Application Note

Enabling MAC2MAC Feature on Jacinto7 SoC

ABSTRACT

The DRA8x and TDA4x processor families are based on the evolutionary Jacinto™ 7 architecture, which is targeted at ADAS, gateway, and autonomous vehicle (AV) applications. They leverage the extensive market knowledge accumulated over a decade of TI's leadership in the ADAS processor market. The Jacinto™ processors are equipped with integrated ethernet switches which are designed to provide reliable communication. They are collectively referred to as CPSW_nG where n stands for the number hardware ports available.

In addition, the Jacinto™ 7 devices provide domain isolation for user and critical system applications. The Main domain with its powerful A cores caters to the user space apps, and the MCU domain with dedicated R and M cores provides safety and security features. Each domain is usually equipped with a dedicated CPSW_nG switch with application domain having more ports. Each CPSW_nG can provide x-1 external ports that can connect to different devices and external switches.

Given the cost-sensitive nature of automotive hardware, the PHY on the device is typically dispensed with and the switches are directly connected to each other using a configuration known as MAC2MAC, which ensures BOM reduction and faster link-up time. MAC2MAC is a widely used application scenario in TI Jacinto™ 7 processor and this document explains this with various examples and use cases while providing more technical details.

For the purposes of this discussion and to ease of reading, this application notes uses the DRA829/TDA4 as the SoC; however the same principles apply to any other Jacinto™ 7 processor.

Project collateral and source code discussed in this document can be downloaded from the following URL: https://www.ti.com/lit/zip/SPRAD07

Table of Contents

Trademarks

Jacinto™ is a trademark of Texas Instruments.
All trademarks are the property of their respective owners.

1 Ethernet and CPSW Introduction

1.1 Ethernet Interface

An Ethernet interface consists of three parts: CPU, MAC and PHY, as shown in Figure 1-1. Usually, the DMA controller is a part of the SoC, where dotted line indicates that processor often uses DMA to access data.

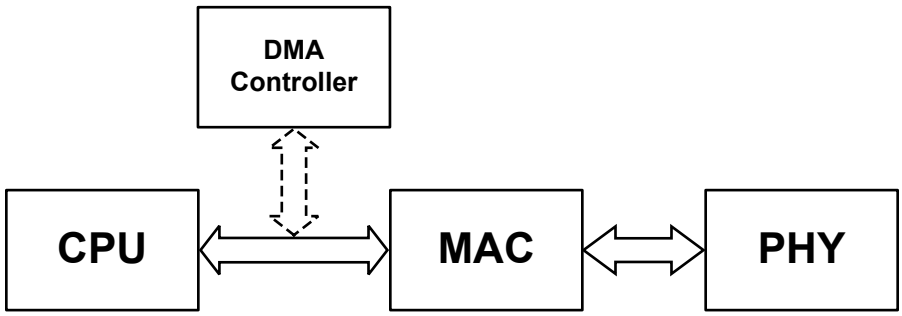

Figure 1-1. Ethernet Interface

CPU is the abbreviation of central processing unit. Ethernet MAC is the abbreviation of Media Access Control, which belongs to the data link layer in the OSI model. PHY is the Physical Interface Device, which belongs to the physical layer in the OSI model. A MAC interfaces with the PHY using a Media Independent Interface (MII). The various types of MIIs include MII, GMII, RMII, RGMII, SGMII, and more. The MII type can vary per SoC and also per port. Refer to the device Technical Reference Manual for details.

1.2 Ethernet Integration

PHY provides the electrical interface between the MAC and the network. Considering the chip area and the mix of digital and analog signals, the following structures are possible:

