Ethernet

Notes for Ethernet subsystem on beaglebone black

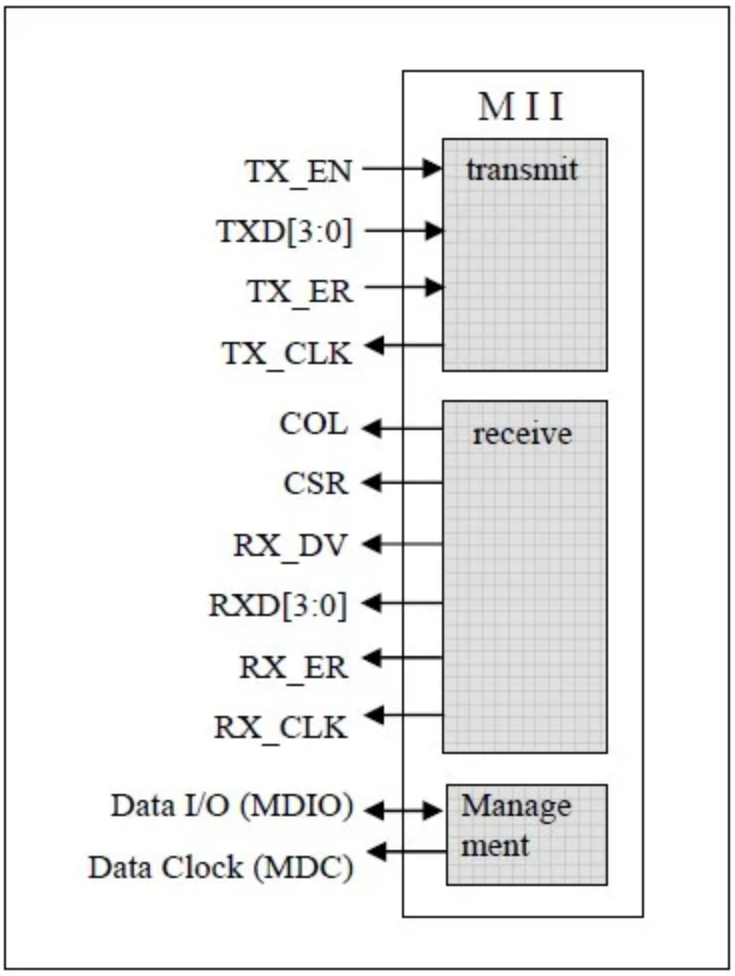
- References
  - https://www.ti.com/lit/ug/spruh73q/spruh73q.pdf?ts=1736331808765
  - https://www.kernel.org/doc/Documentation/devicetree/bindings/net/ti%2Ccpsw-switch.yaml
  - https://software-dl.ti.com/processor-sdk-linux/esd/docs/latest/linux/Foundational_Components/Kernel/Kernel_Drivers/Network/CPSW.html
  - https://embeddedhardwaredesign.com/media-independent-interface-mii-and-rmi-in-ethernet/
  - https://en.wikipedia.org/wiki/Management_Data_Input/Output
  - https://www.ti.com/lit/an/spraan7/spraan7.pdf
  - https://www.ti.com/lit/ds/symlink/am3358.pdf
  - https://ww1.microchip.com/downloads/aemDocuments/documents/UNG/ProductDocuments/DataSheets/LAN8710A-LAN8710Ai-Data-Sheet-DS00002164.pdf
  - https://cdn-shop.adafruit.com/datasheets/BBB_SRM.pdf

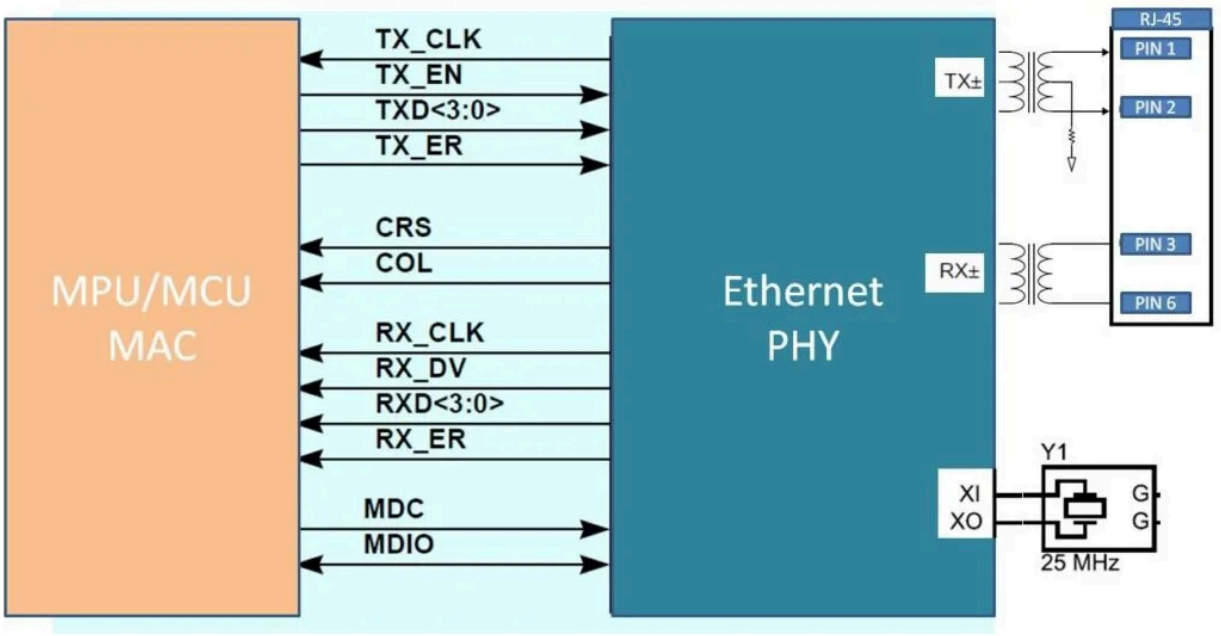
Introduction

- layer 2 3 port switch ethernet subsystem
- layer 2
  - network switch ports that operate in the data link layer
  - these switches forward date packets based on the MAC addresses specified
  - uses ARP to enable network communication
  - can communicate in __ mode
    - unicast
    - multicast
    - broadcast
  - do not have built in security features
  - can support all higher layers and protocols
    - IPV4

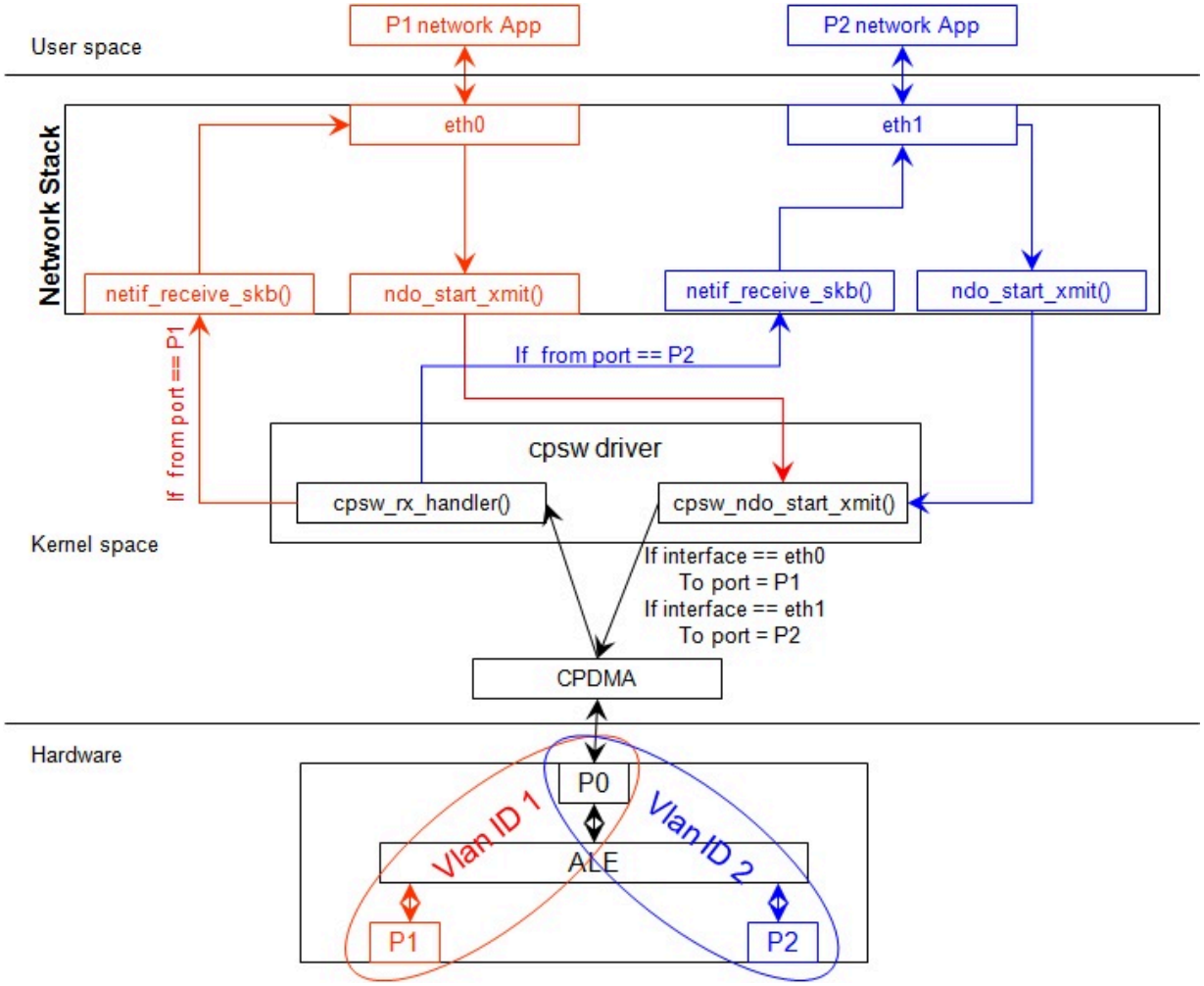
        - IPV5
        - 802.3x
- 3 port switch
    - provides ethernet packet communication and can be configured as an ethernet switch
    - 1 Host port ,2 External ports
    - acts as basic switch allowing data packets to be forwarded between connected devices based on MAC addresses
    - GMII
        - gigabit media independent interface
        - supporting data rates up to 1 Gbps
        - 24 pins
    - RGMII
        - reduced gigabit media independent interface
        - 24 pins for GMII to 12 pins for RGMII interface
    - RMII
        - reduced media independent interface
        - 12 pins for MII to 6 pins for RMII
    - MDIO
        - the management data input output for physical layer device (PHY)
        - PHY management interface, MDIO, used to read and write the control and status registers of the PHY in order to configure each PHY before operation
- media independent interface
    - serves as a standardized method for connecting Ethernet MAC (Media Access Control) devices to PHY (Physical Layer) devices. Its primary purpose is to facilitate communication between these two essential components of an Ethernet system.
    - The Media-Independent Interface (MII) is a parallel interface defined by the IEEE 802.3 standard. The management interface of the MII allows the configuration and control of multiple PHY devices, the gathering of status and error information, and the determination of the type and abilities of the attached PHY(s). It establishes the communication protocol and electrical signaling between the MAC(which manages data transmission) and the PHY (which handles the physical transmission medium, such as copper or fiber optics).
    - Media-Independent Interface (MII) enables interoperability between different manufacturers’ MAC and PHY devices. Standardization ensures that MACs from

one vendor can communicate seamlessly with PHYs from another, promoting compatibility and flexibility in network design

- Media-Independent Interface (MII) facilitates the transfer of Ethernet frames between the MAC and PHY layers. It provides the necessary data, control, and timing signals to ensure reliable and efficient transmission of data packets across the network
- By decoupling the MAC and PHY layers, Media-Independent Interface (MII) allows for flexibility in network deployment. Different PHY devices tailored to specific media types (e.g., copper, fiber) can be easily integrated with a variety of MAC implementations.
- MII supports various Ethernet speeds, including 10 Mbps, 100 Mbps (Fast Ethernet), and 1000 Mbps (Gigabit Ethernet), making it suitable for a wide range of network applications. This scalability ensures that MII interfaces can adapt to evolving network requirements and technologies.
- 



- Existing driver design for Dual MAC mode



- CPPI = Communications Port Programming Interface

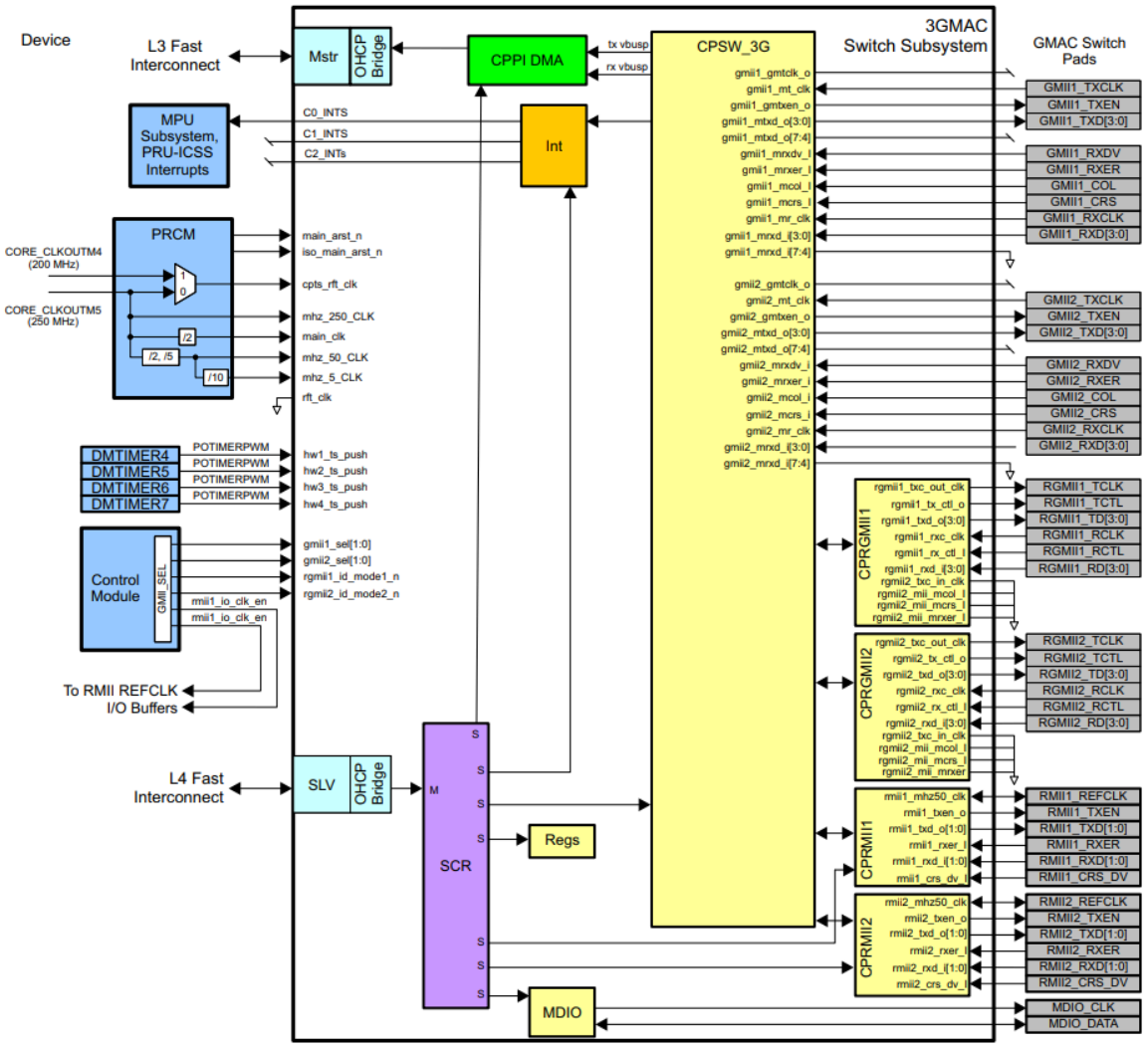
Features

- 2 ethernet 10/100/1000 ethernet ports with the 3 interfaces mentioned above
- logical FIFO based packet buffer structure
- CPPI 3.1 compliant DMA controllers
- Address lookup engine
    - 1024 addresses
    - MAC authentication
    - MAC address blocking
- maximum frame size of 2016 bytes
- 8k (2048 * 32) internal CPPI buffer descriptor memory
- MDIO module for PHY management
- programmable interrupt control
- doesn't support GMII, only supports
    - MII
    - RMII
    - RGMII

