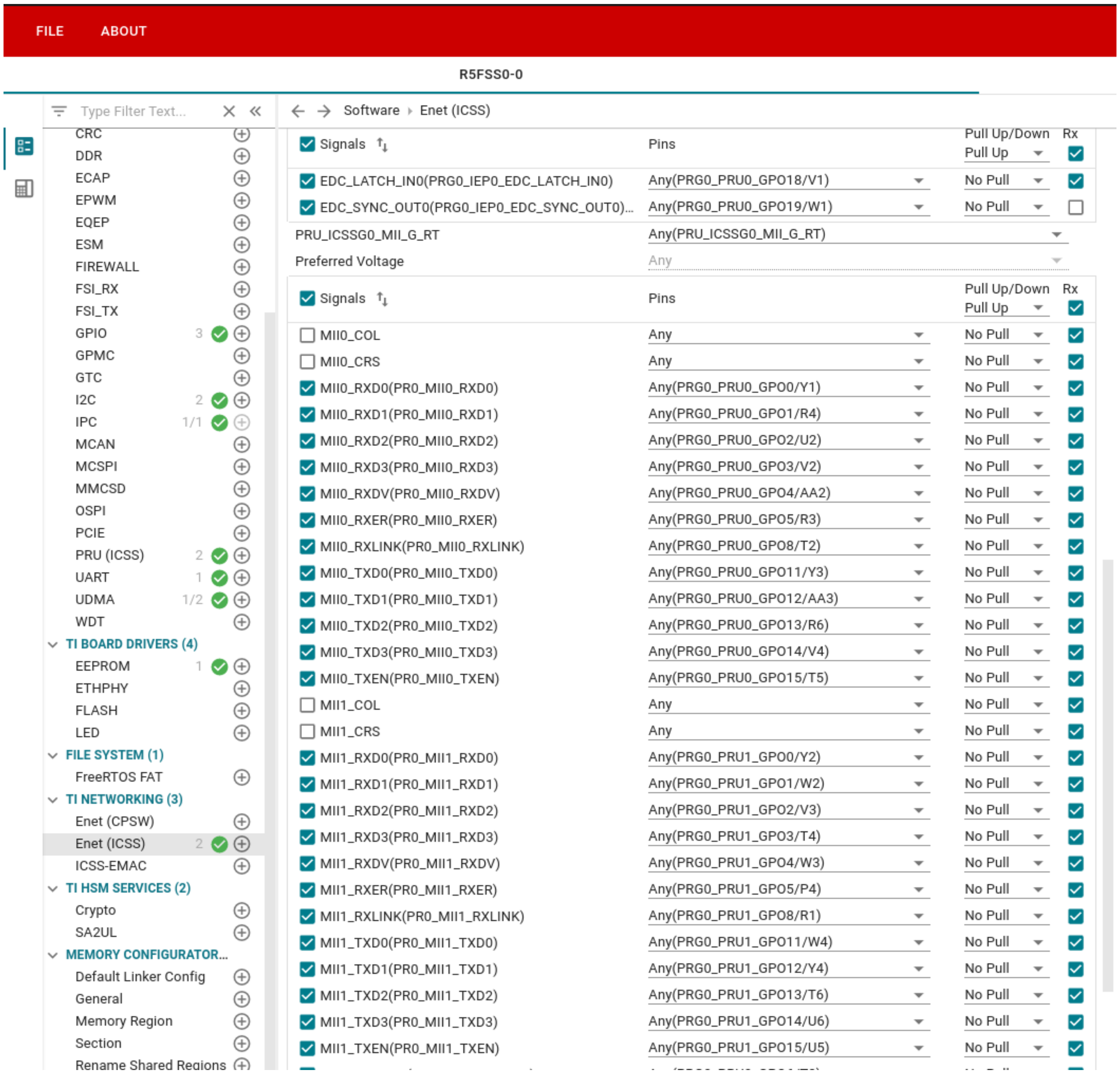

START
R5F0_0
Main Core
R5F0_1
Remote Core
Application boot
Application boot
initial Board Settings and configure SEM board
1. Initialize ICSSG _0 Peripheral
2. Initialize enet driver (enet-lld
3. Initialize LwIP Stack and associate to ICSSG0 peripheral and enet-lld
1. Initialize CPSW3G Periheral
2. Initialize enet driver (enet-lld
3. Initialize LwIP Stack and associate to CPSW3G peripheral and enet-lld
Configure ICSSG0 PORT1 (MII) and PORT2 (MII)
Configure CPSW3G PORT1 (MII) and PORT2 (MII)
1. Configure and initilze Netif with DHCP Client
2. * . Initlzation of LwIP Bridge
1. Configure and initilze Netif with DHCP Client
2. * . Initlzation of LwIP Bridge
Wait for :Linkup and IP allocation from DHCP server
Wait for :Linkup and IP allocation from DHCP server
Start TCP Server socket binding to (PORT 8888) and Its assigned IP
Initialize a TCP Client Application
Send IPC Sync Msg
Wait for IPC Sync Msg
Wait for IPC Sync Ack Msg
Send IPC Pong
R5F0_0 and R5F_0_1 are in sync
Send R5F0_0 application IP address to R5F0_1 core via IPC
Waits for IPC Msg
Recevied IPC Msg with R5F0_1 IP Addr
Wait for TCP Connect from Client
TCP Client: Send a TCP packet with known payload to R5F0_0 IP Address:PORT#
Send the received TCP Packet payload to R5F0_1 via IPC
Waits for IPC Msg
Validate the IPC payload with the TCP Sent Payload in Step A)
TRUE
Loop < 5
FALSE
END