1. CPU with integrated ethernet MAC and PHY (see Figure 1-2).

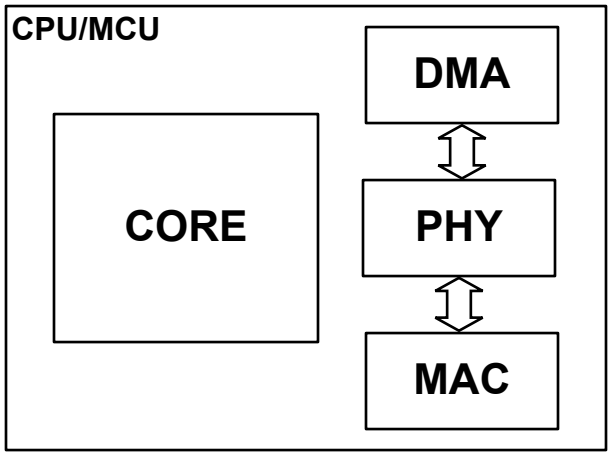

Figure 1-2. CPU With Integrated MAC and PHY

2. CPU with integrated ethernet MAC, PHY is external (see Figure 1-3).

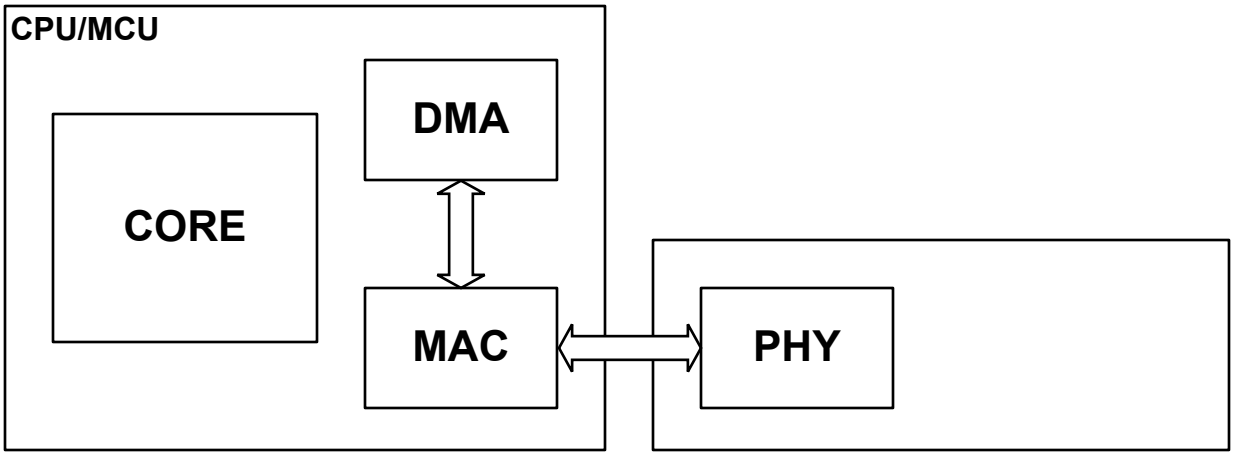

Figure 1-3. CPU With Integrated MAC

3. MAC and PHY are external (see Figure 1-4).

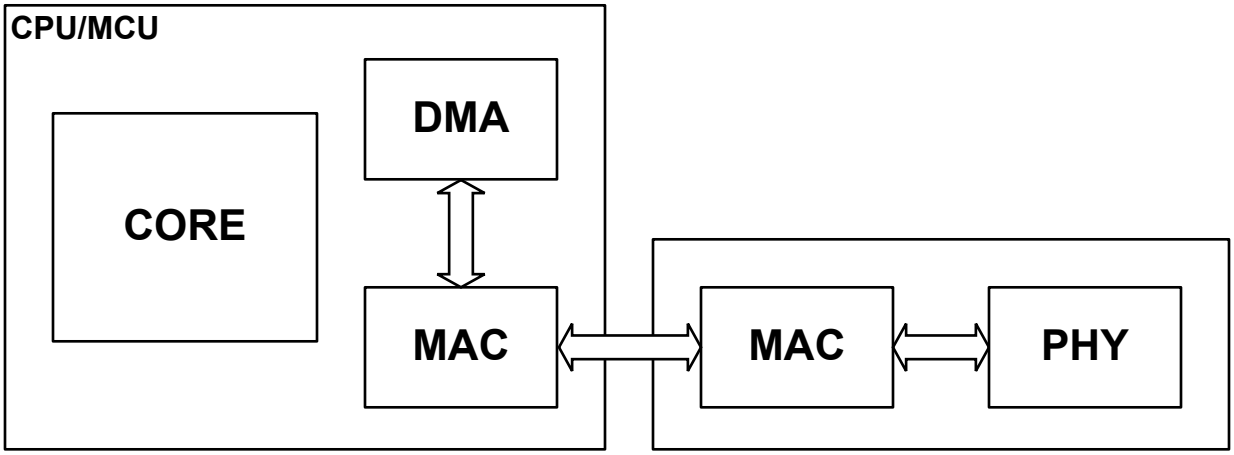

Figure 1-4. Independent MAC and PHY

Out of the three schemes above, the second is the most popular and very few CPU cores come with an integrated PHY. Most CPU cores include only an integrated MAC.

1.3 CPSW

To take the example of DRA829/TDA4, the SoC comes with integrated CPSW9G and CPSW2G interfaces. CPSW is the abbreviation of **C**ommon **P**latform Ethernet **Sw**itch. From DRA829/TDA4 architecture, CPSW9G is in Main domain, and CPSW2G is in MCU domain. CPSW9G has 9 ports: 8 are physical ports, and one is a software port interfacing with the CPU core. Similarly, CPSW2G has two ports: one for the CPU core, and the other a physical port.

All physical ports of these two switches can easily be connected to another device or switch using MAC2MAC.

1.4 MAC2MAC

As explained previously, a MAC is connected to the network using a PHY. But when the objective is to connect to another MAC in a controlled environment such as a PCB, the PHY is often redundant and can increase the project cost. In such a situation the Rx, Tx, and Clock pins are usually connected directly using buffers. This is called a MAC2MAC connection.

2 JACINTO7 MAC2MAC Solution

2.1 Application Requirements

Because Jacinto 7 uses a multicore heterogeneous architecture, it integrates not only an Arm A72 core, but also TI's C7000 and C6000 DSPs, an Arm R5F MCU core, and so on. This multicore heterogeneous architecture not only provides with excellent performance, it also brings great flexibility and convenience to the design.

The architecture of CPSW is shown in Figure 2-1. From the perspective of MAC interface types, the general RGMII (Reduced Gigabit Media Independent Interface (RGMII) and SGMII (Serial GMII) can support 1000M Ethernet.

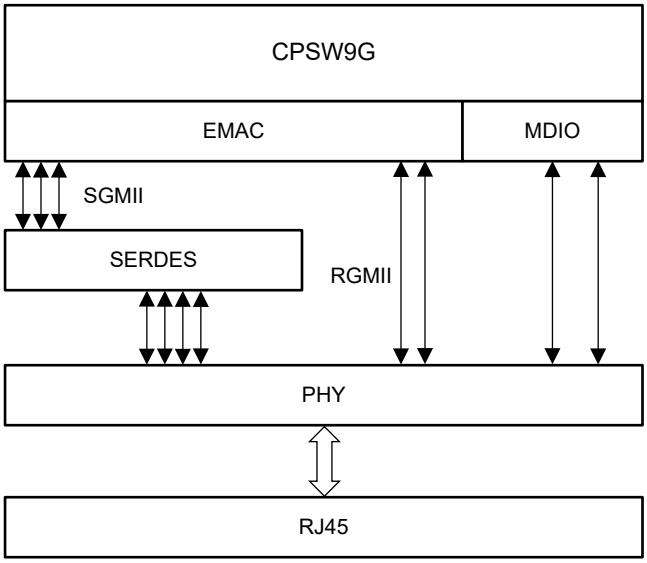

Figure 2-1. CPSW Architecture

RGMII stands for Reduced Gigabit Media-Independent Interface, which is to say pin count is reduced. The clock frequency is still 125MHz, but the TX/RX data width is changed from 8 to 4 bits. In order to keep the same transmission rate of 1Gbps with the reduced data lines, data is sent on both the rising and falling edges of the clock. RGMII is also compatible with speeds of 100Mbps and 10Mbps. For this, the reference clock rates are 25MHz and 2.5MHz respectively. A typical RGMII application connection is shown in Figure 2-2.