Integration

- single instance of three port gigabit ethernet switch subsystem
    - CPSW_3GSS_RG
- 2 external ethernet ports
    - port 1
    - port 2
- internal CPPI interface port
    - port 0
- 2 RGMII interface modules
- 2 RMII interface modules
- one MDIO interface module
- one interrupt controller module
- local CPPI memory of size 8k bytes

Figure 14-1. Ethernet Switch Integration



- SCR
  - Switched central resource
  - The SCR is an N:M crossbar that allows N masters to connect to M slaves. It adds no latency and allows seamless arbitration between the masters and slaves
- CPPI DMA
- Int
- MDIO
- CPSW_3G
- CPRGMII1
- CPRGMII2
- CPRMII1
- CPRMII2
- GMAC switch pads

Connectivity Attributes

- Power domain
  - peripheral Domain
- Clock domain
  - PD_PER_CPSW_125MHZ_GCLK
    - (Main)
  - PD_PER_CPSW_250MHZ_GCLK
    - (MHZ_250_CLK)
  - PD_PER_CPSW_50MHZ_GCLK
    - (MHZ_50_CLK)
  - PD_PER_CPSW_5MHZ_GCLK
    - (MHZ_5_CLK)
  - PD_PER_CPSW_CPTS_RFT_CLK
    - (CPTS_RFT_CLK)
- Reset Signals
  - CPSW_MAIN_ARST_N
  - CPSW_ISO_MAIN_ARST_N
- idle / wakeup signals
  - idle standby
- Interrupt requests
  - 4 interrupts
    - receive threshold interrupt
      - (3PGSWRXTHR0)
      - RX_THRESH
    - receive interrupt
      - (3PGSWRXINT0)
      - RX
    - transmit interrupt
      - (3PGSWTXINT0)
      - TX
    - Misc. interrupts
      - (3PGSWMISC0)
      - other interrupts
- DMA requests

    - none
- Physical address
    - L4 fast slave port
    - L3 fast initiator port

Clock and Rest Management

- ethernet switch controller operates in its own clock domain
- rft_clk
    - Gigabit GMII Tx reference clock
- main_clk
    - Logic/interface clock
- mhz250_clk
    - Gigabit RGMII Reference clock
- mhz50_clk
    - RMII and 100mbps RGMII reference clock
- mhz5_clk
    - 10 mbps RGMII reference clock
- cpts_rft_clk
    - IEEE 1588v2 clock
- gmii1_mr_clk
    - GMII Port 1 Receive clock
- gmii2_mr_clk
    - GMII Port 2 Receive clock
- gmii1_mt_clk
    - GMII port 1 transmit clock
- gmii2_mt_clk
    - GMII port 2 transmit clock
- rgmii1_rxc_clk
    - RGMII Port 1 Receive clock
- rgmii2_rxc_clk
    - RGMII Port 2 Receive clock
- rmii1_mhz_50_clk
    - RMII Port 1 Reference clock
- rmii2_mhz_50_clk

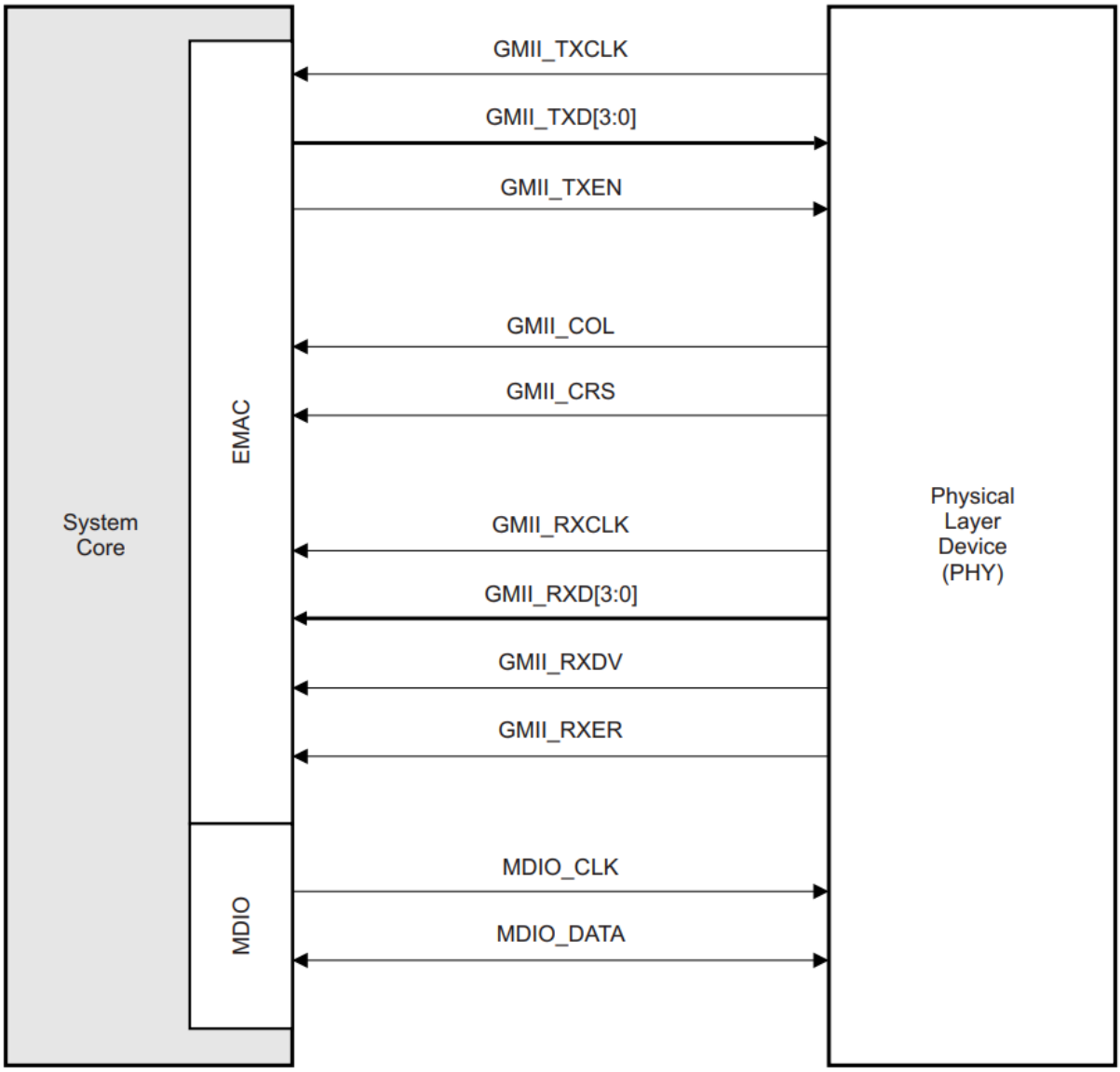
    - RMII Port 2 Reference clock

Pin List

- GMIIx_RXCLK
    - GMII/MII Receive clock
    - The receive clock is a continuous clock that provides the timing reference for receive operations.
    - The clock is generated by the PHY and is 2.5 MHz at 10 Mbps operation and 25 MHz at 100 Mbps operation
- GMIIx_RXD[3:0]
    - GMII/MII Receive data
    - The receive data pins are a collection of 4 data signals comprising 4 bits of data.
    - GMII_RXD[0] is the least-significant bit (LSB)
    - The signals are synchronized by GMII_RXCLK and valid only when GMII_RXDV is asserted
- GMIIx_RXDV
    - GMII/MII Receive data valid
    - The receive data valid signal indicates that the GMII_RXD pins are generating nibble data for use by the 3PSW
- GMIIx_RXER
    - GMII/MII Receive error
- GMIIx_COL
    - GMII/MII Collision detect
    - In half-duplex operation, the GMII_COL pin is asserted by the PHY when it detects a collision on the network
    - It remains asserted while the collision condition persists
- GMIIx_CRS
    - GMII/MII Carrier sense
    - In half-duplex operation, the GMII_CRS pin is asserted by the PHY when the network is not idle in either transmit or receive
    - The pin is deasserted when both transmit and receive are idle
- GMIIx_TXCLK
    - GMII/MII Transmit clock
    - The transmit clock is a continuous clock that provides the timing reference for transmit operations

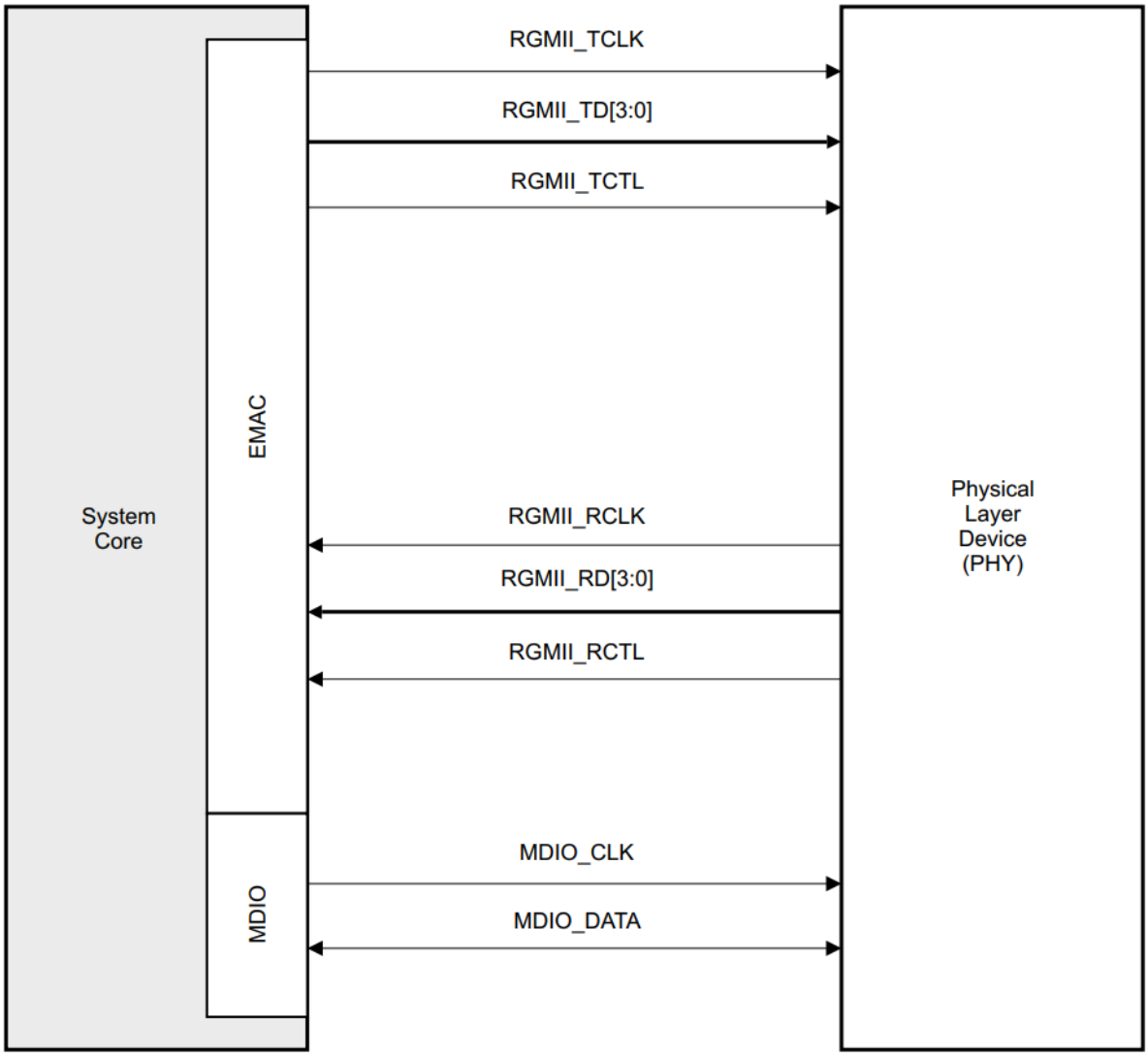
    - The clock is generated by the PHY and is 2.5 MHz at 10 Mbps operation and 25 MHz at 100 Mbps operation
- GMIIx_TXD[3:0]
    - GMII/MII Transmit data
    - The transmit data pins are a collection of 4 data signals GMII_TXD[3:0] comprising 4 bits of data
    - GMII_TXD[0] is the least-significant bit (LSB)
    - The signals are synchronized by GMII_TXCLK and valid only when GMII_TXEN is asserted
- GMIIx_TXEN
    - GMII/MII Transmit enable
    - The transmit enable signal indicates that the GMII_TXD[3:0] pins are generating 4-bit data for use by the PHY

Figure 14-3. MII Interface Connections



- RGMIIx_RCLK
  - RGMII Receive clock
  - The receive clock is a continuous clock that provides the timing reference for receive operations.
  - The clock is generated by the PHY and is 2.5 MHz at 10 Mbps operation and 25 MHz at 100 Mbps operation,125 MHz at 1000Mbps of operation
- RGMIIx_RCTL
  - RGMII Receive control
  - The receive data valid/control signal indicates that the RGMII_RD pins are nibble data for use by the 3PSW.
- RGMIIx_RD[3:0]
  - RGMII Receive data
  - The receive data pins are a collection of 4 bits of data.
  - RGMII_RD is the least significant bit (LSB).
  - The signals are valid only when RGMII_RCTL is asserted
- RGMIIx_TCLK
  - RGMII Transmit clock
  - The transmit reference clock will be 125Mhz, 25Mhz, or 2.5Mhz depending on speed of operation
- RGMIIx_TCTL
  - RGMII Transmit control/enable
  - The transmit enable signal indicates that the RGMII_TD pins are generating data for use by the PHY
- RGMIIx_TD[3:0]
  - RGMII Transmit data
  - The transmit data pins are a collection of 4 bits of data.
  - RGMII_RD0 is the least significant bit (LSB).
  - The signals are valid only when RGMII_TCTL is asserted