Figure 4-3. Simplified Flow Chart of the Example

5 Debug Steps

1. The linker.cmd file must have the inclusions required for the uncached shared memory region.
   a. This region must be specified in the syscfg.
2. Open the ICSSG1 modules first in the syscfg-GUI.
   a. Next, open the ICSSG0 modules.
3. The ICSSG0 MII pinconfig and the CPSW RGMII1 pinconfig share four pins.
   a. The pins are used for collision detection and CRS for two MII Ethernet ports.
   b. Disable the two signals on the ICSSG0 MII pinconfig.


FILE ABOUT
R5FSS0-0
Type Filter Text...
Software › Enet (ICSS)

CRC, DDR, ECAP, EPWM, EQEP, ESM, FIREWALL, FSI_RX, FSI_TX, GPIO 3 ✓, GPMC, GTC, I2C 2 ✓, IPC 1/1 ✓, MCAN, MCSPI, MMCSD, OSPI, PCIE, PRU (ICSS) 2 ✓, UART 1 ✓, UDMA 1/2 ✓, WDT
TI BOARD DRIVERS (4): EEPROM 1 ✓, ETHPHY, FLASH, LED
FILE SYSTEM (1): FreeRTOS FAT
TI NETWORKING (3): Enet (CPSW), Enet (ICSS) 2 ✓, ICSS-EMAC
TI HSM SERVICES (2): Crypto, SA2UL
MEMORY CONFIGURATOR...: Default Linker Config, General, Memory Region, Section, Rename Shared Regions

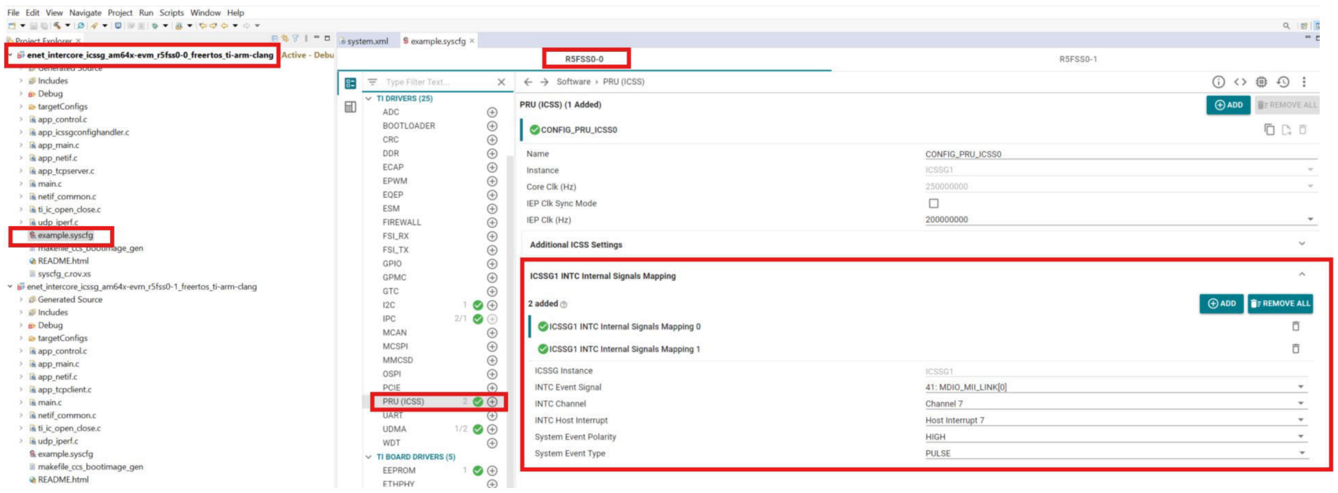
☑ Signals	Pins	Pull Up/Down (Pull Up)	Rx ☑
☑ EDC_LATCH_IN0(PRG0_IEP0_EDC_LATCH_IN0)	Any(PRG0_PRU0_GPO18/V1)	No Pull	☑
☑ EDC_SYNC_OUT0(PRG0_IEP0_EDC_SYNC_OUT0)...	Any(PRG0_PRU0_GPO19/W1)	No Pull	☐