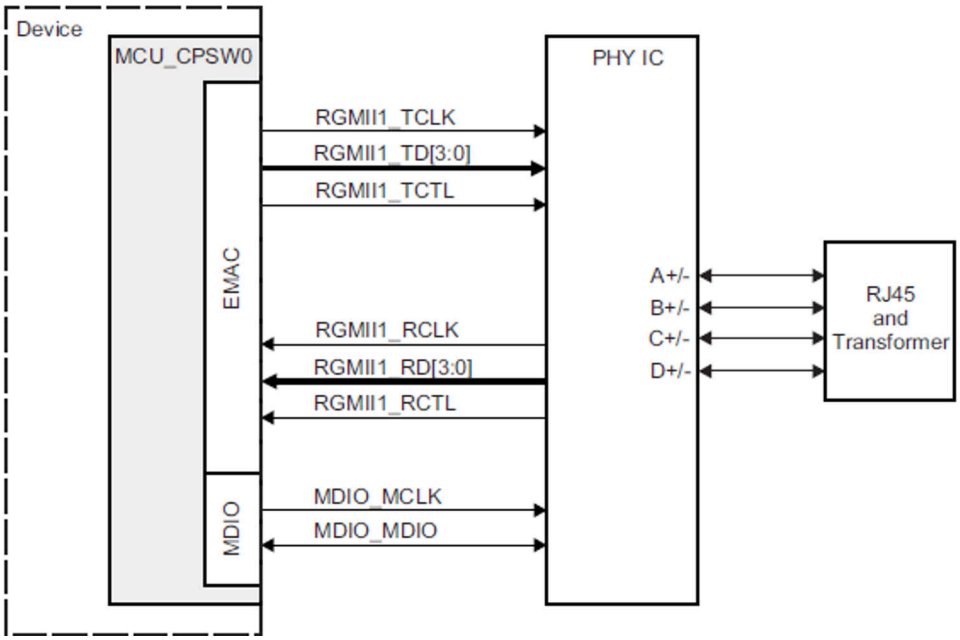

Figure 2-2. RGMII Connection

SGMII stands for Serial GMII, it's used whenever higher speed and lower pin count is desired. The cost is additional complexity in the form of a Serializer, De-serializer module, called SerDes. There are a pair of differential signal lines each for sending and receiving. Clock frequency is 625MHz, which is sampled on both rising and falling edges of the clock signal. Reference clock RX_CLK can be provided by the PHY, but it is optional, mainly used for MAC side when there is no clock. Under normal conditions, RX_CLK is not used, and both transmission and reception can recover the clock from the data. The SGMII interface of most MAC chips can be configured as a SerDes interface (physically compatible, only need to configure the register), directly connected to other modules, without the need for the PHY layer chip, the clock rate is still 625MHz. Figure 2-3 (referenced from J721E TRM), shows how CPSW SGMII can be configured with a SerDes interface.

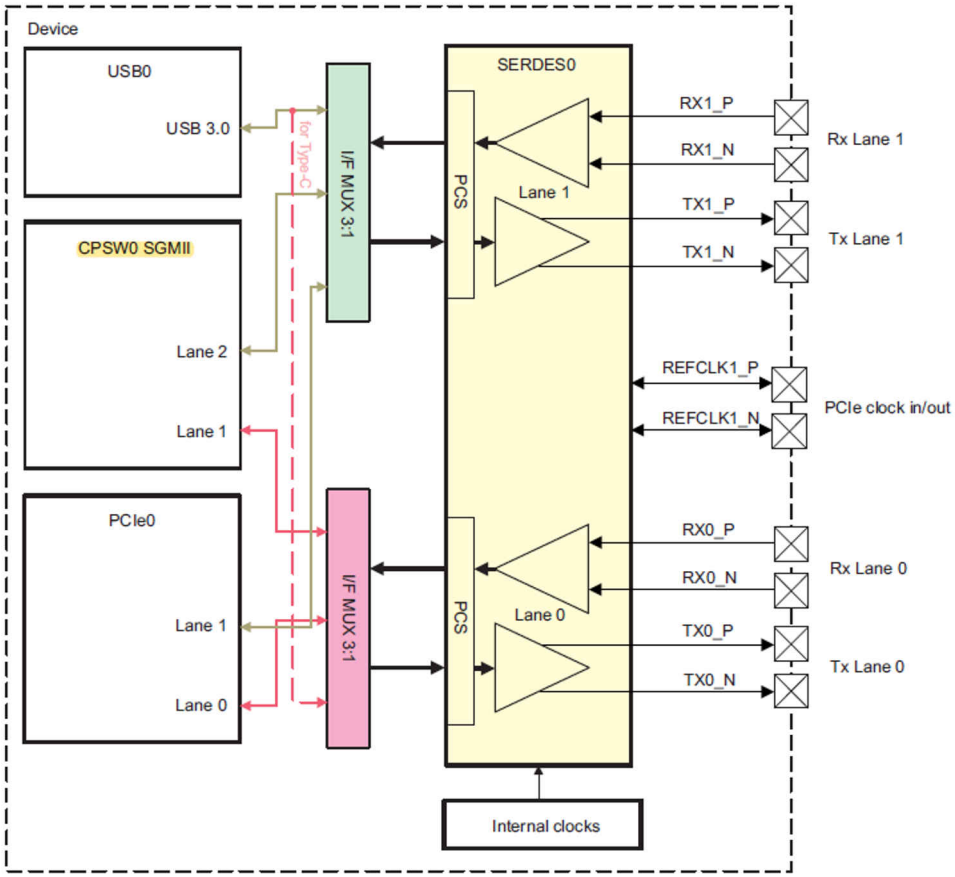

Figure 2-3. Serdes0 Overview

Different SGMII ports corresponds to different SERDES Lanes. Table 2-1 shows this relationship.

Table 2-1. SGMII PORT SERDES

SGMII PORT	SERDES - LANE
PORT 1	SERDES 0 - LAN 0
PORT 2	SERDES 0 - LAN 1
PORT 3	SERDES 1 - LAN 0
PORT 4	SERDES 1 - LAN 1
PORT 5	SERDES 4 - LAN 0
PORT 6	SERDES 4 - LAN 1
PORT 7	SERDES 4 - LAN 2
PORT 8	SERDES 4 - LAN 3

Therefore, there are usually two ways of MAC2MAC interconnection solution, RGMII based and SGMII based. There can be many application scenarios for MAC2MAC, here we explain 3 of them:

1. Jacinto7 is divided two parts, MCU domain and Main domain. CPSW2G is usually allocated to MCU domain while CPSW9G is usually used by main domain cores, A72 and R5F. The MCU domain usually uses AUTOSAR OS due to functional safety requirements. In addition to the internal data path, from the outside, it is also very convenient to connect to the two switches via MAC2MAC. The system architecture is shown in Figure 2-4.