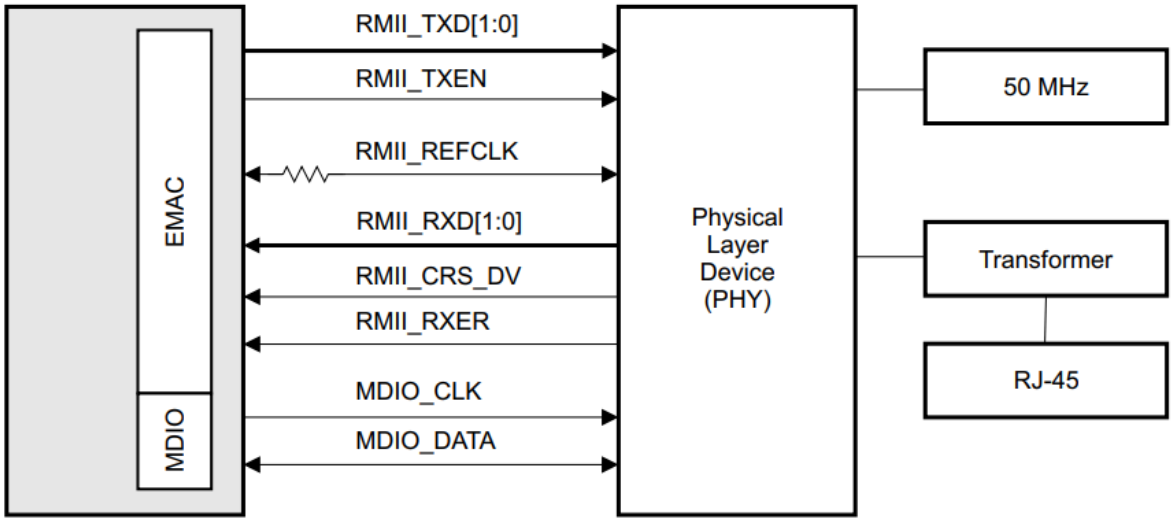
Figure 14-5. RGMII Interface Connections



- RMIIx_RXD[1:0]
  - RMII Receive data
  - The receive data pins are a collection of 2 bits of data
  - RMII_RXD0 is the least-significant bit (LSB)
  - The signals are synchronized by RMII_REFCLK and valid only when RM_CRS_DV is asserted and RMII_RXER is deasserted
- RMIIx_RXER
  - RMII Receiver error
  - The receive error signal is asserted to indicate that an error was detected in the received frame.
- RMIIx_CRS_DV
  - RMII Carrier sense / Data valid
  - Carrier sense/receive data valid.
  - Multiplexed signal between carrier sense and receive data valid

- RMIIx_TXEN
  - RMII Transmit enable
  - Transmit enable. The transmit enable signal indicates that the RMII_TXD pins are generating data for use by the PHY
- RMIIx_REFCLK
  - RMII Reference clock
  - The reference clock is used to synchronize all RMII signals. RMII_REFCLK must be continuous and fixed at 50 MHz.
- RMIIx_TXD[1:0]
  - RMII Transmit data
  - The transmit data pins are a collection of 2 bits of data
  - RMII_TXD0 is the least-significant bit (LSB)
  - The signals are synchronized by RMII_REFCLK and valid only when RMII_TXEN is asserted

Figure 14-4. RMII Interface Connections



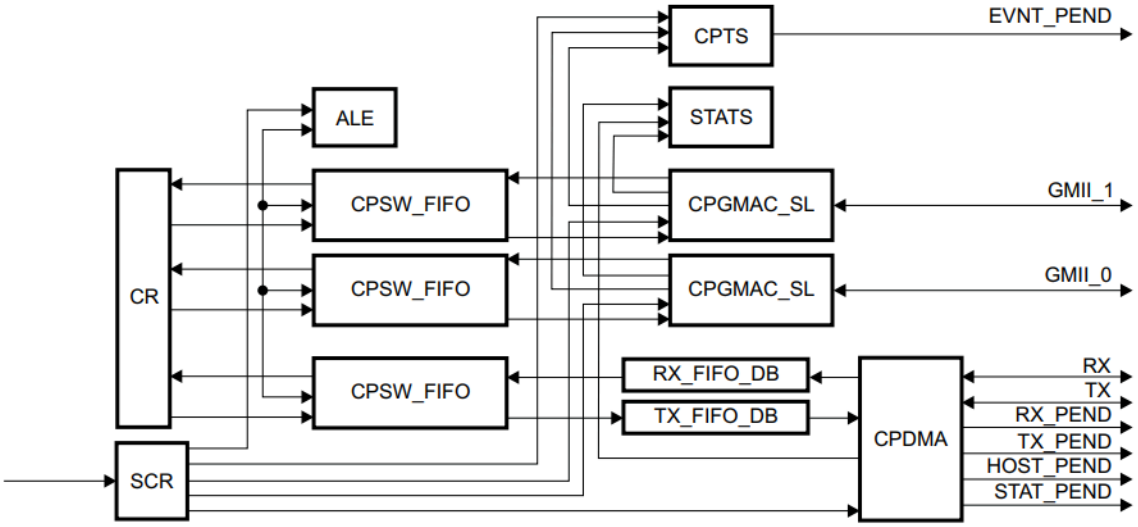
- MDIO_CLK
  - MDIO Serial clock
  - The MDIO data clock is sourced by the MDIO module on the system. It is used to synchronize MDIO data access operations done on the MDIO_DATA pin
- MDIO_DATA
  - MDIO Serial data
  - The MDIO_DATA pin drives PHY management data into and out of the PHY
  - by way of an access frame consisting of
    - start of frame
    - read/write indication

        - PHY address
        - register address
        - data bit cycles
    - The MDIO_DATA pin acts as an output for all but the data bit cycles at which time it is an input for read operations

CPSW_3G Subsystem

- The CPSW_3G GMII interfaces are compliant to the IEEE Std 802.3 Specification
- The CPSW_3G contains
    - two CPGMAC_SL interfaces (ports 1 and 2),
    - one CPPI 3.0 interface Host Port (port 0),
    - Common Platform Time Sync (CPTS),
    - ALE Engine
    - CPDMA.

Figure 14-6. CPSW_3G Block Diagram



Interrupt pacing module

- receive and transmit pulse interrupts can be paced
- 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
- Interrupt pacing increases the CPU cache hit ratio by minimizing the number of times that large interrupt service routines are moved to and from the CPU instruction cache

Reset Isolation

- Reset isolation for the Ethernet switch on Device is that the switch function of the ethernet IP remains active even in case of all device resets except for POR pin reset and ICEPICK COLD reset
- Packet traffic to/from the 3PSW host will be flushed/dropped, but the ethernet switch will remain operational for all traffic between external devices on the switch even though the device is under-going a device reset.

Modes of Operation

- The device has two modes of operation concerning the reset of the 3PSW Ethernet switch
- The mode is controlled by the ISO_CONTROL bit in RESET_ISO register of the device control module
- This bit should default to '0'. Writes to the ISO_CONTROL bit must be supervisor mode writes.
- Mode 1: ISO_CONTROL=0 (reset isolation disabled)
  - This mode is selected when ISO_CONTROL bit of control module is = 0. This should be the default state of the bit after control module reset.
  - Upon any device level resets, the entire CPSW_3GSS_R IP, L3/L4, control module (including all pin mux control and the ISO_CONTROL bit) is immediately reset
- Mode 2: ISO_CONTROL=1 (reset isolation enabled)
  - This mode is selected when ISO_CONTROL bit of control module is = 1
  - Upon any device reset source other than POR pin or ICEPICK cold (so this includes SW global cold, any watchdog reset, warm RESETn pin, ICEPICK warm, SW global warm), the following should be true:
    - The CPSW_3GSS_R is put into ‘isolate’ mode and non-switch related portions of the IP are reset
    - The 50-MHz and 125-MHz reference clocks to the 3PSW Ethernet Subsystem remains active throughout the entire reset condition
    - The control for pin multiplexing for all of the signals should maintain their current configuration throughout the entire reset condition
    - The reset isolated logic inside 3PSW Ethernet Subsystem IP which maintains the switch functionality

- Upon any cold reset sources, the entire 3PSW Ethernet Subsystem, control module (including all pin mux control and the ISO_CONTROL bit itself) is reset.

Interrupts

- the CPSW generates 4 interrupt events
- Receive packet completion pulse interrupt
  - RX_PULSE
  - The RX_PULSE interrupt is a paced pulse interrupt selected from the 3PSW RX_PEND [7:0] interrupts
  - The receive DMA controller has eight channels with each channel having a corresponding (RX_PEND[7:0]).
  - The following steps will enable the receive packet completion interrupt
    - Enable the required channel interrupts of the DMA engine by setting 1 to the appropriate bit in the RX_INTMASK_SET register
    - The receive completion interrupt(s) to be routed to RX_PULSE is selected by setting one or more bits in the receive interrupt enable register Cn_RX_EN
    - The masked interrupt status can be read in the Receive Interrupt Masked Interrupt Status (Cn_RX_STAT) register
  - When the 3PSW completes a packet reception, the subsystem issues an interrupt to the CPU by writing the packet's last buffer descriptor address to the appropriate channel queue's receive completion pointer located in the state RAM block.
  - The interrupt is generated by the write when enabled by the interrupt mask, regardless of the value written
  - Upon interrupt reception, the CPU processes one or more packets from the buffer chain
  - acknowledges one or more interrupt(s) by writing the address of the last buffer descriptor processed to the queue's associated receive completion pointer (RXn_CP) in the receive DMA state RAM.
  - Upon reception of an interrupt, software should perform the following:
    - Read the RX_STAT register to determine which channel(s) caused the interrupt
    - Write the 3PSW completion pointer(s) (RXn_CP)
      - The data written by the host (buffer descriptor address of the last processed buffer) is compared to the data in the register written by the

subsystem (address of last buffer descriptor used by the subsystem). If the two values are not equal (which means that the 3PSW has received more packets than the CPU has processed), the receive packet completion interrupt signal remains asserted
        - If the two values are equal (which means that the host has processed all packets that the system has received), the pending interrupt is de-asserted.
        - The value that the 3PSW is expecting is found by reading the receive channeln completion pointer register (RXn_CP).
    - Write the value 1h to the CPDMA_EOI_VECTOR register
- To disable the interrupt
    - The eight channel interrupts may be individually disabled by writing to 1 the appropriate bit in the RX_INTMASK_CLEAR
    - The receive completion pulse interrupt could be disabled by clearing to 0 all the bits of the RX_EN

- Transmit packet completion pulse interrupt
    - TX_PULSE
    - The TX_PULSE interrupt is a paced pulse interrupt selected from the 3PSW TX_PEND [7:0] interrupts.
    - The transmit DMA controller has eight channels with each channel having a corresponding (TX_PEND[7:0]).
    - To enable the transmit packet completion interrupt
        - Enable the required channel interrupts of the DMA engine by setting 1 to the appropriate bit in the TX_INTMASK_SET register
        - The transmit completion interrupt(s) to be routed to TX_PULSE is selected by setting one or more bits in the transmit interrupt enable register Cn_TX_EN
        - The masked interrupt status can be read in the Transmit Interrupt Masked Interrupt Status (Cn_TX_STAT) register.
    - When the 3PSW completes the transmission of a packet, the 3PSW subsystem issues an interrupt to the CPU by writing the packet’s last buffer descriptor address to the appropriate channel queue’s transmit completion pointer located in the state RAM block
    - The interrupt is generated by the write when enabled by the interrupt mask, regardless of the value written
    - Upon receiving an interrupt, software should perform the following

        - Read the TX_STAT register to determine which channel(s) caused the interrupt
        - Process received packets for the interrupting channel
        - Write the 3PSW completion pointer(s) (TXn_CP)
            - The data written by the host (buffer descriptor address of the last processed buffer) is compared to the data in the register written by the 3PSW (address of last buffer descriptor used by the 3PSW).
            - If the two values are not equal (which means that the 3PSW has transmitted more packets than the CPU has processed), the transmit packet completion interrupt signal remains asserted.
            - If the two values are equal (which means that the host has processed all packets that the subsystem has transferred), the pending interrupt is cleared.
            - The value that the 3PSW is expecting is found by reading the transmit channeln completion pointer register (TXn_CP).
            - Write the 2h to the CPDMA_EOI_VECTOR register
    - To disable the interrupt
        - The eight channel interrupts may be individually disabled by writing to 1 the appropriate bit in the TX_INTMASK_CLEAR
        - The receive completion pulse interrupt could be disabled by clearing to 0 all the bits of the TX_EN. The software could still poll for the TX_INTSTAT_RAW and TX_INTSTAT_MASKED registers if the corresponding interrupts are enabled
- Receive Threshold Pulse Interrupt
- Miscellaneous Pulse Interrupt
- EVNT_PEND (CPTS_PEND) Interrupt
- Statistics Interrupt
- Host Error Interrupt
- MDIO Interrupts

Media independent interface

- The CPSW_3G contains two CPGMAC_SL submodules.
- Each CPGMAC_SL has a single GMII interface
- The CPGMAC_SL submodules are ports 1 and 2

- The following sections cover operation of the Media Independent Interface in 10/100/1000 Mbps modes. An IEEE 802.3 compliant Ethernet MAC controls the interface.
- data reception
  - receive control
    - Data received from the PHY is interpreted and output
    - Interpretation involves detection and removal of the preamble and start of frame delimiter, extraction of the address and frame length, data handling, error checking and reporting, cyclic redundancy checking (CRC), and statistics control signal generation.
- data transmission
  - The Gigabit Ethernet Mac Sliver (GMII) passes data to the PHY when enabled
  - Data is synchronized to the transmit clock rate.
  - The smallest frame that can be sent is two bytes of data with four bytes of CRC
  - transmit control
    - A jam sequence is output if a collision is detected on a transmit packet.
    - If the collision was late (after the first 64 bytes have been transmitted) the collision is ignored
    - If the collision is not late, the controller will back off before retrying the frame transmission
    - When operating in full duplex mode the carrier sense (CRS) and collision sensing modes are disabled.
  - CRC insertion
    - The MAC generates and appends a 32-bit Ethernet CRC onto the transmitted data if the transmit packet header pass_crc bit is zero
    - For the CPMAC_SL generated CRC case, a CRC at the end of the input packet data is not allowed.
    - If a CRC is not needed, set the pass_crc bit to zero and adjust the packet length accordingly
    - If the header word pass_crc bit is set, then the last four bytes of the TX data are transmitted as the frame CRC
    - The four CRC data bytes should be the last four bytes of the frame and should be included in the packet byte count value.
    - The MAC performs no error checking on the outgoing CRC when the pass_crc bit is set
- Frame Classification

- Received frames are proper (good) frames if they are between 64 and rx_maxlen in length (inclusive) and contain no errors (code/align/CRC).
- Received frames are long frames if their frame count exceeds the value in the rx_maxlen register
- The rx_maxlen register reset (default) value is 1518 (dec).
- Long received frames are either oversized or jabber frames.
- Long frames with no errors are oversized frames. Long frames with CRC, code, or alignment errors are jabber frames
- Received frames are short frames if their frame count is less than 64 bytes.
- Short frames that contain no errors are undersized frames. Short frames with CRC, code, or alignment errors are fragment frames.