PRU_ICSSG0_MII_G_RT: Any(PRU_ICSSG0_MII_G_RT)
Preferred Voltage: Any

☑ Signals	Pins	Pull Up/Down (Pull Up)	Rx ☑
☐ MII0_COL	Any	No Pull	☑
☐ MII0_CRS	Any	No Pull	☑
☑ MII0_RXD0(PR0_MII0_RXD0)	Any(PRG0_PRU0_GPO0/Y1)	No Pull	☑
☑ MII0_RXD1(PR0_MII0_RXD1)	Any(PRG0_PRU0_GPO1/R4)	No Pull	☑
☑ MII0_RXD2(PR0_MII0_RXD2)	Any(PRG0_PRU0_GPO2/U2)	No Pull	☑
☑ MII0_RXD3(PR0_MII0_RXD3)	Any(PRG0_PRU0_GPO3/V2)	No Pull	☑
☑ MII0_RXDV(PR0_MII0_RXDV)	Any(PRG0_PRU0_GPO4/AA2)	No Pull	☑
☑ MII0_RXER(PR0_MII0_RXER)	Any(PRG0_PRU0_GPO5/R3)	No Pull	☑
☑ MII0_RXLINK(PR0_MII0_RXLINK)	Any(PRG0_PRU0_GPO8/T2)	No Pull	☑
☑ MII0_TXD0(PR0_MII0_TXD0)	Any(PRG0_PRU0_GPO11/Y3)	No Pull	☑
☑ MII0_TXD1(PR0_MII0_TXD1)	Any(PRG0_PRU0_GPO12/AA3)	No Pull	☑
☑ MII0_TXD2(PR0_MII0_TXD2)	Any(PRG0_PRU0_GPO13/R6)	No Pull	☑
☑ MII0_TXD3(PR0_MII0_TXD3)	Any(PRG0_PRU0_GPO14/V4)	No Pull	☑
☑ MII0_TXEN(PR0_MII0_TXEN)	Any(PRG0_PRU0_GPO15/T5)	No Pull	☑
☐ MII1_COL	Any	No Pull	☑
☐ MII1_CRS	Any	No Pull	☑
☑ MII1_RXD0(PR0_MII1_RXD0)	Any(PRG0_PRU1_GPO0/Y2)	No Pull	☑
☑ MII1_RXD1(PR0_MII1_RXD1)	Any(PRG0_PRU1_GPO1/W2)	No Pull	☑
☑ MII1_RXD2(PR0_MII1_RXD2)	Any(PRG0_PRU1_GPO2/V3)	No Pull	☑
☑ MII1_RXD3(PR0_MII1_RXD3)	Any(PRG0_PRU1_GPO3/T4)	No Pull	☑
☑ MII1_RXDV(PR0_MII1_RXDV)	Any(PRG0_PRU1_GPO4/W3)	No Pull	☑
☑ MII1_RXER(PR0_MII1_RXER)	Any(PRG0_PRU1_GPO5/P4)	No Pull	☑
☑ MII1_RXLINK(PR0_MII1_RXLINK)	Any(PRG0_PRU1_GPO8/R1)	No Pull	☑
☑ MII1_TXD0(PR0_MII1_TXD0)	Any(PRG0_PRU1_GPO11/W4)	No Pull	☑
☑ MII1_TXD1(PR0_MII1_TXD1)	Any(PRG0_PRU1_GPO12/Y4)	No Pull	☑
☑ MII1_TXD2(PR0_MII1_TXD2)	Any(PRG0_PRU1_GPO13/T6)	No Pull	☑
☑ MII1_TXD3(PR0_MII1_TXD3)	Any(PRG0_PRU1_GPO14/U6)	No Pull	☑
☑ MII1_TXEN(PR0_MII1_TXEN)	Any(PRG0_PRU1_GPO15/U5)	No Pull	☑

Figure 5-1. SysConfig: ICSSG0 and ICSSG1: MII mode: Allocation to R5FSS0-0

4. Declare the INTC config for the PRUSS module in the syscfg-GUI as follows:
    a. → Pin 41: MII[0] → Channel 7



PRU (ICSS) (1 Added)

CONFIG_PRU_ICSS0

Name	CONFIG_PRU_ICSS0
Instance	ICSSG1
Core Clk (Hz)	250000000
IEP Clk Sync Mode	☐
IEP Clk (Hz)	200000000