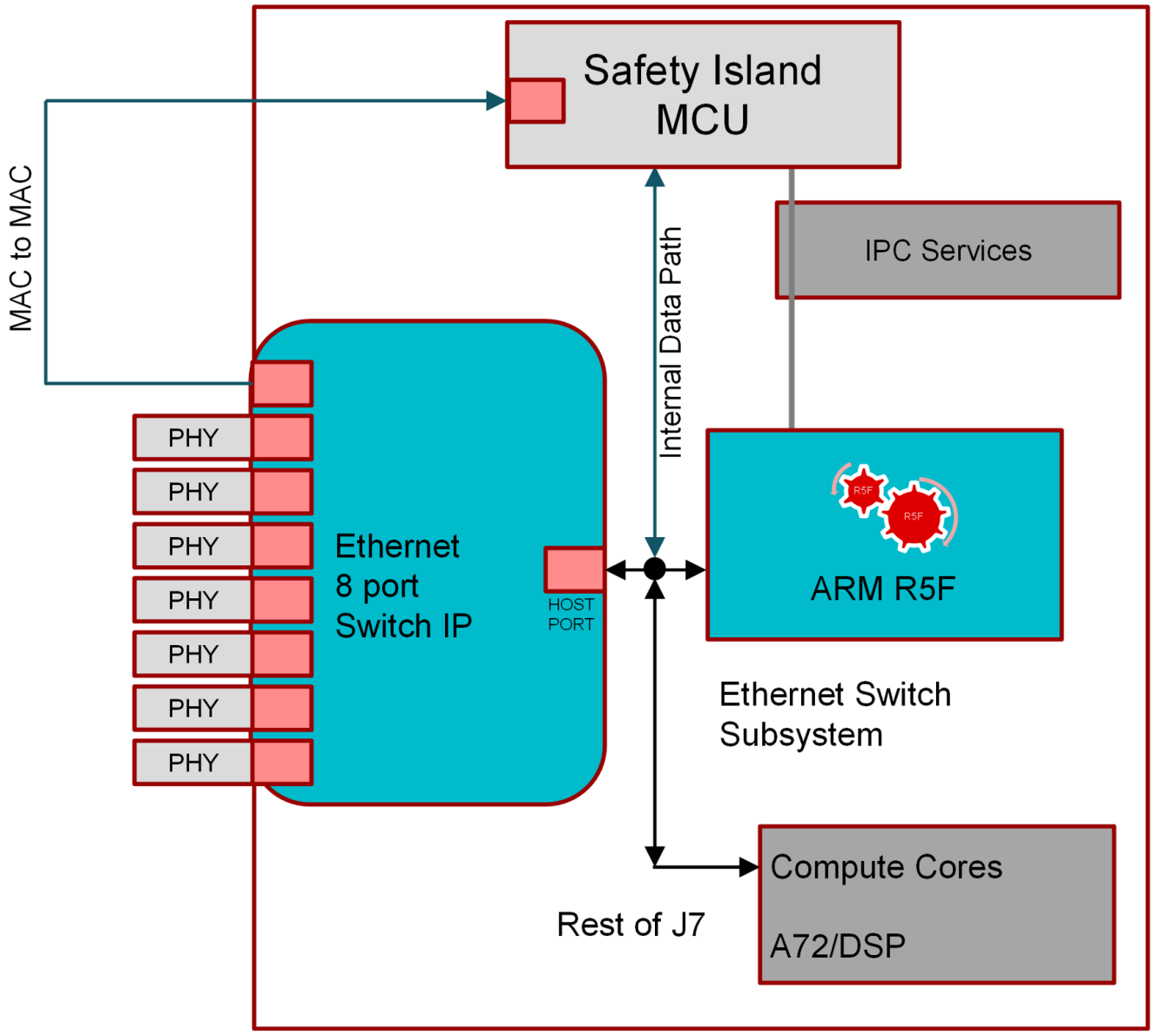

Figure 2-4. CPSW9G and CPSW2G MAC2MAC

2. When the performance of a single Jacinto7 processor cannot meet the system requirements, it is very convenient to integrate multiple Jactinto7 SOC's on a single board. In such a scenario it helps to connect the CPSW switches over MAC2MAC. The system block diagram is shown in Figure 2-5.

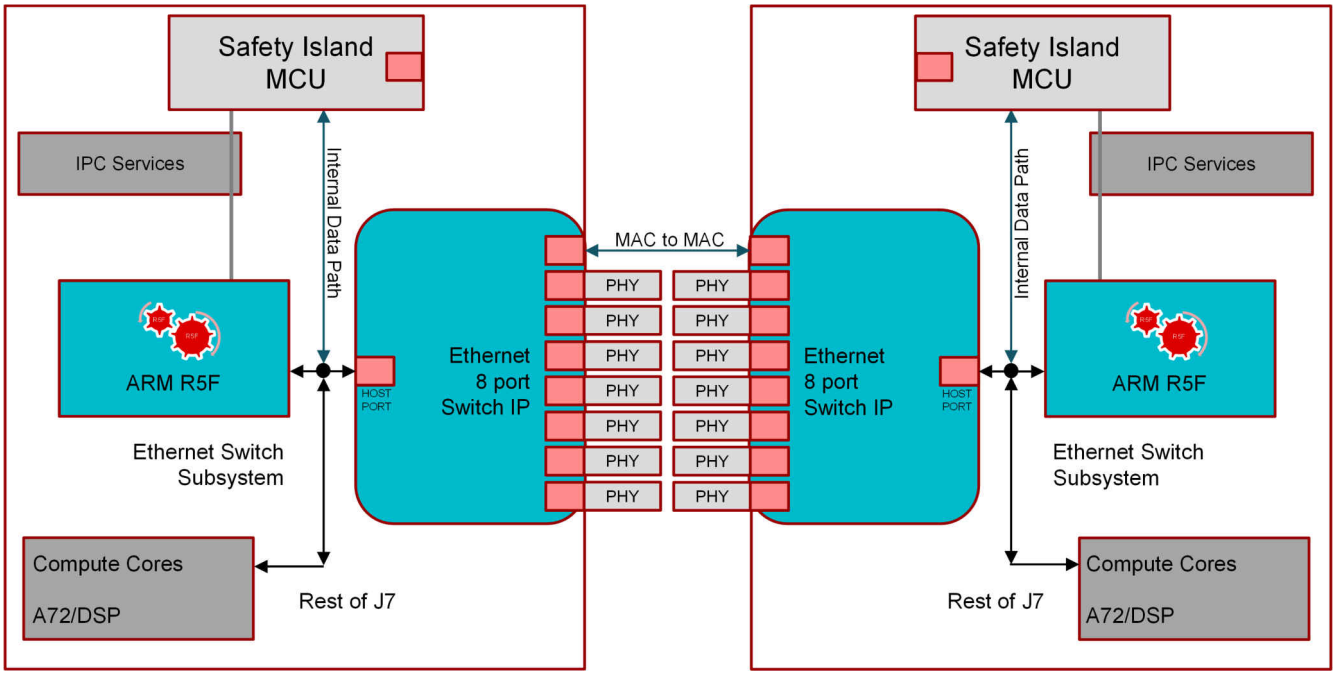

Figure 2-5. MAC2MAC Between SOC to SOC

3. MAC2MAC can be used to connect external switches or the MAC's on other SoC's. It serves two goals, first, it increases the port count, second, it can be used to inter-connect SoC's. The system block diagram is shown in Figure 2-6.