CPDMA RX and TX interfaces

- The CPDMA submodule is a CPPI 3.0 compliant packet DMA transfer controller
- The CPPI 3.0 interface is port 0
- After reset, initialization, and configuration the host may initiate transmit operations
- Transmit operations are initiated by host writes to the appropriate transmit channel head descriptor pointer contained in the STATERAM block.
- The transmit DMA controller then fetches the first packet in the packet chain from memory in accordance with CPPI 3.0 protocol.
- The DMA controller writes the packet into the external transmit FIFO in 64-byte bursts
- Receive operations are initiated by host writes to the appropriate receive channel head descriptor pointer after host initialization and configuration
- The receive DMA controller writes the receive packet data to external memory in accordance with CPPI 3.0 protocol

CPPI Buffer Descriptors

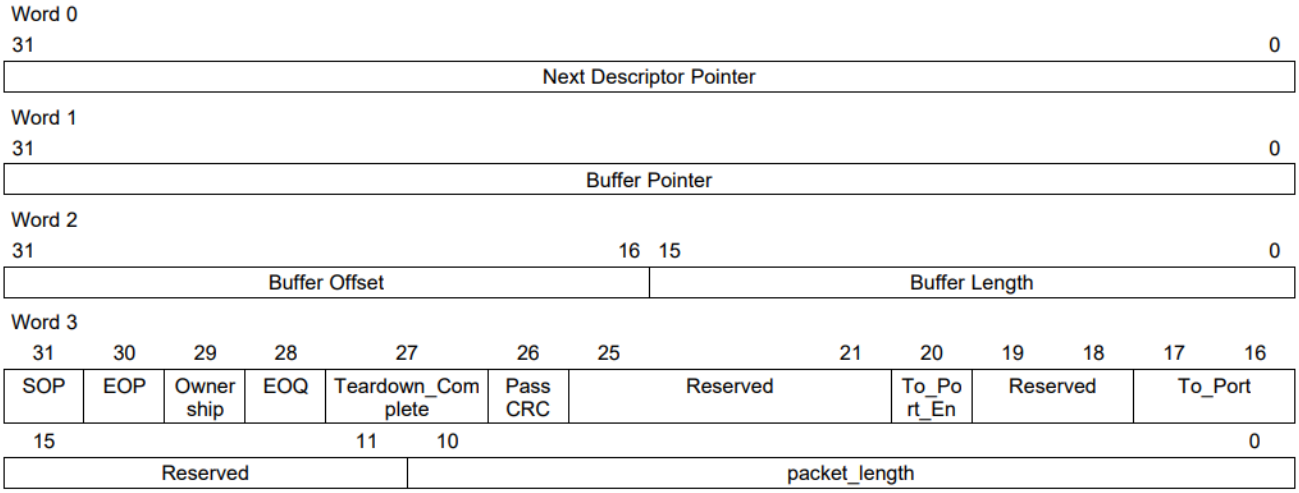
- The buffer descriptor is a central part of the 3PSW Ethernet Subsystem and is how the application software describes Ethernet packets to be sent and empty buffers to be filled with incoming packet data
- Host Software sends and receives network frames via the CPPI 3.0 compliant host interface
- The host interface includes module registers and host memory data structures
- The host memory data structures are buffer descriptors and data buffers

- Buffer descriptors are data structures that contain information about a single data buffer
- Buffer descriptors may be linked together to describe frames or queues of frames for transmission of data and free buffer queues available for received data
- The 8k bytes of Ethernet Subsystem CPPI RAM begin at address 0x4a102000 and end at 0x4a103FFF from the 3PSW perspective
- The buffer descriptors programmed to access the CPPI RAM memory should use the address range from 0x4a102000.

TX Buffer Descriptors

A TX buffer descriptor is a contiguous block of four 32-bit data words aligned on a 32-bit word boundary.

Figure 14-7. Tx Buffer Descriptor Format



- Next descriptor pointer
  - The next descriptor pointer points to the 32-bit word aligned memory address of the next buffer descriptor in the transmit queue.
  - This pointer is used to create a linked list of buffer descriptors
  - If the value of this pointer is zero, then the current buffer is the last buffer in the queue
  - The software application must set this value prior to adding the descriptor to the active transmit list
  - This pointer is not altered by the EMAC.The value of pNext should never be altered once the descriptor is in an active transmit queue, unless its current value is NULL
  - If the pNext pointer is initially NULL, and more packets need to be queued for transmit, the software application may alter this pointer to point to a newly

appended descriptor
    - The EMAC will use the new pointer value and proceed to the next descriptor unless the pNext value has already been read
    - In this latter case, the transmitter will halt on the transmit channel in question, and the software application may restart it at that time
    - The software can detect this case by checking for an end of queue (EOQ) condition flag on the updated packet descriptor when it is returned by the EMAC
- Buffer pointer
    - The byte aligned memory address of the buffer associated with the buffer descriptor
    - The host sets the buffer_pointer.
    - The software application must set this value prior to adding the descriptor to the active transmit list.
    - This pointer is not altered by the EMAC
- Buffer Offset
    - Indicates how many unused bytes are at the start of the buffer
    - A value of 0x0000 indicates that no unused bytes are at the start of the buffer and that valid data begins on the first byte of the buffer
    - A value of 0x000F (decimal 15) indicates that the first 15 bytes of the buffer are to be ignored by the port and that valid buffer data starts on byte 16 of the buffer
    - valid only on start of packet
- Buffer Length
    - Indicates how many valid data bytes are in the buffer
    - Unused or protocol specific bytes at the beginning of the buffer are not counted in the Buffer Length field
    - The buffer_length must be greater than zero
- SOP
    - Start of packet
    - When set, this flag indicates that the descriptor points to a packet buffer that is the start of a new packet
    - In the case of a single fragment packet, both the SOP and end of packet (EOP) flags are set
    - Otherwise, the descriptor pointing to the last packet buffer for the packet sets the EOP flag
    - 0 - Not start of packet buffer
    - 1 - Start of packet buffer

- EOP
  - End of packet
  - When set, this flag indicates that the descriptor points to a packet buffer that is last for a given packet.
  - 0 - Not end of packet buffer
  - 1 - End of packet buffer.
- Ownership
  - When set this flag indicates that all the descriptors for the given packet (from SOP to EOP) are currently owned by the EMAC
  - This flag is set by the software application on the SOP packet descriptor before adding the descriptor to the transmit descriptor queue
  - For a single fragment packet, the SOP, EOP, and OWNER flags are all set
  - The OWNER flag is cleared by the EMAC once it is finished with all the descriptors for the given packet.
  - Note that this flag is valid on SOP descriptors only
- EOQ
  - When set, this flag indicates that the descriptor in question was the last descriptor in the transmit queue for a given transmit channel, and that the transmitter has halted
  - This flag is initially cleared by the software application prior to adding the descriptor to the transmit queue
  - This bit is set by the EMAC when the EMAC identifies that a descriptor is the last for a given packet (the EOP flag is set) , and there are no more descriptors in the transmit list (next descriptor pointer is NULL).
  - The software application can use this bit to detect when the EMAC transmitter for the corresponding channel has halted.
  - This is useful when the application appends additional packet descriptors to a transmit queue list that is already owned by the EMAC
  - Note that this flag is valid on EOP descriptors only
  - 0 - The Tx queue has more packets to transfer.
  - 1 - The Descriptor buffer is the last buffer in the last packet in the queue
- teardown complete
  - This flag is used when a transmit queue is being torn down, or aborted, instead of allowing it to be transmitted
  - This would happen under device driver reset or shutdown conditions

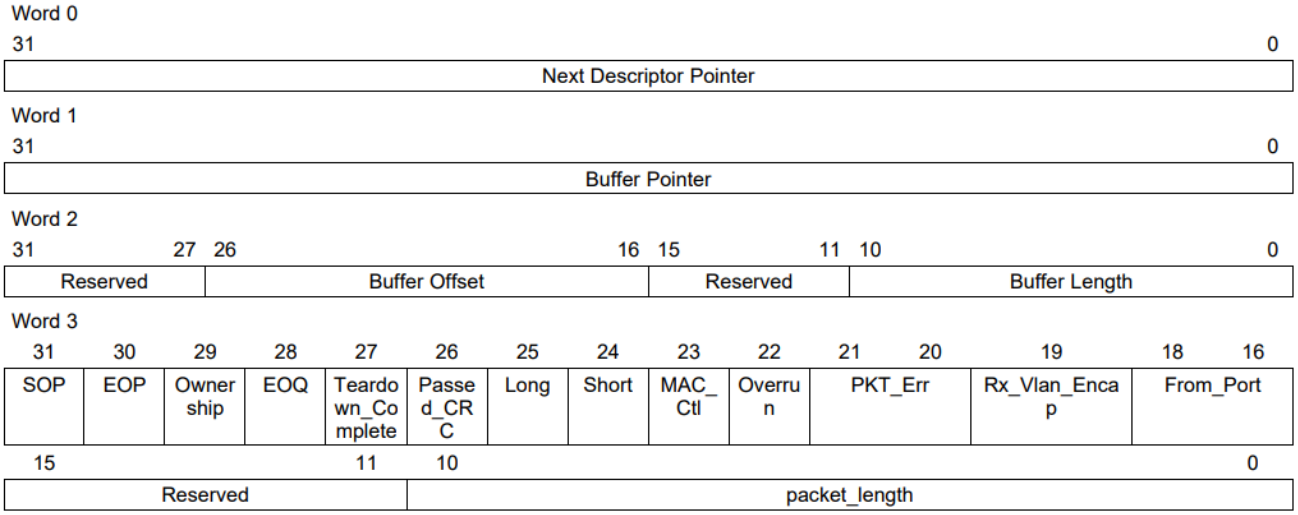
    - The EMAC sets this bit in the SOP descriptor of each packet as it is aborted from transmission
    - Note that this flag is valid on SOP descriptors only
    - Also note that only the first packet in an unsent list has the TDOWNCMPLT flag set. Subsequent descriptors are not processed by the EMAC
    - 0 - The port has not completed the teardown process.
    - 1 - The port has completed the commanded teardown process
- pass crc
    - This flag is set by the software application in the SOP packet descriptor before it adds the descriptor to the transmit queue
    - Setting this bit indicates to the EMAC that the 4 byte Ethernet CRC is already present in the packet data, and that the EMAC should not generate its own version of the CRC
    - When the CRC flag is cleared, the EMAC generates and appends the 4-byte CRC
        - The buffer length and packet length fields do not include the CRC bytes
    - When the CRC flag is set, the 4-byte CRC is supplied by the software application and is already appended to the end of the packet data.
        - The buffer length and packet length fields include the CRC bytes
    - Note that this flag is valid on SOP descriptors only
    - 0 – The CRC is not included with the packet data and packet length
    - 1 – The CRC is included with the packet data and packet length.
- to port
    - Port number to send the directed packet to
    - This field is set by the host.
    - This field is valid on SOP
    - Directed packets go to the directed port
    - ALE lookup is performed to determine untagged egress in VLAN_AWARE mode
    - 1 – Send the packet to port 1 if to_port_en is asserted.
    - 2 – Send the packet to port 2 if to_port_en is asserted.
- to port enable
    - Indicates when set that the packet is a directed packet to be sent to the to_port field port number
    - This field is set by the host
    - The packet is sent to one port only (index not mask).

    - This bit is valid on SOP
    - 0 – not a directed packet
    - 1 – directed packet
- packet length
    - Specifies the number of bytes in the entire packet
    - Offset bytes are not included
    - The sum of the buffer_length fields should equal the packet_length
    - Valid only on SOP
    - The packet length must be greater than zero
    - The packet data will be truncated to the packet length if the packet length is shorter than the sum of the packet buffer descriptor buffer lengths
    - A host error occurs if the packet length is greater than the sum of the packet buffer descriptor buffer lengths

RX Buffer Descriptors

- An RX buffer descriptor is a contiguous block of four 32-bit data words aligned on a 32-bit word boundary.

Figure 14-8. Rx Buffer Descriptor Format



- next descriptor pointer
    - The 32-bit word aligned memory address of the next buffer descriptor in the RX queue
    - This is the mechanism used to reference the next buffer descriptor from the current buffer descriptor
    - If the value of this pointer is zero then the current buffer is the last buffer in the queue

    - The host sets the next_descriptor_pointer
- buffer pointer
    - The byte aligned memory address of the buffer associated with the buffer descriptor.
    - The host sets the buffer_pointer
- buffer offset
    - indicates how many unused bytes are at the start of the buffer
    - A value of 0x0000 indicates that there are no unused bytes at the start of the buffer and that valid data begins on the first byte of the buffer
    - A value of 0x000F (decimal 15) indicates that the first 15 bytes of the buffer are to be ignored by the port and that valid buffer data starts on byte 16 of the buffer
    - The port writes the buffer_offset with the value from the rx_buffer_offset register value
    - The host initializes the buffer_offset to zero for free buffers
    - The buffer_length must be greater than the rx_buffer_offset value
    - The buffer offset is valid only on sop
- buffer length
    - Indicates how many valid data bytes are in the buffer
    - Unused or protocol specific bytes at the beginning of the buffer are not counted in the Buffer Length field
    - The host initializes the buffer_length, but the port may overwrite the host initiated value with the actual buffer length value on SOP and/or EOP buffer descriptors.
    - SOP buffer length values will be overwritten if the packet size is less than the size of the buffer or if the offset is nonzero
    - . EOP buffer length values will be overwritten if the entire buffer is not filled up with data.
    - The buffer_length must be greater than zero.
- SOP
    - start of packet
    - When set, this flag indicates that the descriptor points to a packet buffer that is the start of a new packet
    - In the case of a single fragment packet, both the SOP and end of packet (EOP) flags are set
    - Otherwise, the descriptor pointing to the last packet buffer for the packet has the EOP flag set

    - This flag is initially cleared by the software application before adding the descriptor to the receive queue
    - This bit is set by the EMAC on SOP descriptors
- EOP
    - End of packet
    - When set, this flag indicates that the descriptor points to a packet buffer that is last for a given packet
    - In the case of a single fragment packet, both the start of packet (SOP) and EOP flags are set.
    - Otherwise, the descriptor pointing to the last packet buffer for the packet has the EOP flag set.
    - This flag is initially cleared by the software application before adding the descriptor to the receive queue.
    - This bit is set by the EMAC on EOP descriptors.
- Ownership
    - When set, this flag indicates that the descriptor is currently owned by the EMAC
    - This flag is set by the software application before adding the descriptor to the receive descriptor queue
    - This flag is cleared by the EMAC once it is finished with a given set of descriptors, associated with a received packet
    - The flag is updated by the EMAC on SOP descriptor only
    - So when the application identifies that the OWNER flag is cleared on an SOP descriptor, it may assume that all descriptors up to and including the first with the EOP flag set have been released by the EMAC
    - Note that in the case of single buffer packets, the same descriptor will have both the SOP and EOP flags set
- EOQ
    - end of queue
    - When set, this flag indicates that the descriptor in question was the last descriptor in the receive queue for a given receive channel, and that the corresponding receiver channel has halted
    - This flag is initially cleared by the software application prior to adding the descriptor to the receive queue
    - This bit is set by the EMAC when the EMAC identifies that a descriptor is the last for a given packet received and there are no more descriptors in the receive list
    - also sets the EOP flag

    - The software application can use this bit to detect when the EMAC receiver for the corresponding channel has halted
    - This is useful when the application appends additional free buffer descriptors to an active receive queue.
    - Note that this flag is valid on EOP descriptors only.
- Teardown complete
    - This flag is used when a receive queue is being torn down, or aborted, instead of being filled with received data
    - This would happen under device driver reset or shutdown conditions.
    - The EMAC sets this bit in the descriptor of the first free buffer when the tear down occurs.
    - No additional queue processing is performed.
- pass crc
    - This flag is set by the EMAC in the SOP buffer descriptor if the received packet includes the 4-byte CRC
    - This flag should be cleared by the software application before submitting the descriptor to the receive queue
- long
    - This flag is set by the EMAC in the SOP buffer descriptor, if the received packet is a jabber frame and was not discarded because the RX_CEF_EN bit was set in the MacControl
    - Jabber frames are frames that exceed the RXMAXLEN in length, and have CRC, code, or alignment errors.
- short
    - This flag is set by the EMAC in the SOP buffer descriptor, if the received packet is only a packet fragment and was not discarded because the RX_CSF_EN bit was set in the MacControl
- control flag
    - This flag is set by the EMAC in the SOP buffer descriptor, if the received packet is an EMAC control frame and was not discarded because the RX_CMF_EN bit was set in the MacControl
- overrun flag
    - This flag is set by the EMAC in the SOP buffer descriptor, if the received packet was aborted due to a receive overrun.
- pkt error flag
    - Packet Contained Error on Ingress

    - 00 – no error
    - 01 – CRC error on ingress
    - 10 – Code error on ingress
    - 11 – Align error on ingress
- rx vlan encap
    - VLAN Encapsulated Packet
    - Indicates when set that the packet data contains a 32-bit VLAN header word that is included in the packet byte count
    - This field is set by the port to be the value of the CPSW control register rx_vlan_encap bit
- from port
    - Indicates the port number that the packet was received on (ingress to the switch).
- packet length
    - Specifies the number of bytes in the entire packet.
    - The packet length is reduced to 12-bits.
    - Offset bytes are not included.
    - The sum of the buffer_length fields should equal the packet_length.
    - Valid only on SOP.