ICSSG1 INTC Internal Signals Mapping

2 added

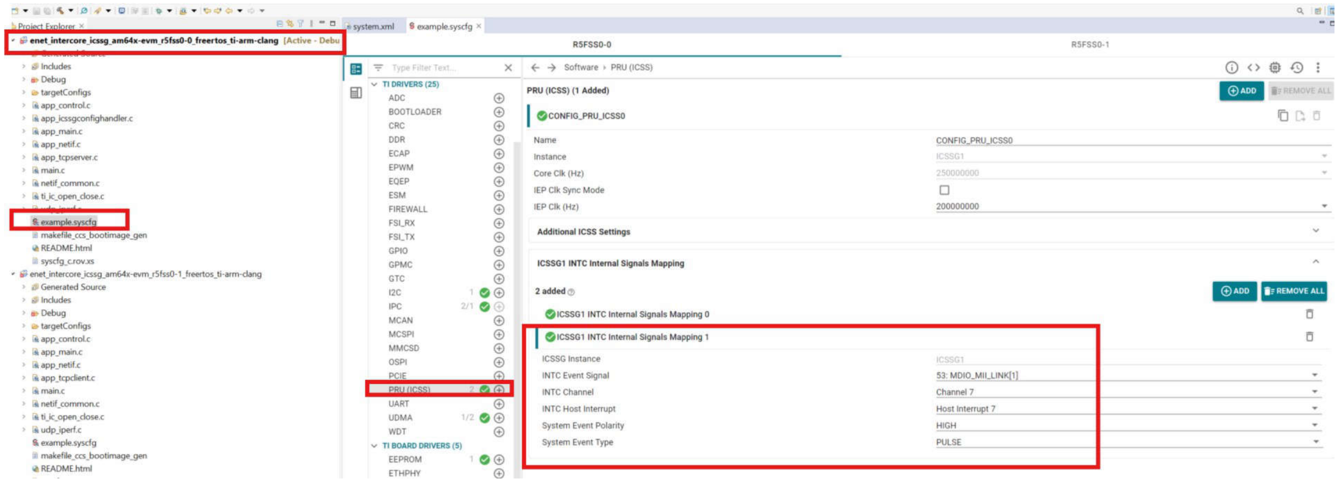
ICSSG1 INTC Internal Signals Mapping 0

ICSSG1 INTC Internal Signals Mapping 1

ICSSG Instance	ICSSG1
INTC Event Signal	41: MDIO_MII_LINK[0]
INTC Channel	Channel 7
INTC Host Interrupt	Host Interrupt 7
System Event Polarity	HIGH
System Event Type	PULSE

Figure 5-2. SysConfig: INTC CONFIG: PRU Pin-41

    b. → Pin 53: MII[1] → Channel 7



PRU (ICSS) (1 Added)

CONFIG_PRU_ICSS0

Name	CONFIG_PRU_ICSS0
Instance	ICSSG1
Core Clk (Hz)	250000000
IEP Clk Sync Mode	☐
IEP Clk (Hz)	200000000