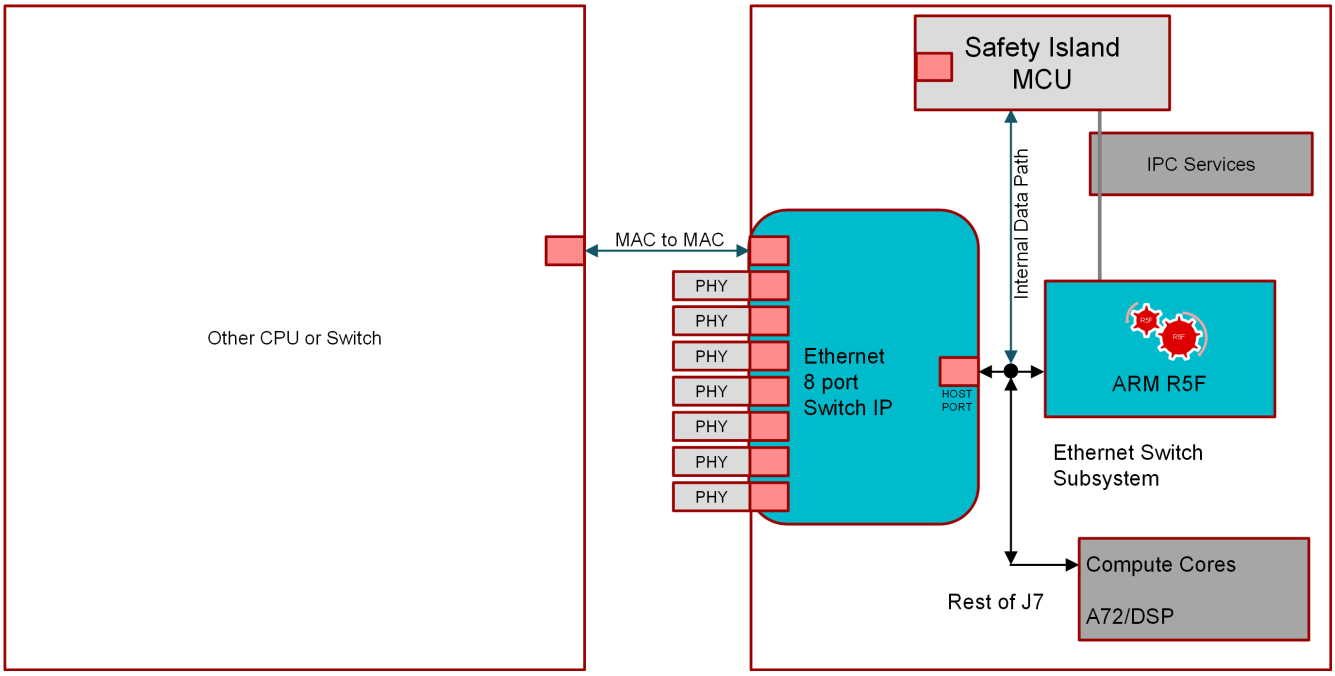

Figure 2-6. External Soc Connect to TDA4VM

2.2 RGMII Solution

Here, let's the check the first example that used a CPSW2G and CPSW9G interconnection. The internal MAC and MAC are interconnected through RGMII. RGMII is relatively simpler than SGMII, and MAC is directly connected to MAC. The connection is shown in Figure 2-7.

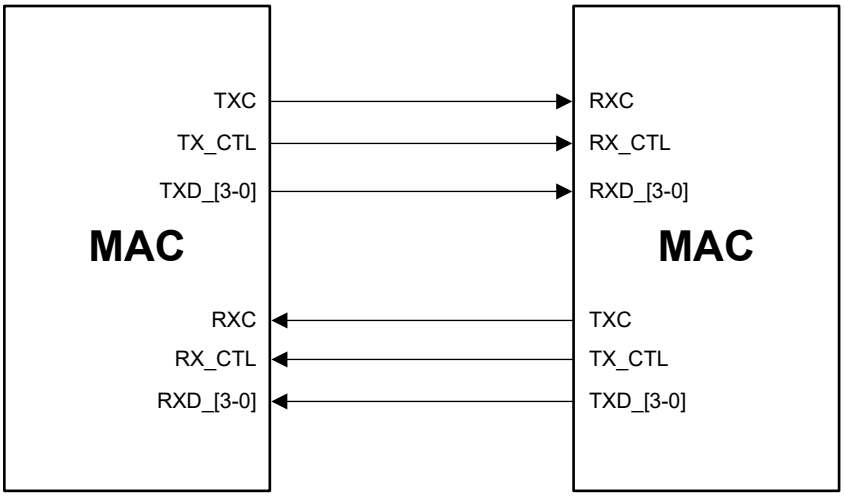

Figure 2-7. RGMII MAC2MAC

2.3 SGMII Solution

Here we give an example of SGMII interconnection. The two TDA4VMs are interconnected through SGMII MAC2MAC. Compared to RGMII, SGMII needs to additionally configure the SERDES. The system block diagram is shown as in Figure 2-8.