Receive DMA interface

- The Receive DMA is an eight channel CPPI 3.0 compliant interface.
- Each channel has a single queue for frame reception
- host configuration
    - To configure the Rx DMA for operation the host must perform the following:
        - Initialize the receive addresses
        - Initialize the Rx_HDP Registers to zero
        - Enable the desired receive interrupts in the IntMask register
        - Write the rx_buffer_offset register value
        - Setup the receive channel(s) buffer descriptors in host memory as required by CPPI 3.0
        - Enable the RX DMA controller by setting the rx_en bit in the Rx_Control register.

Transmit DMA interface

- The Transmit DMA is an eight channel CPPI 3.0 compliant interface
- Priority between the eight queues may be either fixed or round robin as selected by tx_ptype in the DMAControl register
- If the priority type is fixed, then channel 7 has the highest priority and channel 0 has the lowest priority
- Round robin priority proceeds from channel 0 to channel 7.
- Packet Data transfers occur on the TX_VBUSP interface in 64- byte maximum burst transfers
- host configuration
    - To configure the TX DMA for operation the host must do the following:
    - Initialize the Tx_HDP registers to a zero value
    - Enable the desired transmit interrupts in the IntMask register
    - Setup the transmit channel(s) buffer descriptors in host memory as defined in CPPI 3.0
    - Configure and enable the transmit operation as desired in the TxControl register
    - Write the appropriate Tx_HDP registers with the appropriate values to start transmit operations

VLAN Aware mode

- The CPSW_3G is in VLAN aware mode when the CPSW Control register vlan_aware bit is set
- In VLAN aware mode ports 0 receive packets (out of the CPSW_3G) may or may not be VLAN encapsulated
- depending on the CPSW Control register rx_vlan_encap bit
- Port 0 receive packet data is never modified
- VLAN is not removed regardless of the force untagged egress bit for Port 0
- VLAN encapsulated receive packets have a 32-bit VLAN header encapsulation word added to the packet data
- VLAN encapsulated packets are specified by a set rx_vlan_encap bit in the packet buffer descriptor
- Port 0 transmit packets are never VLAN encapsulated (encapsulation is not allowed).
- In VLAN aware mode, transmitted packet data is changed depending on the packet type (pkt_type), packet priority (pkt_pri), and VLAN information as shown in the below tables

Figure 14-9. VLAN Header Encapsulation Word

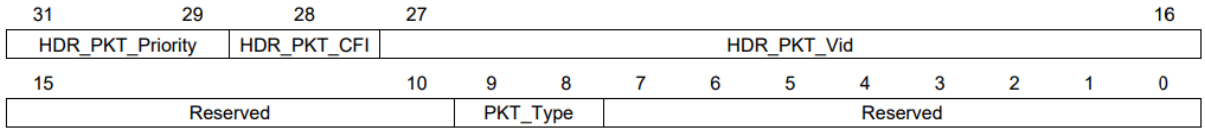


Table 14-8. VLAN Header Encapsulation Word Field Descriptions

Field	Description
HDR_PKT_Priority	Header Packet VLAN priority (Highest priority: 7)
HDR_PKT_CFI	Header Packet VLAN CFI bit.
HDR_PKT_Vid	Header Packet VLAN ID
PKT_Type	Packet Type. Indicates whether the packet is VLAN-tagged, priority-tagged, or non-tagged. 00: VLAN-tagged packet 01: Reserved 10: Priority-tagged packet 11: Non-tagged packet

- The CPSW_3G is in VLAN unaware mode when the CPSW Control register vlan_aware bit is cleared

ALE

- address lookup engine
- The address lookup engine (ALE) processes all received packets to determine which port(s) if any that the packet should the forwarded to
- The ALE uses the incoming packet received port number, destination address, source address, length/type, and VLAN information to determine how the packet should be forwarded
- The ALE outputs the port mask to the switch fabric that indicates the port(s) the packet should be forwarded to
- The ALE is enabled when the ale_enable bit in the ALE_Control register is set. All packets are dropped when the ale_enable bit is cleared to zero.
- In normal operation, the CPGMAC_SL modules are configured to issue an abort, instead of an end of packet, at the end of a packet that contains an error (runt, frag, oversize, jabber, crc, alignment, code etc.) or at the end of a mac control packet
- However, when the CPGMAC_SL configuration bit(s) cef, csf, or cmf are set, error frames, short frames or mac control frames have a normal end of packet instead of an abort at the end of the packet.
- When the ALE receives a packet that contains errors (due to a set header error bit), or a mac control frame and does not receive an abort, the packet will be forwarded only to the host port (port 0).
- No ALE learning occurs on packets with errors or mac control frames. Learning is based on source address and lookup is based on destination address

- The ALE may be configured to operate in bypass mode by setting the ale_bypass bit in the ALE_Control register
- When in bypass mode, all CPGMAC_SL received packets are forwarded only to the host port (port 0).
- Packets from the two ports can be on separate Rx DMA channels by configuring the CPDMA_Rx_Ch_Map register.
- In bypass mode, the ALE processes host port transmit packets the same as in normal mode. In general, packets would be directed by the host in bypass mode.
- The ALE may be configured to operate in OUI deny mode by setting the enable_oui_deny bit in the ALE_Control register
- When in OUI deny mode, a packet with a non-matching OUI source address will be dropped unless the destination address matches a multicast table entry with the super bit set
- Broadcast packets will be dropped unless the broadcast address is entered into the table with the super bit set
- Unicast packets will be dropped unless the unicast address is in the table with block and secure both set
- Multicast supervisory packets are designated by the super bit in the table entry
- Unicast supervisory packets are indicated when block and secure are both set. Supervisory packets are not dropped due to rate limiting, OUI, or VLAN processing.

Address table entry

- The ALE table contains 1024 entries. Each table entry represents a free entry, an address, a VLAN, an address/VLAN pair, or an OUI address
- Software should ensure that there are not double address entries in the table
- The double entry used would be indeterminate. Reserved table bits must be written with zeroes.
- Source Address learning occurs for packets with a unicast, multicast or broadcast destination address and a unicast or multicast (including broadcast) source address
- Multicast source addresses have the group bit (bit 40) cleared before ALE processing begins, changing the multicast source address to a unicast source address
- .A multicast address of all ones is the broadcast address which may be added to the table. A learned unicast source address is added to the table with the following control bits:

Table 14-9. Learned Address Control Bits

unicast_type	11
Block	0
Secure	0

- If a received packet has a source address that is equal to the destination address then the following occurs:
  - The address is learned if the address is not found in the table.
  - The address is updated if the address is found.
  - The packet is dropped

14.3.2.7.1.1 Free Table Entry

Table 14-10. Free (Unused) Address Table Entry Bit Values

71:62	61:60	59:0
Reserved	Entry Type (00)	Reserved

14.3.2.7.1.2 Multicast Address Table Entry

Table 14-11. Multicast Address Table Entry Bit Values

71:70	68:66	65	64	63:62	61:60	59:48	47:0
Reserved	Port Mask	Super	Reserved	Mcast Fwd State	Entry Type (01)	Reserved	Multicast Address

Table Entry Type

00: Free Entry

01: Address Entry : unicast or multicast determined by dest **address bit 40 .**

10: VLAN entry

11: VLAN Address Entry : unicast or multicast determined by **address bit 40.**

Supervisory Packet (SUPER)

When set, this field indicates that the packet with a matching multicast destination address is a supervisory packet.

0: Non-supervisory packet

1: Supervisory packet

Port Mask(2:0) (PORT_MASK)

This three bit field is the port bit mask that is returned with a found multicast destination address. There may be multiple bits set indicating that the multicast packet may be forwarded to multiple ports (but not the receiving port).

Multicast Forward State (MCAST_FWD_STATE)

Multicast Forward State – Indicates the port state(s) required for the received port on a destination address lookup in order for the multicast packet to be forwarded to the transmit port(s). A transmit port must be in the Forwarding state in order to forward the packet. If the transmit port_mask has multiple set bits then each forward decision is independent of the other transmit port(s) forward decision.

00: Forwarding

01: Blocking/Forwarding/Learning

10: Forwarding/Learning

11: Forwarding

The forward state test returns a true value if both the Rx and Tx ports are in the required state.

Table Entry Type (ENTRY_TYPE)

Address entry type. Unicast or multicast determined by address bit 40.

01: Address entry. Unicast or multicast determined by address bit 40.

Packet Address (MULTICAST_ADDRESS)

This is the 48-bit packet MAC address. For an OUI address, only the upper 24-bits of the address are used in the source or destination address lookup. Otherwise, all 48-bits are used in the lookup.

14.3.2.7.1.3 VLAN/Multicast Address Table Entry

Table 14-12. VLAN/Multicast Address Table Entry Bit Values

71:69	68:66	65	64	63:62	61:60	59:48	47:0
Reserved	Port Mask	Super	Reserved	Mcast Fwd State	Entry Type (11)	vlan_id	Multicast Address

Supervisory Packet (SUPER)

When set, this field indicates that the packet with a matching multicast destination address is a supervisory packet.

0: Non-supervisory packet

1: Supervisory packet

Port Mask(2:0) (PORT_MASK)

This three bit field is the port bit mask that is returned with a found multicast destination address. There may be multiple bits set indicating that the multicast packet may be forwarded to multiple ports (but not the receiving port).

Multicast Forward State (MCAST_FWD_STATE)

Multicast Forward State – Indicates the port state(s) required for the received port on a destination address lookup in order for the multicast packet to be forwarded to the transmit port(s). A transmit port must be in the Forwarding state in order to forward the packet. If the transmit port_mask has multiple set bits then each forward decision is independent of the other transmit port(s) forward decision.

00 – Forwarding

01 – Blocking/Forwarding/Learning

10 – Forwarding/Learning

11 – Forwarding

The forward state test returns a true value if both the Rx and Tx ports are in the required state.

Table Entry Type (ENTRY_TYPE)

Address entry type. Unicast or multicast determined by address bit 40.

11: VLAN address entry. Unicast or multicast determined by address bit 40.

VLAN ID (VLAN_ID)

The unique identifier for VLAN identification. This is the 12-bit VLAN ID.

Packet Address (MULTICAST_ADDRESS)

This is the 48-bit packet MAC address. For an OUI address, only the upper 24-bits of the address are used in the source or destination address lookup. Otherwise, all 48-bits are used in the lookup.

14.3.2.7.1.4 Unicast Address Table Entry

Table 14-13. Unicast Address Table Entry Bit Values

71:70	69	68	67:66	65	64	63:62	61:60	59:48	47:0
Reserved	DLR Unicast	Reserved	Port Number	Block	Secure	Unicast Type (00) or (X1)	Entry Type (01)	Reserved	Unicast Address

DLR Unicast

DLR Unicast – When set, this bit indicates that the address is a Device Level Ring (DLR) unicast address. Received packets with a matching destination address will be flooded to the vlan_member_list (minus the receive port and the host port). The port_number field is a don't care when this bit is set. Matching packets received on port 1 egress on port 2. Matching packets received on port 2 egress on port 1. Matching packets received on port 0 egress on ports 1 and 2.

Port Number (PORT_NUMBER)

Port Number – This field indicates the port number (not port mask) that the packet with a unicast destination address may be forwarded to. Packets with unicast destination addresses are forwarded only to a single port (but not the receiving port).

Block (BLOCK)

Block – The block bit indicates that a packet with a matching source or destination address should be dropped (block the address).

0 – Address is not blocked.

1 – Drop a packet with a matching source or destination address (secure

must be zero)

If block and secure are both set, then they no longer mean block and secure. When both are set, the block and secure bits indicate that the packet is a unicast supervisory (super) packet and they determine the unicast forward state test criteria. If both bits are set then the packet is forwarded if the receive port is in the Forwarding/Blocking/Learning state. If both bits are not set then the packet is forwarded if the receive port is in the Forwarding state.

Secure (SECURE)

Secure – This bit indicates that a packet with a matching source address should be dropped if the received port number is not equal to the table entry port_number.

0 – Received port number is a don't care.

1 – Drop the packet if the received port is not the secure port for the source

address and do not update the address (block must be zero)

Unicast Type (UNICAST_TYPE)

Unicast Type – This field indicates the type of unicast address the table entry contains.

00 – Unicast address that is not ageable.

01 – Ageable unicast address that has not been touched.

10 – OUI address - lower 24-bits are don't cares (not ageable).

11 – Ageable unicast address that has been touched.

Table Entry Type (ENTRY_TYPE)

Address entry. Unicast or multicast determined by address bit 40.

01: Address entry. Unicast or multicast determined by address bit 40.

Packet Address (UNICAST_ADDRESS)

This is the 48-bit packet MAC address. All 48-bits are used in the lookup.

14.3.2.7.1.6 VLAN/Unicast Address Table Entry

Table 14-15. Unicast Address Table Entry Bit Values

71:68	67:66	65	64	63:62	61:60	59:48	47:0
Reserved	Port Number	Block	Secure	Unicast Type (00) or (X1)	Entry Type (11)	vlan_id	Unicast Address

Port Number (PORT_NUMBER)

Port Number – This field indicates the port number (not port mask) that the packet with a unicast destination address may be forwarded to. Packets with unicast destination addresses are forwarded only to a single port (but not the receiving port).]

Block (BLOCK)

Block – The block bit indicates that a packet with a matching source or destination address should be dropped (block the address).

0 – Address is not blocked.

1 – Drop a packet with a matching source or destination address (secure

must be zero)

If block and secure are both set, then they no longer mean block and secure. When both are set, the block and secure bits indicate that the packet is a unicast supervisory (super) packet and they determine the unicast forward state test criteria. If both bits are set then the packet is forwarded if the receive port is in the Forwarding/Blocking/Learning state. If both bits are not set then the packet is forwarded if the receive port is in the Forwarding state.