ICSSG1 INTC Internal Signals Mapping

2 added

ICSSG1 INTC Internal Signals Mapping 0

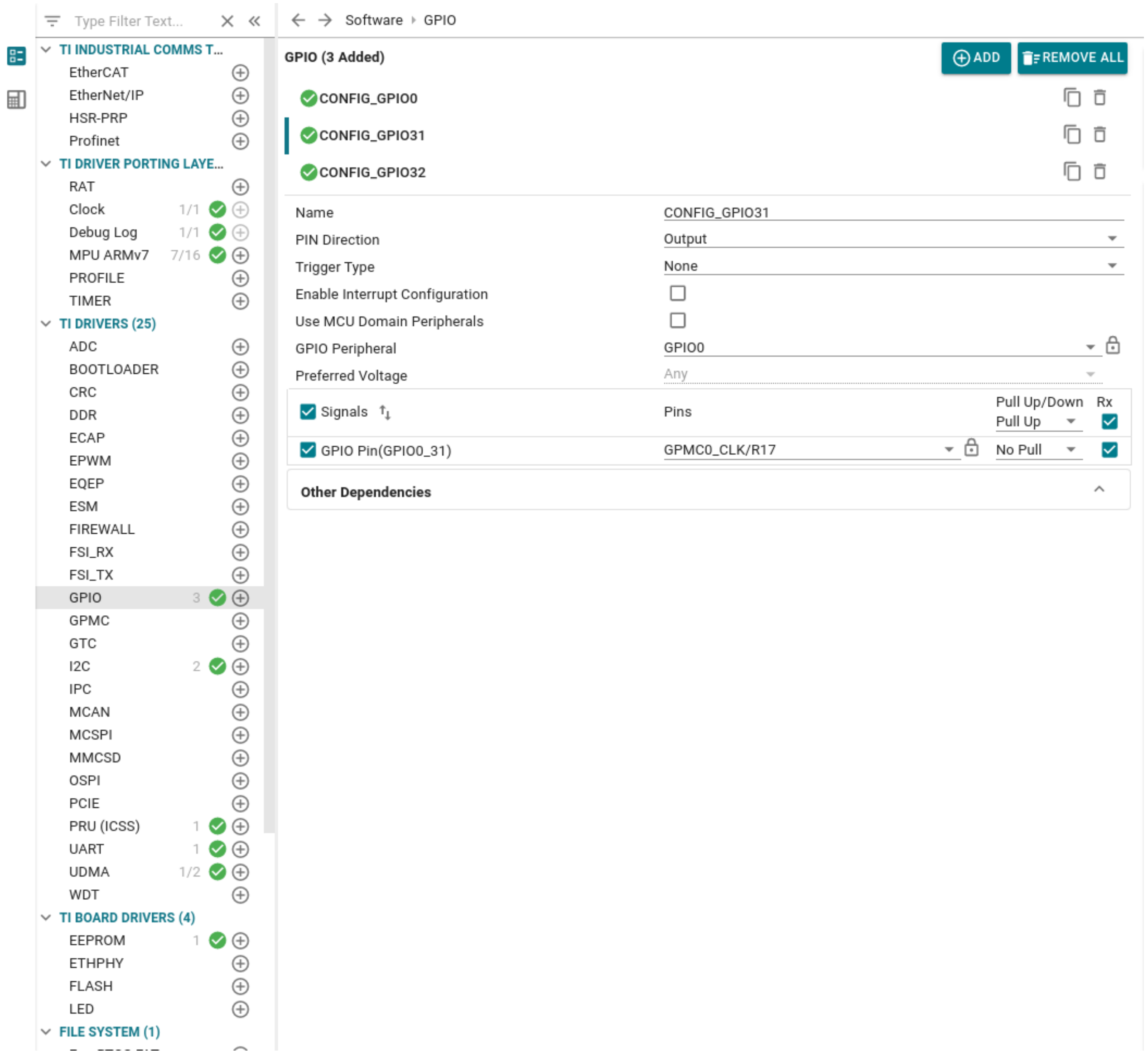
ICSSG1 INTC Internal Signals Mapping 1

ICSSG Instance	ICSSG1
INTC Event Signal	53: MDIO_MII_LINK[1]
INTC Channel	Channel 7
INTC Host Interrupt	Host Interrupt 7
System Event Polarity	HIGH
System Event Type	PULSE

Figure 5-3. SysConfig: INTC CONFIG: PRU Pin-53

5. Make sure to open LwIP in all modules.
6. SEM boards support MII PHYs only (DP83826e).
    a. As a result, only 100M is supported for testing purposes.

7. DP83826e PHYs need reset Pins to be declared in syscfg-GUI with the following configuration:
   a. → AM64x: CONFIG_GPIO_31 → Output → GPIO0 → Pin R17 → Pull Up



Type Filter Text...

Software ▸ GPIO

TI INDUSTRIAL COMMS T...
- EtherCAT
- EtherNet/IP
- HSR-PRP
- Profinet

TI DRIVER PORTING LAYE...
- RAT
- Clock 1/1 ✓
- Debug Log 1/1 ✓
- MPU ARMv7 7/16 ✓
- PROFILE
- TIMER

TI DRIVERS (25)
- ADC
- BOOTLOADER
- CRC
- DDR
- ECAP
- EPWM
- EQEP
- ESM
- FIREWALL
- FSI_RX
- FSI_TX
- GPIO 3 ✓
- GPMC
- GTC
- I2C 2 ✓
- IPC
- MCAN
- MCSPI
- MMCSD
- OSPI
- PCIE
- PRU (ICSS) 1 ✓
- UART 1 ✓
- UDMA 1/2 ✓
- WDT

TI BOARD DRIVERS (4)
- EEPROM 1 ✓
- ETHPHY
- FLASH
- LED

FILE SYSTEM (1)