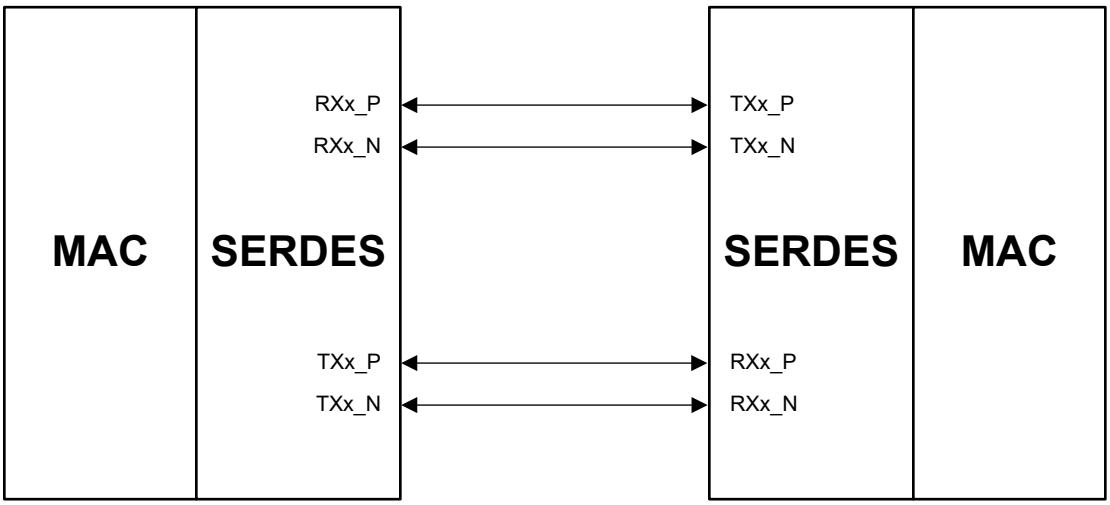

Figure 2-8. SGMII MAC2MAC

3 MAC2MAC Implementation

3.1 RGMII Implementation

Below, we discuss the CPSW2G and CPSW9G interconnection use-case mentioned previously. CPSW2G is controlled by A72 and CPSW9G is controlled by Main R5F, and CPSW2G and CPSW9G are connected through a MAC2MAC link. Configuration mode is RGMII 1000M, Full duplex.

3.1.1 CPSW 2G Changes

Because CPSW 2G has a native Linux driver (QNX use-case is not covered in this paper), it has a corresponding device tree node that refers to the PHY. As there is no PHY involved in a MAC2MAC connection, we will leverage the fixed-link capability to configure the link properties at init. The PHY state machine logic looking for link status change notifications will also be disabled in this case.

Changes for this are shown below. A reference patch (0001-*.patch for ti-processor-sdk-linux-j7-evm-07_03_00_05) can be downloaded from project collateral and source code link.

Note

Apply 0001-*.patch on ti-processor-sdk-linux-j7-evm-07_03_00_05. After applying the patch, rebuild the dtb that using 'make linux-dtbs'. Next, update the new dtb file to the boot partition.

For default ti-processor-sdk-linux-j7-evm-07_03_00_05, we configure CPSW2G as PHY mode. So, we need to remove MDIO and PHY configuration.

3.1.2 CPSW 9G Changes

CPSW 9G is controlled by Ethernet Firmware running on Main R5F 2_0, so all changes must be done in Ethernet Firmware and ENET-LLD. The main changes required are in the port open sequence, which deals with getting port parameters and configuring the PHY. A reference patch (0002-*.patch for ti-processor-sdk-rtos-j721e-evm-07_03_00_07) can be downloaded from project collateral and source code link.

Note

Apply 0002-*.patch on ti-processor-sdk-rtos-j721e-evm-07_03_00_07. After applying the patch, rebuild the PDK/ETHFW/VISION_APPS that are using 'make pdk ; make ethfw; make vision_apps', Next, run 'make linux_fs_install' to update the images to targetfs.

1. Set the port number in *EnetPer_PortLinkCfg::macPort*.
2. Set the MAC port interface (RMII, RGMII, SGMII, etc.) through the layer, sublayer and variant fields of *EnetPer_PortLinkCfg::mii*.
3. Initialize the MAC configuration parameters using *CpswMacPort_initCfg()* and manually overwrite any parameters that differ from default ones.
4. Set PHY address to *ENETPHY_INVALID_PHYADDR* to indicate that this is a PHY-less connection
5. Set the link speed and duplexity configuration in *EnetPer_PortLinkCfg::linkCfg* to match that of the partner MAC port. The speed and duplexity must be fixed values, they can't be *ENET_SPEED_AUTO* or *ENET_DUPLEX_AUTO* as they are used for auto-negotiation which is not relevant in MAC-to-MAC mode
6. When all the MAC and PHY configurations are done, the ports can be opened by calling the *ENET_PER_IOCTL_OPEN_PORT_LINK* command
7. If all the above steps succeeded without any errors, then the Enet driver, the Ethernet peripheral and a MAC port have been opened successfully
8. When link up is detected, the application should set the ALE port state to Forward state using *CPSW_ALE_IOCTL_SET_PORT_STATE IOCTL* command and enable the host port of CPSW by calling the *ENET_HOSTPORT_IOCTL_ENABLE* command.

The changes described are from ENET-LLD point of view however in the Ethernet Firmware context, the implementation of these steps is split between Ethernet Firmware and Enet LLD utils library. Here is how Ethernet Firmware and ENET-LLD interact for the port configuration.

When configuring a port in Ethernet Firmware, the following sequence occurs:

1. *EthFw_initLinkArgs()* is called by Enet MCM (Multiclient Manager) to get the port configuration parameters to be used for a given port.

2. Ethernet Firmware populates common parameters in *EthFw_initLinkArgs()* such as CPSW MAC port default config, and speed/duplexity.
   a. Speed and duplexity are currently set to 1 Gbps full-duplex for MAC-to-MAC, you can change this if needed.
3. Ethernet Firmware calls *EnetBoard_setPhyConfig()* to have the interface type and PHY configuration parameters set (such as PHY address).
4. *EnetBoard_setPhyConfig()* is an Enet LLD board utils function.
   a. This function is where one needs to make changes for MAC-to-MAC.
5. A reference patch is being provided with the changes in *EnetBoard_setPhyConfig()* where MAC port 4 is the desired port used for MAC 2 MAC connection. A reference patch(0003-*.patch) can be downloaded from project collateral and source code link.
6. When all changes are done, compile Ethernet Firmware and load again. If Linux is being used then copy the firmware (*app_remoteswitchcfg_server_strip.xer5f*) to */lib/firmware/ethfw* in *rootfs* partition and reboot.

3.2 SGMII Implementation

Here is an example of interconnecting two TDA4VMs from TI's CPSW9G to CPSW9G. Connect one TDA4VM CPSW9G and another TDA4VM CPSW9G over SGMII MAC2MAC as described in Figure 12. As CPSW9G is controlled by Ethernet Firmware the only changes required are to be done there. In that sense most of the changes required are similar to what is required for RGMII (as described above) except the part where MAC port interface is configured to RGMII. The only additional changes required are: A reference patch (0004-*.patch for ti-processor-sdk-rtos-j721e-evm-07_03_00_07) can be downloaded from project collateral and source code link.