Secure (SECURE)

Secure – This bit indicates that a packet with a matching source address should be dropped if the received port number is not equal to the table entry port_number.

0 – Received port number is a don't care.

1 – Drop the packet if the received port is not the secure port for the source

address and do not update the address (block must be zero)

Unicast Type (UNICAST_TYPE)

Unicast Type – This field indicates the type of unicast address the table entry contains.

00 – Unicast address that is not ageable.

01 – Ageable unicast address that has not been touched.

10 – OUI address - lower 24-bits are don't cares (not ageable).

11 – Ageable unicast address that has been touched.

Table Entry Type (ENTRY_TYPE)

Address entry. Unicast or multicast determined by address bit 40.

11 – VLAN address entry. Unicast or multicast determined by address bit 40.

VLAN ID (VLAN_ID)

The unique identifier for VLAN identification. This is the 12-bit VLAN ID.

- Packet Address (UNICAST_ADDRESS) This is the 48-bit packet MAC address. All 48-bits are used in the lookup

14.3.2.7.1.7 VLAN Table Entry

Table 14-16. VLAN Table Entry

71:62	61:60	59:48	47:27	26:24	23:19	18:16	15:11	10:8	7:3	2:0
Reserved	Entry Type (10)	vlan_id	Reserved	Force Untagged Egress	Reserved	Reg Mcast Flood Mask	Reserved	Unreg Mcast Flood Mask	Reserved	Vlan Member List

Table Entry Type (ENTRY_TYPE)
10: VLAN entry

VLAN ID (VLAN_ID)
The unique identifier for VLAN identification. This is the 12-bit VLAN ID.

Force Untagged Packet Egress (FORCE_UNTAGGED_EGRESS)
This field causes the packet VLAN tag to be removed on egress (except on port 0).

Registered Multicast Flood Mask (REG_MCAST_FLOOD_MASK)
Mask used for multicast when the multicast address is found

Unregistered Multicast Flood Mask (UNREG_MCAST_FLOOD_MASK)
Mask used for multicast when the multicast address is not found

VLAN Member List (VLAN_MEMBER_LIST)
This three bit field indicates which port(s) are members of the associated VLAN.

Packet forwarding and learning

- There are four processes that an incoming received packet may go through to determine packet forwarding. The processes are Ingress Filtering, VLAN_Aware Lookup, VLAN_Unaware Lookup, and Egress
- Packet processing begins in the Ingress Filtering process. Each port has an associated packet forwarding state that can be one of four values (Disabled, Blocked, Learning, or Forwarding). The default state for all ports is disabled.
- The host sets the packet forwarding state for each port.
- The learning, updating, and touching processes are applied to each receive packet that is not aborted
- The processes are concurrent with the packet forwarding process.
- In addition to the following, a packet must be received without error in order to learn/update/touch an address.

FIFO

- Each of the three CPSW_3G ports has an identical associated FIFO. Each FIFO contains a single logical receive (ingress) queue and four logical transmit queues (priority 0 through 3).
- memory control

- Each FIFO memory contains 20,480 bytes (20k) total organized as 2560 by 64-bit words contained in a single memory instance
- The FIFO memory is used for the associated port transmit and receive queues
- The tx_max_blks field in the FIFO's associated Max_Blks register determines the maximum number of 1k FIFO memory blocks to be allocated to the four logical transmit queues
- The rx_max_blks field in the FIFO's associated Max_Blks register determines the maximum number of 1k memory blocks to be allocated to the logical receive queue
- The tx_max_blks value plus the rx_max_blks value must sum to 20 (the total number of blocks in the FIFO)
- If the sum were less than 20 then some memory blocks would be unused.The default is 17 (decimal) transmit blocks and three receive blocks
- The FIFO's follow the naming convention of the Ethernet ports.Host Port is Port0 and External Ports are Port1,2

Modes

- Normal Priority mode
  - When operating in normal mode, lower priority frames are dropped before higher priority frames
  - . The intention is to give preference to higher priority frames.
  - Priority 3 is the highest priority and is allowed to fill the FIFO
  - Priority 2 will drop packets if the packet is going to take space in the last 2k available
  - Priority 1 will drop packets if the packet is going to take space in the last 4k available
  - Priority 0 will drop packets if the packet is going to take space in the last 6k available
  - If fewer than 4 priorities are to be implemented then the priorities should be mapped such that the highest priorities are used
  - For example, if two priorities are going to be used then all packets should be mapped to priorities 3 and 2 and priorities 1 and 0 should be unused
  - Normal priority mode is configured as described below:
    - Select normal priority mode by setting tx_in_sel[1:0] = 00 for all ports (default value in P0/1/2_Tx_In_Ctl)

        - Configure priority mapping to use only the highest priorities if less than 4 priorities are used
- Dual Mac mode
    - When operating in dual mac mode the intention is to transfer packets between ports 0 and 1 and ports 0 and 2, but not between ports 1 and 2
    - Each CPGMAC_SL appears as a single MAC with no bridging between MAC's.
    - Each CPGMAC_SL has at least one unique (not the same) mac address.
    - Dual mac mode is configured as described below:
        - Set the ale_vlan_aware bit in the ALE_Control register.
        - Configure the Port 1 to Port 0 VLAN
            - Add a VLAN Table Entry with ports 0 and 1 as members (clear the flood masks).
            - Add a VLAN/Unicast Address Table Entry with the Port1/0 VLAN and a port number of 0.
            - Packets received on port 1 with this unicast address will be sent only to port 0 (egress).
        - Configure the Port 2 to Port 0 VLAN
            - Add a VLAN Table Entry with ports 0 and 2 as members (clear the flood masks)
            - Add a VLAN/Unicast Address Table Entry with the Port2/0 VLAN and a port number of 0.
            - Packets received on port 2 with this unicast address will be sent only to port 0 (egress)
        - Packets from the host (port 0) to ports 1 and 2 should be directed
        - Select the dual mac mode on the port 0 FIFO by setting tx_in_sel[1:0] = 01 in P0_Tx_In_Ctl
        - The intention of this mode is to allow packets from both ethernet ports to be written into the FIFO without one port starving the other port.
        - The priority levels may be configured such that packets received on port 1 egress on one CPDMA RX channel while packets received on port 2 egress on a different CPDMA RX channel.

Software reset

- The CPSW_3G software reset register, CPSW_3GSS software reset register and the three submodule software reset registers enable the CPSW_3GSS to be reset by

software

- There are three CPSW_3G submodules that contain software reset registers (CPGMAC_SL1, CPGMAC_SL2, and CPDMA).
- Each of the three submodules may be individually commanded to be reset by software.
- For the CPDMA, the reset state is entered at packet boundaries, at which time the CPDMA reset occurs
- The CPGMAC_SL soft reset is immediate.
- Submodule reset status is determined by reading or polling the submodule reset bit.
- If the submodule reset bit is read as a one, then the reset process has not yet completed
- The submodule soft reset process could take up to 2ms each.
- The reset has completed if the submodule reset bit is read as a zero

Network statistics

- The CPSW_3G has a set of statistics that record events associated with frame traffic on selected switch ports
- The statistics values are cleared to zero 38 clocks after the rising edge of VBUSP_RST_N
- When one or more port enable bits (stat_port_en[2:0]) are set, all statistics registers are write to decrement
- The value written will be subtracted from the register value with the result being stored in the register.
- If a value greater than the statistics value is written, then zero will be written to the register (writing 0xffffffff will clear a statistics location).
- When all port enable bits are cleared to zero, all statistics registers are read/write
- The statistics interrupt (STAT_PEND) will be issued if enabled when any statistics value is greater than or equal to 0x80000000.
- The statistics interrupt is removed by writing to decrement any statistics value greater than 0x80000000.
- The statistics are mapped into internal memory space and are 32-bits wide. All statistics rollover from 0xFFFFFFFF to 0x00000000.

CPTS module

- The CPTS module is used to facilitate host control of time sync operations

- The CPTS collects time sync events and then presents them to the host for processing.
- There are five types of time sync events (ethernet receive event, ethernet transmit event, time stamp push event, time stamp rollover event, and time stamp half-rollover event).
- Each ethernet port can cause transmit and receive events. The time stamp push is initiated by software.
- The CPTS module should be configured as shown:
  - Complete the reset sequence (VBUSP_RST_N) to reset the module
  - Write the rftclk_sel[4:0] value in the RFTCLK_Sel register with the desired reference clock multiplexor value. This value is allowed to be written only when the cpts_en bit is cleared to zero
  - Write a one to the cpts_en bit in the TS_Control register. The RCLK domain is in reset while this bit is low
  - Enable the interrupt by writing a one to the ts_pend_en bit in the TS_Int_Enable register (if using interrupts and not polling)

Ethernet port events

- receive event
  - Each ethernet port can generate a receive ethernet event
  - Receive ethernet events are generated for valid received time sync packets
- transmit event
  - Each ethernet port can generate a transmit ethernet event
  - Transmit ethernet events are generated for valid transmitted time sync packets

Event Interrupt handling

- When an event is push onto the Event FIFO, an interrupt can be generated to indicate to software that a time sync event occurred
- The following steps should be taken to process time sync events using interrupts
  - Enable the TS_PEND interrupt by setting the TS_PEND_EN bit of the CPTS_INT_ENABLE register
  - Upon interrupt, read the CPTS_EVENT_LOW and CPTS_EVENT_HIGH register values
  - Set the EVENT_POP field (bit zero) of the CPTS_EVENT_POP register to pop the previously read value off of the event FIFO

    - Process the interrupt as required by the application software
- Software has the option of processing more than a single event from the event FIFO in the interrupt service routine in the following way
    - Enable the TS_PEND interrupt by setting the TS_PEND_EN bit of the CPTS_INT_ENABLE register.
    - Upon interrupt enter the CPTS service routine
    - Read the CPTS_EVENT_LOW and CPTS_EVENT_HIGH register values
    - Set the EVENT_POP bit of the CPTS_EVENT_POP register to pop the previously read value off of the event FIFO
    - Wait for an amount of time greater than eight CPTS_RFT_CLK periods
    - Read the ts_pend_raw bit in the CPTS_INTSTAT_RAW register to determine if another valid event is in the event FIFO. If it is asserted then goto step 3. Otherwise goto step 7
    - Process the interrupt(s) as required by the application software
- Software also has the option of disabling the interrupt and polling the ts_pend_raw bit of the CPTS_INTSTAT_RAW register to determine if a valid event is on the event FIFO

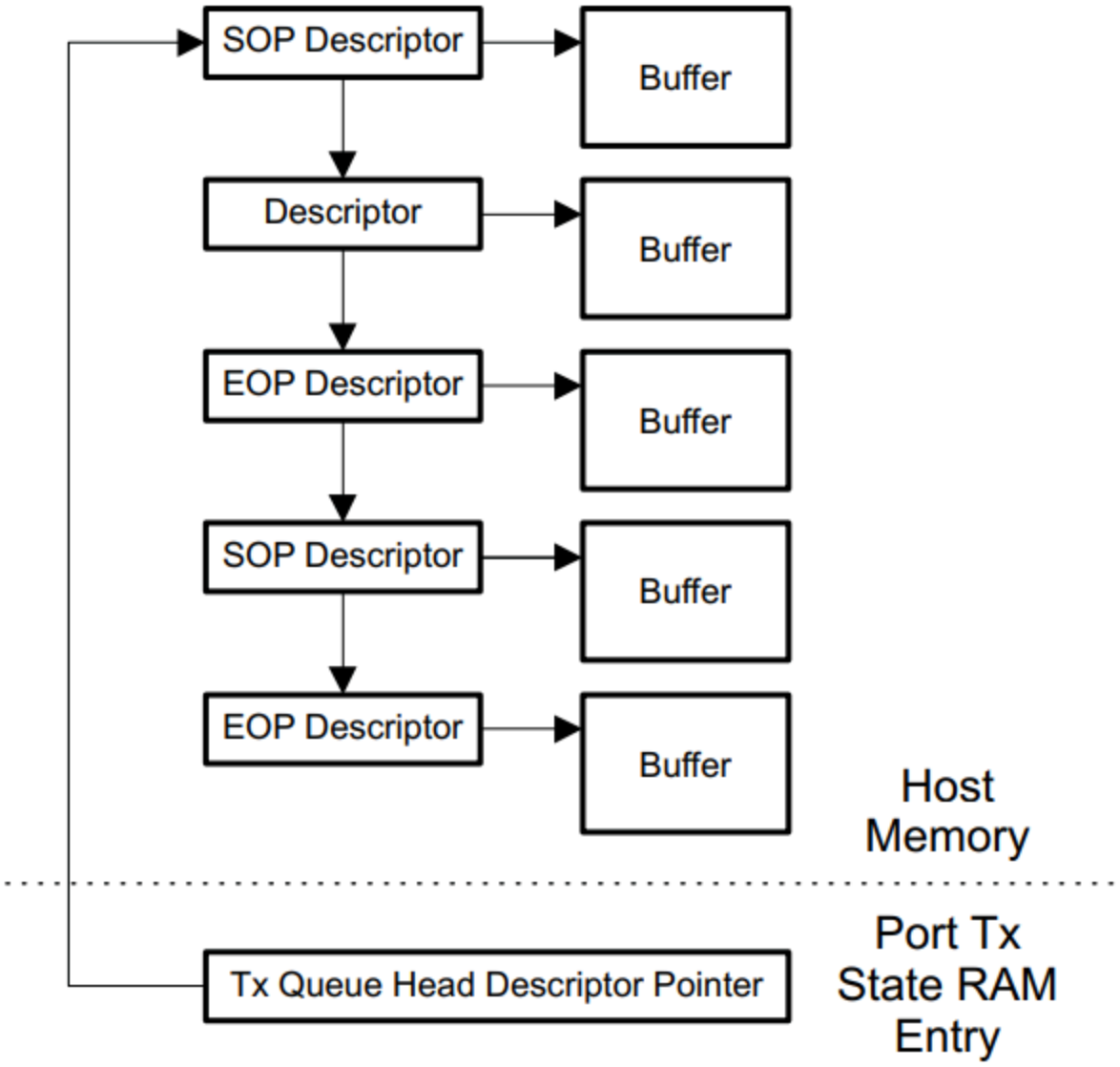
Software Operations

- transmit operation
    - After reset the host must write zeroes to all Tx DMA State head descriptor pointers
    - The Tx port may then be enabled
    - To initiate packet transmission the host constructs transmit queues in memory (one or more packets for transmission) and then writes the appropriate Tx DMA state head descriptor pointers.
    - For each buffer added to a transmit queue, the host must initialize the Tx buffer descriptor values as follows:
        - Write the Next Descriptor Pointer with the 32-bit aligned address of the next descriptor in the queue
        - Write the Buffer Pointer with the byte aligned address of the buffer data
        - Write the Buffer Length with the number of bytes in the buffer
        - Write the Buffer Offset with the number of bytes in the offset to the data (nonzero with SOP only)
        - Set the SOP, EOP, and Ownership bits as appropriate