GPIO (3 Added) ADD REMOVE ALL

- ✓ CONFIG_GPIO0
- ✓ CONFIG_GPIO31
- ✓ CONFIG_GPIO32

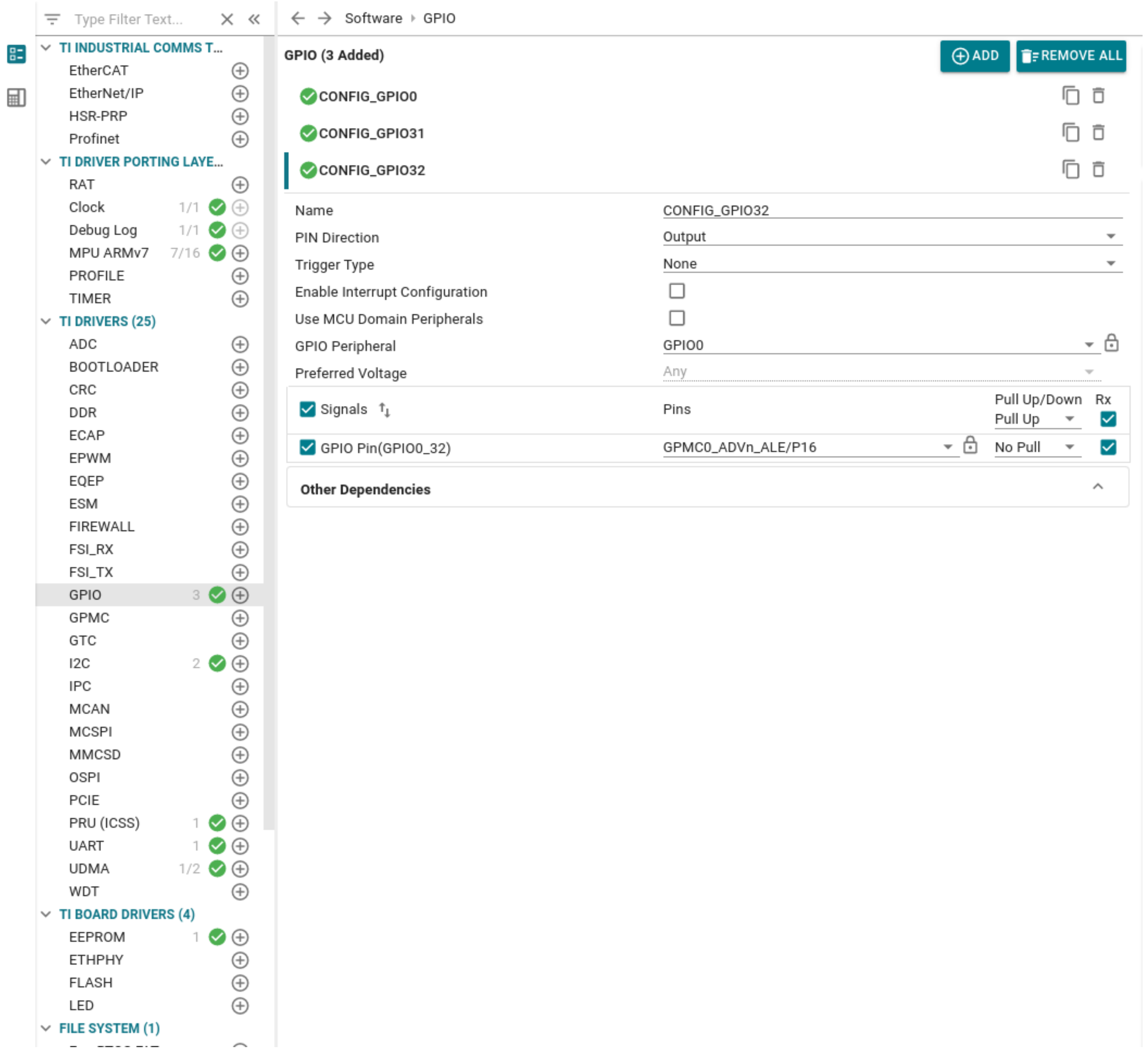
Name	CONFIG_GPIO31
PIN Direction	Output
Trigger Type	None
Enable Interrupt Configuration	☐
Use MCU Domain Peripherals	☐
GPIO Peripheral	GPIO0
Preferred Voltage	Any

Signals	Pins	Pull Up/Down	Rx
✓		Pull Up	✓
✓ GPIO Pin(GPIO0_31)	GPMC0_CLK/R17	No Pull	✓

Other Dependencies

Figure 5-4. SysConfig: GPIO Pin 31: DP83826e PHYs Reset Pins

b. CONFIG_GPIO_32 → Output → GPIO0 → Pin P16 → Pull Up



Name	CONFIG_GPIO32
PIN Direction	Output
Trigger Type	None
Enable Interrupt Configuration	☐
Use MCU Domain Peripherals	☐
GPIO Peripheral	GPIO0
Preferred Voltage	Any

Signals	Pins	Pull Up/Down	Rx
✓ Signals		Pull Up	✓
✓ GPIO Pin(GPIO0_32)	GPMC0_ADVn_ALE/P16	No Pull	✓

Figure 5-5. SysConfig: GPIO Pin 32: DP83826e PHYs Reset Pins

8. Enable the IPC for the active cores only.
   a. Do not have mismatched IPC configurations open for different cores.
   b. Make sure that the IPC is configured the same way for all cores.