Note

Please apply 0004-*.patch on ti-processor-sdk-rtos-j721e-evm-07_03_00_07. Once you apply the patch, you should rebuild the PDK/ETHFW/VISION_APPS that using `make pdk ; make ethfw; make vision_apps`, Then, you need to run `make linux_fs_install` to update the images to targetfs.

1. Setting SERDES's clocks and setting SERDES
2. Mapping the SERDES lanes and CPSW ports as per Table 1. SGMII PORT SERDES table

Care must be taken to ensure that SERDES lanes do not conflict with PCIe or USB.

3.3 MAC2MAC Debug

Problems can occur during the bring-up process. The problems can be classified into three categories in order of their likelihood:

1. Software and configuration issues
2. Pinmux issues
3. Hardware issues

3.3.1 Software and Configuration Issues

These happen when any one or multiple issues occur:

1. Changes in Ethernet Firmware or ENET-LLD have not been done correctly
2. Linux DT changes have not been done correctly
3. SERDES muxing is not done correctly as per recommendation (for SGMII only)
4. RGMII Connection should pay attention to set the recommended RGMII delay through the ENETn_CTRL Register.

Typically, they manifest themselves as errors in software initialization sequence. For example, some IOCTL might fail or the initialization routine might return an error. In most cases it results in complete failure and the port not working.

3.3.2 Debugging Software and Configuration Issues

To debug such issues, first verify the Linux bootlog and Ethernet Firmware log, make sure that there is no "error" keyword in either of them.

Ethernet Firmware logs are available on the 3rd Main UART instance, if that is not accessible, then one can also get them from the Linux terminal by running the command "*cat /sys/kernel/debug/remoteproc/remoteproc*/trace0"*

A sample Ethernet Firmware log (ethfw_freertos_log.txt for ti-processor-sdk-rtos-j721e-evm-07_03_00_07) can be downloaded from project collateral and source code link. Note that it shows:

1. Successful ENET registration
2. Successful remote core initialization
3. Phy initialization and Link up
4. Rx Flow and VLAN Routing configuration

Depending on the configuration enabled, the log might vary, but these basic steps will remain the same.

One can also dump the contents of various registers by running the GEL files present in *pdk/packages/ti/drv/enet/tools/debug_gels*

The details on the GEL files are provided below:

1. **cpsw_ale_print_table.gel**: Used to print ALE configuration. Can be used to check if the ALE configuration was successful
2. **cpsw_enetctrl_cfg.gel**: This is important for debugging SGMII and RGMII configurations. It prints information from ENET Control registers which control whether RGMII or SGMII mode is configured.
3. **cpsw_mac.gel**: This GEL file is used to dump information pertaining to duplexity, flow enablement etc. Relevant for MAC 2 MAC configuration
4. **cpsw_mdio_config.gel**: This is used to print MDIO configuration, not relevant for MAC 2 MAC debug
5. **cpsw_print_reg.gel**: It is a superset of all other GEL's and can be used to print all CPSW registers.
6. **cpsw_sgmii_diag.gel**: This is used to print SERDES configuration and it's an important GEL file for **debugging SGMII**. Users should run this and verify that the SERDES configuration is what they have configured.

cpsw_stats.gel: Used to dump statistics, number of frames received/transmitted from all ports (including host port). Users should run this to check if there are any error frames and if all packets are being received/sent correctly.If ping isn't working, then dump the statistics first and figure out whether the issue is with Receive or Transmit or both.

3.3.3 Pinmux Issues

Pinmux issues occur when:

1. The pins have not been set correctly, for example if RGMII Tx pins are configured for PCIe or something else
2. Pinmux files have not been imported correctly into bootloader.
3. Another core or software is overwriting the pin-muxing pads.

Pinmux issues are not specific to MAC2MAC, but can create generic issues which are harder to catch and will manifest themselves as a disabled clock or some data line not toggling. Once software issues have been ruled out it's better to connect a scope to the data and clock lines and make sure that they are toggling as per standard.

Pinmux issues can result in error statistics like short frames, SFD errors etc. This doesn't occur with software issues.

To avoid such problems:

1. Use the pinmux tool and avoid manual editing
2. Dump the pad registers after complete boot-up and make sure that the values are on expected lines
3. Use a pinmux similar to EVM as much as possible, this will help avoid errors where pads get over-written by other drivers.

3.3.4 Hardware Issues

Hardware issues refer to routing errors on the board, issues with buffers, crystals or capacitance. To avoid these, get the schematic reviewed by TI. Your TI representative will help with that.

These issues usually manifest themselves as clock skew, bad signals, errors in statistics. If running *pdk/packages/ti/drv/enet/tools/debug_gels/**cpsw_stats.gel***shows error frames, then it's most likely that there are either HW or Pinmux issues. To debug such issues, connect a scope and check the signals.

Below, we describe some register and corresponding settings for RGMII and SGMII, which can be used to check if the configuration has been done correctly. These can also be dumped using the debug gels mentioned previously.

3.3.4.1 RGMII Debugging

1. As shown in Figure 3-1, confirm the corresponding PORT mode setting: RGMII

Table 5-915. CTRLMMR_ENET1_CTRL Instances

Instance	Proxy0 Physical Address	Proxy1 Physical Address
CTRL_MMR0	0010 4044h	0010 6044h

Figure 5-435. CTRLMMR_ENET1_CTRL Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED			RGMII_ID_MODE	RESERVED	PORT_MODE_SEL		
R-0h			R/W-0h	R-0h	R/W-2h		

LEGEND: R = Read Only; R/W = Read/Write; *-n* = value after reset

Table 5-916. CTRLMMR_ENET1_CTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Reserved
4	RGMII_ID_MODE	R/W	0h	Port1 RGMII internal transmit delay selection 0h - Internal transmit delay 1h - Reserved
3	RESERVED	R	0h	Reserved
2-0	PORT_MODE_SEL	R/W	2h	Selects Ethernet switch Port1 interface 0h - GMII/MII (not supported) 1h - RMII 2h - RGMII 3h - SGMII 4h - QSGMII 5h - XFI (not supported) 6h - QSGMII_SUB 7h - Reserved