    - Clear the End Of Queue bit
- The port begins Tx packet transmission on a given channel when the host writes the channel's Tx queue head descriptor pointer with the address of the first buffer descriptor in the queue
- Each channel may have one or more queues, so each channel may have one or more head descriptor pointers
- The first buffer descriptor for each Tx packet must have the Start of Packet (SOP) bit and the Ownership bit set to one by the host
- The last buffer descriptor for each Tx packet must have the End of Packet (EOP) bit set to one by the host.
- The port will transmit packets until all queued packets have been transmitted and the queue(s) are empty
- When each packet transmission is complete, the port will clear the Ownership bit in the packet's SOP buffer descriptor and issue an interrupt to the host by writing the packet's last buffer descriptor address to the queue's Tx DMA State Completion Pointer.
- The interrupt is generated by the write, regardless of the value written
- When the last packet in a queue has been transmitted, the port sets the End Of Queue bit in the EOP buffer descriptor, clears the Ownership bit in the SOP Descriptor, zeroes the appropriate DMA state head descriptor pointer, and then issues a Tx interrupt to the host by writing to the queue's associated Tx completion pointer (address of the last buffer descriptor processed by the port).
- The port issues a maskable level interrupt (which may then be routed through external interrupt control logic to the host).
- On interrupt from the port, the host processes the buffer queue, detecting transmitted packets by the status of the Ownership bit in the SOP buffer descriptor.
- If the Ownership bit is cleared to zero, then the packet has been transmitted and the host may reclaim the buffers associated with the packet.
- The host continues queue processing until the end of the queue or until a SOP buffer descriptor is read that contains a set Ownership bit indicating that the packet transmission is not complete
- The host determines that all packets in the queue have been transmitted when the last packet in the queue has a cleared Ownership bit in the SOP buffer descriptor, the End of Queue bit is set in the last packet EOP buffer descriptor, and the Next Descriptor Pointer of the last packet EOP buffer descriptor is zero

- The host acknowledges an interrupt by writing the address of the last buffer descriptor to the queue's associated Tx Completion Pointer in the Tx DMA State
- If the host written buffer address value is different from the buffer address written by the port, then the level interrupt remains asserted
- If the host written buffer address value is equal to the port written value, then the level interrupt is deasserted.
- The port write to the completion pointer actually stores the value in the state register (ram)
- The host written value is actually not written to the register location
- The host written value is compared to the register contents (which was written by the port) and if the two values are equal, the interrupt is removed, otherwise the interrupt remains asserted.
- The host may process multiple packets previous to acknowledging an interrupt, or the host may acknowledge interrupts for every packet
- The host may add packets to the tail end of an active Tx queue at any time by writing the Next Descriptor Pointer to the current last descriptor in the queue.
- If a Tx queue is empty (inactive), the host may initiate packet transmission at any time by writing the appropriate Tx DMA State head descriptor pointer
- If the End of Packet bit is set and the Next Descriptor Pointer is nonzero, then the queue still contains one or more packets to be transmitted
- If the EOP bit is set with a zero Next Descriptor Pointer, then the port will set the EOQ bit in the packet's EOP buffer descriptor and then zero the appropriate head descriptor pointer previous to interrupting the port (by writing the completion pointer) when the packet transmission is complete

Figure 14-13. Port TX State RAM Entry

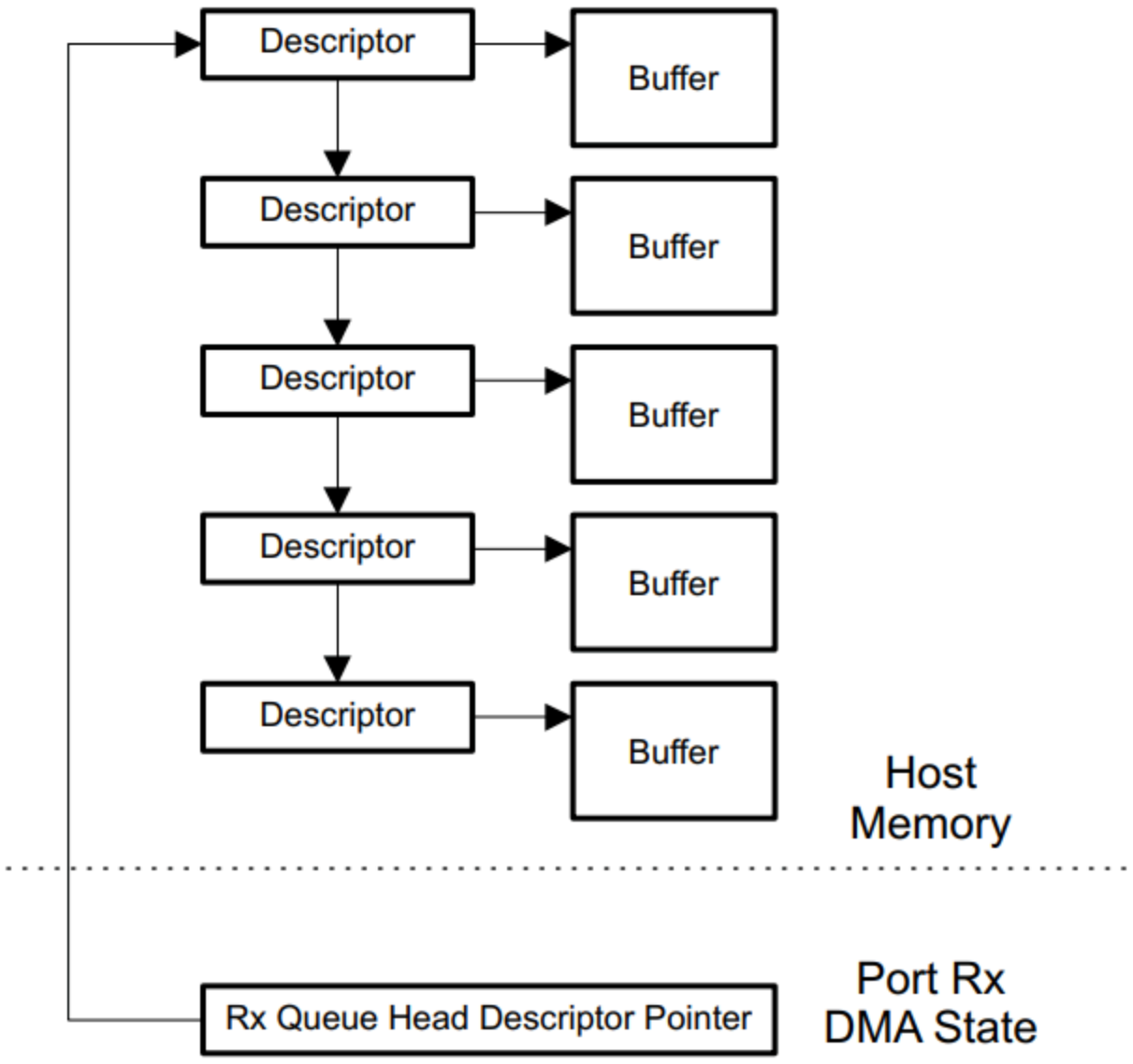


- Receive Operation
  - After reset the host must write zeroes to all Rx DMA State head descriptor pointers
  - The Rx port may then be enabled
  - To initiate packet reception, the host constructs receive queues in memory and then writes the appropriate Rx DMA state head descriptor pointer
  - For each Rx buffer descriptor added to the queue, the host must initialize the Rx buffer descriptor values as follows
    - Write the Next Descriptor Pointer with the 32-bit aligned address of the next descriptor in the queue (zero if last descriptor)
    - Write the Buffer Pointer with the byte aligned address of the buffer data
    - Clear the Offset field

    - Write the Buffer Length with the number of bytes in the buffer
    - Clear the SOP, EOP, and EOQ bits
    - Set the Ownership bit
- The host enables packet reception on a given channel by writing the address of the first buffer descriptor in the queue (nonzero value) to the channel's head descriptor pointer in the channel's Rx DMA state
- When packet reception begins on a given channel, the port fills each Rx buffer with data in order starting with the first buffer and proceeding through the Rx queue
- If the Buffer Offset in the Rx DMA State is nonzero, then the port will begin writing data after the offset number of bytes in the SOP buffer
- The port performs the following operations at the end of each packet reception:
    - Overwrite the buffer length in the packet's EOP buffer descriptor with the number of bytes actually received in the packet's last buffer
    - The host initialized value is the buffer size. The overwritten value will be less than or equal to the host initialized value.
    - Set the EOP bit in the packet's EOP buffer descriptor
    - Set the EOQ bit in the packet's EOP buffer descriptor if the current packet is the last packet in the queue
    - Overwrite the packet's SOP buffer descriptor Buffer Offset with the Rx DMA state value (the host initialized the buffer descriptor Buffer Offset value to zero).
    - All non SOP buffer descriptors must have a zero Buffer Offset initialized by the host
    - Overwrite the packet's SOP buffer descriptor buffer length with the number of valid data bytes in the buffer. If the buffer is filled up, the buffer length will be the buffer size minus buffer offset
    - Set the SOP bit in the packet's SOP buffer descriptor
    - Write the SOP buffer descriptor Packet Length field
    - Clear the Ownership bit in the packet's SOP buffer descriptor
    - Issue an Rx host interrupt by writing the address of the packet's last buffer descriptor to the queue's Rx DMA State Completion Pointer. The interrupt is generated by the write to the Rx DMA State Completion Pointer address location, regardless of the value written.
- On interrupt the host processes the Rx buffer queue detecting received packets by the status of the Ownership bit in each packet's SOP buffer descriptor.

- If the Ownership bit is cleared then the packet has been completely received and is available to be processed by the host
- The host may continue Rx queue processing until the end of the queue or until a buffer descriptor is read that contains a set Ownership bit indicating that the next packet's reception is not complete.
- The host determines that the Rx queue is empty when the last packet in the queue has a cleared Ownership bit in the SOP buffer descriptor, a set End of Queue bit in the EOP buffer descriptor, and the Next Descriptor Pointer in the EOP buffer descriptor is zero.

Figure 14-14. Port RX DMA State



Initializing MDIO module

- The following steps are performed by the application software or device driver to initialize the MDIO device
    - Configure the PREAMBLE and CLKDIV bits in the MDIO control register
    - Enable the MDIO module by setting the ENABLE bit in MDIOCONTROL
    - The MDIO PHY alive status register (MDIOALIVE) can be read in polling fashion until a PHY connected to the system responded, and the MDIO PHY link status register (MDIOLINK) can determine whether this PHY already has a link
    - Setup the appropriate PHY addresses in the MDIO user PHY select register (MDIOUSERPHYSELn), and set the LINKINTENB bit to enable a link change event interrupt if desirable
    - If an interrupt on general MDIO register access is desired, set the corresponding bit in the MDIO user command complete interrupt mask set register (MDIOUSERINTMASKSET) to use the MDIO user access register (MDIOUSERACCESSn).
    - Since only one PHY is used in this device, the application software can use one MDIOUSERACCESSn to trigger a completion interrupt; the other MDIOUSERACCESSn is not setup.

I/O for PHY registers

- The MDIO module includes a user access register (MDIOUSERACCESSn) to directly access a specified PHY device
- writing
    - To write a PHY register, perform the following:
        - Ensure the GO bit in the MDIO user access register (MDIOUSERACCESSn) is cleared
        - Write to the GO, WRITE, REGADR, PHYADR, and DATA bits in MDIOUSERACCESSn corresponding to the PHY and PHY register you want to write
        - The write operation to the PHY is scheduled and completed by the MDIO module. Completion of the write operation can be determined by polling the GO bit in MDIOUSERACCESSn for a 0
        - Completion of the operation sets the corresponding USERINTRAW bit (0 or 1) in the MDIO user command complete interrupt register (MDIOUSERINTRAW) corresponding to USERACCESSn used. If interrupts have been enabled on this bit using the MDIO user command complete interrupt mask set register (MDIOUSERINTMASKSET), then the bit is also

set in the MDIO user command complete interrupt register (MDIOUSERINTMASKED) and an interrupt is triggered on the CPU

- reading
  - To read a PHY register, perform the following
    - Ensure the GO bit in the MDIO user access register (MDIOUSERACCESSn) is cleared
    - Write to the GO, REGADR, and PHYADR bits in MDIOUSERACCESSn corresponding to the PHY and PHY register you want to read
    - The read data value is available in the DATA bits in MDIOUSERACCESSn after the module completes the read operation on the serial bus. Completion of the read operation can be determined by polling the GO and ACK bits in MDIOUSERACCESSn. After the GO bit has cleared, the ACK bit is set on a successful read
    - Completion of the operation sets the corresponding USERINTRAW bit (0 or 1) in the MDIO user command complete interrupt register (MDIOUSERINTRAW) corresponding to USERACCESSn used. If interrupts have been enabled on this bit using the MDIO user command complete interrupt mask set register (MDIOUSERINTMASKSET), then the bit is also set in the MDIO user command complete interrupt register (MDIOUSERINTMASKED) and an interrupt is triggered on the CPU

Initialization of CPSW

- To configure the 3PSW Ethernet Subsystem for operation the host must perform the following:
  - Select the Interface (GMII/RGMII/MII) Mode in the Control Module.
  - Configure pads (PIN muxing) as per the Interface Selected using the appropriate pin muxing conf_xxx registers in the Control Module.
  - Enable the 3PSW Ethernet Subsystem Clocks. See Section 14.2.2 and Section 8.1 to enable the appropriate clocks.
  - Configure the PRCM registers CM_PER_CPSW_CLKSTCTRL and CM_PER_CPGMAC0_CLKCTRL to enable power and clocks to the 3PSW Ethernet Subsystem. See Section 8.1 for register details.
  - Apply soft reset to 3PSW Subsytem, CPSW_3G, CPGMAC_SL1/2, and CPDMA (see the soft reset registers in the following sections).
  - Initialize the HDPs (Header Description Pointer) and CPs (Completion Pointer) to NULL

- Configure the Interrupts (see Chapter 6).
- Configure the CPSW_3G Control register.
- Configure the Statistics Port Enable register
- Configure the ALE. (See Section 14.3.2.7.)
- Configure the MDIO.
- Configure the CPDMA receive DMA controller.
- Configure the CPDMA transmit DMA controller.
- Configure the CPPI Tx and Rx Descriptors.
- Configure CPGMAC_SL1 and CPGMAC_SL2 as per the desired mode of operations.
- Start up RX and TX DMA (write to HDP of Rx and Tx).
- Wait for the completion of the transfer (HDP cleared to zero).