6 Reference Logs

Main Core:

```
==========================
ICSSG LWIP TCP ECHO SERVER
==========================
EXT PHY Reset
Enabling clocks!
Enabling clocks!
Enabling clocks!
Enabling clocks!

Init  configs EnetType:1, InstId :2
----------------------------------------------
PHY 3 is alive

Init  configs EnetType:1, InstId :3
----------------------------------------------
EnetPhy_bindDriver: PHY 3: OUI:080028 Model:0f Ver:01 <-> 'dp83869' : OK
PHY 3 is alive
PHY 15 is alive

Init  configs EnetType:1, InstId :0
----------------------------------------------
EnetPhy_bindDriver: PHY 15: OUI:080028 Model:0f Ver:01 <-> 'dp83869' : OK
PHY 1 is alive
PHY 3 is alive

Init  configs EnetType:1, InstId :1
----------------------------------------------
EnetPhy_bindDriver: PHY 3: OUI:080028 Model:11 Ver:01 <-> 'generic' : OK
PHY 1 is alive
PHY 3 is alive
Starting lwIP, local interface IP is dhcp-enabled
[LWIPIF_LWIP] NETIF INIT SUCCESS
Host MAC address-0 : 34:08:e1:80:a9:36
[LWIPIF_LWIP] Enet has been started successfully
[0]Enet IF UP Event. Local interface IP:0.0.0.0
[LWIPIF_LWIP] NETIF INIT SUCCESS
Host MAC address-1 : 70:ff:76:1e:9c:4e
[1]Enet IF UP Event. Local interface IP:0.0.0.0
[LWIPIF_LWIP] NETIF INIT SUCCESS
Host MAC address-2 : 70:ff:76:1e:9c:4f
[2]Enet IF UP Event. Local interface IP:0.0.0.0
[LWIPIF_LWIP] NETIF INIT SUCCESS
Host MAC address-3 : 70:ff:76:1e:9c:50
[3]Enet IF UP Event. Local interface IP:0.0.0.0
[0]Waiting for network UP ...
EnetPhy_bindDriver: PHY 1: OUI:080028 Model:11 Ver:01 <-> 'generic' : OK
Icssg_handleLinkUp: icssg0-1: Port 1: Link up: 100-Mbps Full-Duplex
[2]Network Link UP Event
[1]Waiting for network UP ...
Icssg_handleLinkUp: icssg1-1: Port 1: Link up: 1-Gbps Full-Duplex
[0]Network Link UP Event
[2]Waiting for network UP ...
[3]Waiting for network UP ...
[0]Waiting for network UP ...
[1]Waiting for network UP ...
[2]Enet IF UP Event. Local interface IP:10.24.68.135
[0]Enet IF UP Event. Local interface IP:10.24.68.74
[3]Waiting for network UP ...
[1]Waiting for network UP ...
[3]Waiting for network UP ...
[1]Waiting for network UP ...
[3]Waiting for network UP ...
Network is UP ...
[IPC RPMSG ECHO] Main core start !!!
[IPC RPMSG ECHO] Message exchange started by main core !!!
[IPC RPMSG ECHO] All echoed messages received by main core from 1 remote cores !!!
[IPC RPMSG ECHO] Messages sent to each core = 100
[IPC RPMSG ECHO] Number of remote cores = 1
[IPC RPMSG ECHO] Total execution time = 3332 usecs
[IPC RPMSG ECHO] One way message latency = 16660 nsec
All tests have passed!!
[IPC txn]Remote Server interface IP:10.24.68.74
     30. 59s : CPU load =   8.08 %
     35. 60s : CPU load =   8.41 %
     40. 61s : CPU load =   8.30 %
accepted new connection 810C4280
[IPC_ECHO_BACK] Sent Packet number 0
[IPC_ECHO_BACK] Sent Packet number 0
[IPC_ECHO_BACK] Sent Packet number 0
     45. 62s : CPU load =   8.54 %
[IPC_ECHO_BACK] Sent Packet number 0
[IPC_ECHO_BACK] Sent Packet number 0
accepted new connection 810C4280
[IPC_ECHO_BACK] Sent Packet number 0
[IPC_ECHO_BACK] Sent Packet number 0
[IPC_ECHO_BACK] Sent Packet number 0
[IPC_ECHO_BACK] Sent Packet number 0
[IPC_ECHO_BACK] Sent Packet number 0
     50. 63s : CPU load =   8.54 %
     55. 64s : CPU load =   8.40 %
     60. 65s : CPU load =   8.40 %
     65. 66s : CPU load =   8.31 %
     70. 67s : CPU load =   8.39 %
```