Figure 3-1. CTRLMMR_ENET1_CTRL Register

2. Confirm the corresponding RGMII FULLDUPLEX/SPEED/LINK setting as shown in Figure 3-2, and make sure Link is up.

Table 12-2139. CPSW_SS_RGMII1_STATUS_REG Instances

Instance	Physical Address
CPSW0_NUSS_SS	0C00 0030h

Figure 12-921. CPSW_SS_RGMII1_STATUS_REG Register

31	30	29	28	27	26	25	24
RESERVED							
R-X							
23	22	21	20	19	18	17	16
RESERVED							
R-X							
15	14	13	12	11	10	9	8
RESERVED							
R-X							
7	6	5	4	3	2	1	0
RESERVED				FULLDUPLEX	SPEED		LINK
R-X				R-0h	R-0h		R-0h

LEGEND: R = Read Only; *-n* = value after reset

Table 12-2140. CPSW_SS_RGMII1_STATUS_REG Register Field Descriptions

Bit	Field	Type	Reset	Description
31-4	RESERVED	R	X	
3	FULLDUPLEX	R	0h	RGMII Port 1 full duplex: 0h = Half-duplex 1h = Full-duplex
2-1	SPEED	R	0h	RGMII Port 1 speed: 0h = 10Mbps 1h = 100Mbps 2h = 1000Mbps 3h = Reserved
0	LINK	R	0h	RGMII Port 1 link indicator: 0h = Link is down, 1h = Link is up

Figure 3-2. CPSW_SS_RGMII1_STATUS_REG Register

3.3.4.2 SGMII Debugging

1. As shown in Figure 3-3, confirm the corresponding PORT mode setting: RGMII

Table 5-915. CTRLMMR_ENET1_CTRL Instances

Instance	Proxy0 Physical Address	Proxy1 Physical Address
CTRL_MMR0	0010 4044h	0010 6044h

Figure 5-435. CTRLMMR_ENET1_CTRL Register

31	30	29	28	27	26	25	24
RESERVED							
R-0h							
23	22	21	20	19	18	17	16
RESERVED							
R-0h							
15	14	13	12	11	10	9	8
RESERVED							
R-0h							
7	6	5	4	3	2	1	0
RESERVED			RGMII_ID_MODE	RESERVED	PORT_MODE_SEL		
R-0h			R/W-0h	R-0h	R/W-2h		

LEGEND: R = Read Only; R/W = Read/Write; -*n* = value after reset

Table 5-916. CTRLMMR_ENET1_CTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	RESERVED	R	0h	Reserved
4	RGMII_ID_MODE	R/W	0h	Port1 RGMII internal transmit delay selection 0h - Internal transmit delay 1h - Reserved
3	RESERVED	R	0h	Reserved
2-0	PORT_MODE_SEL	R/W	2h	Selects Ethernet switch Port1 interface 0h - GMII/MII (not supported) 1h - RMII 2h - RGMII 3h - SGMII 4h - QSGMII 5h - XFI (not supported) 6h - QSGMII_SUB 7h - Reserved

Figure 3-3. CTRLMMR_ENET1_CTRL Register

2. Confirm that the phyAddr of the corresponding port is configured as *CPSW_PHY_INVALID_PHYADDR*
3. As shown in Figure 3-4, confirm the MASTER Mode of the corresponding port.

Table 12-2175. CPSW_SGMII_CONTROL_REG_j Register Field Descriptions

Bit	Field	Type	Reset	Description
31-7	RESERVED	R/W	X	
6	TEST_PATTERN_EN	R/W	0h	Test Pattern Enable. Force the output of K28.5 on TX_ENC for test purposes. 0h = Operation 1h = Forced K28.5 on transmit output
5	MASTER	R/W	0h	Master Mode. 0h = Slave Mode 1h = Master mode – Set to one for one side of a direct connection. When this bit is set, the control logic uses the CPSW_SGMII_MR_ADV_ABILITY_REG register to determine speed and duplexity instead of the CPSW_SGMII_MR_LP_ADV_ABILITY_REG register. Master mode allows a CPSGMII direct connection with auto-negotation or with a forced link.

Figure 3-4. CPSW_SGMII_CONTROL_REG_j Register

4. As shown in Figure 3-5, confirm the corresponding SGMII STATUS: SERDES PLL and LINK status.

Table 12-2178. CPSW_SGMII_STATUS_REG_j Register Field Descriptions

Bit	Field	Type	Reset	Description
31-6	RESERVED	R	X	
5	FIB_SIG_DETECT	R	0h	Fiber Signal Detect. This is the FIB_SIG_DETECT input pin.
4	LOCK	R	0h	Lock. This is the LOCK input pin. Indicates that the SERDES PLL is locked.
3	MR_PAGE_RX	R	0h	Next Page Received. This bit is set to one by the auto-negotation state machine when the next page has been received. This bit is cleared to 0h by a host write of 1h to the [3]MR_NP_LOADED bit in the CPSW_SGMII_CONTROL_REG register. This value is not valid until the lock status bit is ([4] LOCK) asserted.
2	MR_AN_COMPLETE	R	0h	Auto negotiation complete. This value is not valid until the lock status bit is asserted. 0h = Auto-negotation is not complete 1h = Auto-negotation is completed.
1	AN_ERROR	R	0h	Auto negotiation error. For SGMII mode, an auto-negotiation error occurs when halfduplex Gigabit is commanded. For FIBER mode, an auto-negotiation error occurs if both sides cannot be fullduplex. This value is not valid until the lock status bit is asserted. 0h = No auto-negotation error 1h = Auto-negotation error
0	LINK	R	0h	Link indicator. This value is not valid until the lock status bit is asserted. 0h = Link is not up. 1h = Link is up.

Figure 3-5. CPSW_SGMII_STATUS_REG_j Register

4 Summary

First of all, this paper introduces the background of MAC2MAC. Secondly, it describes various application scenarios to assist customer in designing their product. Then we introduce two MAC2MAC solutions: SGMII based solution and SGMII based solution. Finally, we show the steps how to implement those two solutions and give some debug suggestions.

In practical applications, we recommend using the MAC2MAC solution based on RGMII first when the IO port resources are abundant. This solution is simple to configure and easy to use due to the lack of SERDES.

5 References

1. Processor SDK Linus for AM335x – CPSW Kernel Drivers
2. Wikipedia – OSI Model
3. DRA829/TDA4VM/AM752x Technical Reference Manual
4. TDA4VM Jacinto™ Processors for ADAS and Autonomous Vehicles data sheet

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265