Remote Core:

```
Remote Core:
[IPC RPMSG ECHO] Remote core start !!!
[IPC RPMSG ECHO] Remote Core Sending sync messages to main core ... !!!
[IPC RPMSG ECHO] Remote Core waiting for messages from main core ... !!!
Closing Remote Core!!
[IPC txn]Remote Server interface IP:10.24.68.74
==========================
  CPSW LWIP TCP ECHO SERVER
==========================
Enabling clocks!
EnetAppUtils_reduceCoreMacAllocation: Reduced Mac Address Allocation for CoreId:2 From 4 To 1
Mdio_open: MDIO Manual_Mode enabled

EnetPhy_bindDriver: PHY 0: OUI:080028 Model:23 Ver:01 <-> 'dp83867' : OK

PHY 0 is alive
Starting lwIP, local interface IP is dhcp-enabled
[LWIPIF_LWIP] NETIF INIT SUCCESS
Host MAC address-0 : 70:ff:76:1e:9c:51

[0]Enet IF UP Event. Local interface IP:0.0.0.0
[LWIPIF_LWIP] Enet has been started successfully
[0]Waiting for network UP ...
[0]Waiting for network UP ...
Cpsw_handleLinkUp: Port 1: Link up: 1-Gbps Full-Duplex

MAC Port 1: link up
[0]Network Link UP Event
[0]Waiting for network UP ...
[0]Waiting for network UP ...
[0]Waiting for network UP ...
[0]Waiting for network UP ...
[0]Enet IF UP Event. Local interface IP:10.24.69.120
Network is UP ...
 IP eneterd is: 10.24.68.74
<<<< ITERATION 1 >>>>
 Connecting to: :8888
Connection with the server is established
"Hello over TCP 1" was sent to the Server
[IPC_ECHO_BACK]"Hello over TCP 1" was received from the Server: Packet num 1
Successfully received the packet 1
"Hello over TCP 2" was sent to the Server
[IPC_ECHO_BACK]"Hello over TCP 2" was received from the Server: Packet num 2
Successfully received the packet 2
"Hello over TCP 3" was sent to the Server
[IPC_ECHO_BACK]"Hello over TCP 3" was received from the Server: Packet num 3
Successfully received the packet 3
"Hello over TCP 4" was sent to the Server
[IPC_ECHO_BACK]"Hello over TCP 4" was received from the Server: Packet num 4
Successfully received the packet 4
"Hello over TCP 5" was sent to the Server
[IPC_ECHO_BACK]"Hello over TCP 5" was received from the Server: Packet num 5
Successfully received the packet 5
Connection closed
<<<< ITERATION 2 >>>>
 Connecting to: :8888
Connection with the server is established
"Hello over TCP 1" was sent to the Server
[IPC_ECHO_BACK]"Hello over TCP 1" was received from the Server: Packet num 1
Successfully received the packet 1
"Hello over TCP 2" was sent to the Server
[IPC_ECHO_BACK]"Hello over TCP 2" was received from the Server: Packet num 2
Successfully received the packet 2
"Hello over TCP 3" was sent to the Server
[IPC_ECHO_BACK]"Hello over TCP 3" was received from the Server: Packet num 3
Successfully received the packet 3
"Hello over TCP 4" was sent to the Server
[IPC_ECHO_BACK]"Hello over TCP 4" was received from the Server: Packet num 4
Successfully received the packet 4
"Hello over TCP 5" was sent to the Server
[IPC_ECHO_BACK]"Hello over TCP 5" was received from the Server: Packet num 5
Successfully received the packet 5
Connection closed
```

7 Testing for the ICSSG0 and ICSSG1 Functionality

1. Locate the PHY addresses of the add-on board with an MDIO scan or check the corresponding registers in the MDIO-PHY interface.
2. Change the PHY addresses in the SysConfig-GUI for instances of ICSSG0 in the Board Config section to the PHY address locations found on the specific board.
3. Build the following example in the test folder:
   `make -s -C test/networking/lwip/enet_icssg_tcpserver/am64x-evm/r5fss0-0_freertos/ti-arm-clang clean all`

8 ICSSG and CPSW

ICSSG and CPSW can run together on different cores. The current example uses the ICSSG on core R5Fss_0-0 as the main core and CPSW as the remote core. This example uses an independent LwIP stack and consumes more memory than the typical case. This can be replaced with CPSW running a Layer2 test case and ICSSG acting as a remote controller with only one LwIP stack running, and an interface layer between ICSSG and CPSW with IPC as the transfer medium.

```
make -s -C test/networking/lwip/enet_icssg_tcpserver/am64x-evm/system_cpsw_icssg clean all
```

This results in two .out files in the respective test folders.

1. Load the file through CCS and start the ICSSG core first.
2. Start the CPSW core.
   a. Four IPs on the UART console by ICSSG
   b. and one IP on the CCS console are then visible.
3. All ports are in MAC mode out of box.

This example was tested with CPSW, the default LwIP example running on R5Fss_0-1 independently, and tested with iPerf software. Now, Ported the CPSW TCP server example into the Test folder and added IPC remote core functions. This cannot be used in parallel with the default IPC example testing as this uses different Vring config in the ti_drivers_config.c file. Make sure to allocate the correct number of cores to the example combo for testing, and to allocate the correct number of gcounts for the number of RPmessages.

Load this onto R5Fss_0-1 core and test for functionality.

9 Summary

This document details the implementation of how to use five Ethernet ports on AM64x/AM243x using MCU+SDK for users. Users can use two Ethernet ports of ICSSG0, two Ethernet ports of ICSSG1, and one Ethernet port of CPSW3G available on AM64x/AM243x SoC.

10 References

- Texas Instruments, *Ethernet PRU_ICSSG Instance-0 (PRU_ICSSG0) Usage Guide*, webpage.
- AM243x-Academy
- AM64x-Academy
- MCU-PLUS-SDK-AM64x-Ethernet-Networking-Documentation
- MCU-PLUS-SDK-AM64x-Ethernet-Networking-Examples
- MCU-PLUS-SDK-AM243x-Ethernet-Networking-Documentation
- MCU-PLUS-SDK-AM243x-Ethernet-Networking-Examples
- MCU-PLUS-SDK-Git