Registers

CPSW_ALE

- base address
  - 0x4A10_0D00
- IDVER
  - offset
    - 0h
  - Address lookup engine ID/Version register
    - contains ALE identification number
    - contains ALE major and minor version numbers
- CONTROL
  - offset
    - 8h
  - Address lookup engine control register
    - enable ale
      - if 0 it drops all packets
    - clear table
    - enable different ALE modes
    - enable bypass mode
      - when set all packets received on ports 0 and 1 are sent to the host

        - rate limiting
- PRESCALE
    - offset
        - 10h
    - Address lookup engine prescale register
        - the input clock is divided by this value for use in the multicast/boradcast rate limiters
- UNKNOWN_VLAN
    - offset
        - 18h
    - Address lookup engine unknown vlan register
        - No clue
- TBLCTL
    - offset
        - 20h
    - Address lookup engine table control
        - The entry_pointer contains the table entry value that will be read/written with accesses to the table word registers
        - Writing a 1 to this bit causes the three table word register values to be written to the entry_pointer location in the address table
        - Writing a 0 to this bit causes the three table word register values to be loaded from the entry_pointer location in the address table so that they may be subsequently read.
- TBLW2
    - offset
        - 34h
    - Address lookup engine table word 2 register
        - Table entry bits 71:64
- TBLW1
    - offset
        - 38h
    - Address lookup engine table word 1 register
        - Table entry bits 63:32
- TBLW0
    - offset

        - 3Ch
    - Address lookup engine table word 0 register
        - Table entry bits 31:0
- PORTCTL0
    - offset
        - 40h
    - Address lookup engine port 0 control register
        - set port state
        - set rate limit
- PORTCTL1
    - offset
        - 44h
    - Address lookup engine port 1 control register
        - set port state
        - set rate limit
- PORTCTL2
    - offset
        - 48h
    - Address lookup engine port 2 control register
        - set port state
        - set rate limit
- PORTCTL3
    - offset
        - 4Ch
    - Address lookup engine port 3 control register
        - set port state
        - set rate limit
- PORTCTL4
    - offset
        - 50h
    - Address lookup engine port 4 control register
        - set port state
        - set rate limit
- PORTCTL5
    - offset

        - 54h
    - Address lookup engine port 5 control register
        - set port state
        - set rate limit

CPSW_CPDMA

- base address
    - 0x4A10_0800
- TX_IDVER
    - offset
        - 0h
    - CPDMA regs TX identification and version register
        - TX identification value
        - TX major and minor version number
- TX_CONTROL
    - offset
        - 4h
    - CPDMA regs TX control register
        - TX DMA enable and disable
- TX_TEARDOWN
    - offset
        - 8h
    - CPDMA regs TX teardown register
        - used to command a software teardown of a TX channel
- RX_IDVER
    - offset
        - 10h
    - CPDMA regs RX identification and version register
        - RX identification value
        - RX major and minor version number
- RX_CONTROL
    - offset
        - 14h
    - CPDMA regs RX control register

        - RX DMA enable and disable
- RX_TEARDOWN
    - offset
        - 18h
    - CPDMA regs RX teardown register
        - used to command a software teardown of a RX channel
- CPDMA_SOFT_RESET
    - offset
        - 1Ch
    - CPDMA regs soft reset register
        - used to software reset CPDMA
- DMACONTROL
    - offset
        - 20h
    - CPDMA regs CPDMA control register
        - used to rate limit transmit channels
        - configure to copy error frames to memory or not
        - set transmit priority type
- DMASTATUS
    - offset
        - 24h
    - CPDMA regs CPDMA status register
        - check if DMA controller is idle
        - TX error codes
        - TX error channel
        - RX error codes
        - RX error channel
- RX_BUFFER_OFFSET
    - offset
        - 28h
    - CPDMA regs receive buffer offset
        - offset of frame data in buffer
- EMCONTROL
    - offset

        - 2Ch
    - CPDMA regs emulation control
        - no clue
- TX_PRI0_RATE
    - offset
        - 30h
    - CPDMA regs transmit ingress priority 0 rate
        - set priority idle and send count
- TX_PRI1_RATE
    - offset
        - 34h
    - CPDMA regs transmit ingress priority 1 rate
        - set priority idle and send count
- TX_PRI2_RATE
    - offset
        - 38h
    - CPDMA regs transmit ingress priority 2 rate
        - set priority idle and send count
- TX_PRI3_RATE
    - offset
        - 3Ch
    - CPDMA regs transmit ingress priority 3 rate
        - set priority idle and send count
- TX_PRI4_RATE
    - offset
        - 40h
    - CPDMA regs transmit ingress priority 4 rate
        - set priority idle and send count
- TX_PRI5_RATE
    - offset
        - 44h
    - CPDMA regs transmit ingress priority 5 rate
        - set priority idle and send count
- TX_PRI6_RATE
    - offset

        - 48h
    - CPDMA regs transmit ingress priority 6 rate
        - set priority idle and send count
- TX_PRI7_RATE
    - offset
        - 4Ch
    - CPDMA regs transmit ingress priority 7 rate
        - set priority idle and send count
- TX_INTSTAT_RAW
    - offset
        - 80h
    - CPDMA int TX interrupt status register (raw value)
        - what channel caused the interrupt before mask
- TX_INTSTAT_MASKED
    - offset
        - 84h
    - CPDMA int TX interrupt status register (masked value)
        - what channel caused the interrupt after mask
- TX_INTMASK_SET
    - offset
        - 88h
    - CPDMA int TX interrupt mask set register
        - set what channels can interrupt
- TX_INTMASK_CLEAR
    - offset
        - 8Ch
    - CPDMA int TX interrupt mask clear register
        - disable channel interrupts
- CPDMA_IN_VECTOR
    - offset
        - 90h
    - CPDMA int input vector (read only)
        - no clue
- CPDMA_EOI_VECTOR
    - offset

        - 94h
    - CPDMA int end of interrupt vector
        - No clue
- RX_INTSTAT_RAW
    - offset
        - A0h
    - CPDMA int RX interrupt status register (raw value)
        - what channel caused interrupt before mask
- RX_INTSTAT_MASKED
    - offset
        - A4h
    - CPDMA int RX interrupt status register (masked value)
        - what channel caused interrupt after mask
- RX_INTMASK_SET
    - offset
        - A8h
    - CPDMA int RX interrupt mask set register
        - enable channel interrupts
- RX_INTMASK_CLEAR
    - offset
        - ACh
    - CPDMA int RX interrupt mask clear register
        - disable channel interrupts
- DMA_INTSTAT_RAW
    - offset
        - B0h
    - CPDMA int DMA interrupt status register (raw value)
        - tells us if a host pending or statistics pending interrupt is waiting before mask
- DMA_INTSTAT_MASKED
    - offset
        - B4h
    - CPDMA int DMA interrupt status register (masked value)
        - tells us if a host pending or statistics pending interrupt is waiting after mask
- DMA_INTMASK_SET

    - offset
        - B8h
    - CPDMA int DMA interrupt mask set register
        - enable host error or statistics interrupt
- DMA_INTMASK_CLEAR
    - offset
        - BCh
    - CPDMA int DMA interrupt mask clear register
        - disable host error or statistics interrupt
- RX0_PENDTHRESH
    - offset
        - C0h
    - CPDMA int receive threshold pending register channel 0
        - contains the threshold value for issuing receive threshold pending interrupts
- RX1_PENDTHRESH
    - offset
        - C4h
    - CPDMA int receive threshold pending register channel 1
        - contains the threshold value for issuing receive threshold pending interrupts
- RX2_PENDTHRESH
    - offset
        - C8h
    - CPDMA int receive threshold pending register channel 2
        - contains the threshold value for issuing receive threshold pending interrupts
- RX3_PENDTHRESH
    - offset
        - CCh
    - CPDMA int receive threshold pending register channel 3
        - contains the threshold value for issuing receive threshold pending interrupts
- RX4_PENDTHRESH
    - offset
        - D0h
    - CPDMA int receive threshold pending register channel 4
        - contains the threshold value for issuing receive threshold pending interrupts
- RX5_PENDTHRESH

    - offset
        - D4h
    - CPDMA int receive threshold pending register channel 5
        - contains the threshold value for issuing receive threshold pending interrupts
- RX6_PENDTHRESH
    - offset
        - D8h
    - CPDMA int receive threshold pending register channel 6
        - contains the threshold value for issuing receive threshold pending interrupts
- RX7_PENDTHRESH
    - offset
        - DCh
    - CPDMA int receive threshold pending register channel 7
        - contains the threshold value for issuing receive threshold pending interrupts
- RX0_FREEBUFFER
    - offset
        - E0h
    - CPDMA int receive free buffer register channel 0
        - number of free buffers available
- RX1_FREEBUFFER
    - offset
        - E4h
    - CPDMA int receive free buffer register channel 1
        - number of free buffers available
- RX2_FREEBUFFER
    - offset
        - E8h
    - CPDMA int receive free buffer register channel 2
        - number of free buffers available
- RX3_FREEBUFFER
    - offset
        - ECh
    - CPDMA int receive free buffer register channel 3
        - number of free buffers available
- RX4_FREEBUFFER

    - offset
        - F0h
    - CPDMA int receive free buffer register channel 4
        - number of free buffers available
- RX5_FREEBUFFER
    - offset
        - F4h
    - CPDMA int receive free buffer register channel 5
        - number of free buffers available
- RX6_FREEBUFFER
    - offset
        - F8h
    - CPDMA int receive free buffer register channel 6
        - number of free buffers available
- RX7_FREEBUFFER
    - offset
        - FCh
    - CPDMA int receive free buffer register channel 7
        - number of free buffers available

CPSW CPTS

- base address
    - 0x4A10_0C00
- CPTS_IDVER
    - offset
        - 0h
    - identification and version register
        - TX identification value
        - RTL version value
        - major and minor version
- CPTS_CONTROL
    - offset
        - 4h
    - time sync control register

        - enable or disable time sync
- CPTS_TS_PUSH
    - offset
        - Ch
    - time stamp event push register
        - no clue
- CPTS_TS_LOAD_VAL
    - offset
        - 10h
    - time stamp load value register
        - no clue
- CPTS_TS_LOAD_EN
    - offset
        - 14h
    - time stamp load enable register
        - enable time stamp load
- CPTS_INTSTAT_RAW
    - offset
        - 20h
    - time sync interrupt status raw register
        - indicates if there are events in FIFO before enable
- CPTS_INTSTAT_MASKED
    - offset
        - 24h
    - time sync interrupt status masked register
        - indicates if there are events in FIFO after enable
- CPTS_INT_ENABLE
    - offset
        - 28h
    - time sync interrupt enable register
        - enable interrupts
- CPTS_EVENT_POP
    - offset
        - 30h
    - event interrupt pop register

        - no clue
- CPTS_EVENT_LOW
    - offset
        - 34h
    - lower 32 bits of the event value
        - time stamp value
- CPTS_EVENT_HIGH
    - offset
        - 38h
    - upper 32 bits of the event value
        - time stamp header

CPSW_STATS

- come back to when and if i want to implement statistics

CPDMA_STATERAM

- memory mapped register for the CPSW_CPDMA
- base address
    - 0x4A10_0A00
- TX0_HDP
    - offset
        - 00h
    - Tx channel 0 head descriptor pointer
        - Writing a TX DMA Buffer Descriptor address to a head pointer location initiates TX DMA operations in the queue for the selected channel
- TX1_HDP
    - offset
        - 04h
    - Tx channel 1 head descriptor pointer
        - Writing a TX DMA Buffer Descriptor address to a head pointer location initiates TX DMA operations in the queue for the selected channel
- TX2_HDP
    - offset
        - 08h

- Tx channel 2 head descriptor pointer
    - Writing a TX DMA Buffer Descriptor address to a head pointer location initiates TX DMA operations in the queue for the selected channel
- TX3_HDP
    - offset
        - 0Ch
    - Tx channel 3 head descriptor pointer
        - Writing a TX DMA Buffer Descriptor address to a head pointer location initiates TX DMA operations in the queue for the selected channel
- TX4_HDP
    - offset
        - 10h
    - Tx channel 4 head descriptor pointer
        - Writing a TX DMA Buffer Descriptor address to a head pointer location initiates TX DMA operations in the queue for the selected channel
- TX5_HDP
    - offset
        - 14h
    - Tx channel 5 head descriptor pointer
        - Writing a TX DMA Buffer Descriptor address to a head pointer location initiates TX DMA operations in the queue for the selected channel
- TX6_HDP
    - offset
        - 18h
    - Tx channel 6 head descriptor pointer
        - Writing a TX DMA Buffer Descriptor address to a head pointer location initiates TX DMA operations in the queue for the selected channel
- TX7_HDP
    - offset
        - 1Ch
    - Tx channel 7 head descriptor pointer
        - Writing a TX DMA Buffer Descriptor address to a head pointer location initiates TX DMA operations in the queue for the selected channel
- RX0_HDP
    - offset

        - 20h
    - Rx channel 0 head descriptor pointer
        - Writing an RX DMA Buffer Descriptor address to this location allows RX DMA operations in the selected channel when a channel frame is received
- RX1_HDP
    - offset
        - 24h
    - Rx channel 1 head descriptor pointer
        - Writing an RX DMA Buffer Descriptor address to this location allows RX DMA operations in the selected channel when a channel frame is received
- RX2_HDP
    - offset
        - 28h
    - Rx channel 2 head descriptor pointer
        - Writing an RX DMA Buffer Descriptor address to this location allows RX DMA operations in the selected channel when a channel frame is received
- RX3_HDP
    - offset
        - 2Ch
    - Rx channel 3 head descriptor pointer
        - Writing an RX DMA Buffer Descriptor address to this location allows RX DMA operations in the selected channel when a channel frame is received
- RX4_HDP
    - offset
        - 30h
    - Rx channel 4 head descriptor pointer
        - Writing an RX DMA Buffer Descriptor address to this location allows RX DMA operations in the selected channel when a channel frame is received
- RX5_HDP
    - offset
        - 34h
    - Rx channel 5 head descriptor pointer
        - Writing an RX DMA Buffer Descriptor address to this location allows RX DMA operations in the selected channel when a channel frame is received
- RX6_HDP

    - offset
        - 38h
    - Rx channel 6 head descriptor pointer
        - Writing an RX DMA Buffer Descriptor address to this location allows RX DMA operations in the selected channel when a channel frame is received
- RX7_HDP
    - offset
        - 3Ch
    - Rx channel 7 head descriptor pointer
        - Writing an RX DMA Buffer Descriptor address to this location allows RX DMA operations in the selected channel when a channel frame is received
- TX0_CP
    - offset
        - 40h
    - Tx channel 0 completion pointer register
        - This register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing. The port uses the value written to determine if the interrupt should be deasserted.
- TX1_CP
    - offset
        - 44h
    - Tx channel 1 completion pointer register
        - This register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing. The port uses the value written to determine if the interrupt should be deasserted.
- TX2_CP
    - offset
        - 48h
    - Tx channel 2 completion pointer register
        - This register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing. The port uses the value written to determine if the interrupt should be deasserted.
- TX3_CP
    - offset
        - 4Ch

    - Tx channel 3 completion pointer register
        - This register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing. The port uses the value written to determine if the interrupt should be deasserted.
- TX4_CP
    - offset
        - 50h
    - Tx channel 4 completion pointer register
        - This register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing. The port uses the value written to determine if the interrupt should be deasserted.
- TX5_CP
    - offset
        - 54h
    - Tx channel 5 completion pointer register
        - This register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing. The port uses the value written to determine if the interrupt should be deasserted.
- TX6_CP
    - offset
        - 58h
    - Tx channel 6 completion pointer register
        - This register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing. The port uses the value written to determine if the interrupt should be deasserted.
- TX7_CP
    - offset
        - 5Ch
    - Tx channel 7 completion pointer register
        - This register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing. The port uses the value written to determine if the interrupt should be deasserted.
- RX0_CP
    - offset
        - 60h

- Rx channel 0 completion pointer register
  - This register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing. The port uses the value written to determine if the interrupt should be deasserted
- RX1_CP
  - offset
    - 64h
  - Rx channel 1 completion pointer register
    - This register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing. The port uses the value written to determine if the interrupt should be deasserted
- RX2_CP
  - offset
    - 68h
  - Rx channel 2 completion pointer register
    - This register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing. The port uses the value written to determine if the interrupt should be deasserted
- RX3_CP
  - offset
    - 6Ch
  - Rx channel 3 completion pointer register
    - This register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing. The port uses the value written to determine if the interrupt should be deasserted
- RX4_CP
  - offset
    - 70h
  - Rx channel 4 completion pointer register
    - This register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing. The port uses the value written to determine if the interrupt should be deasserted
- RX5_CP
  - offset
    - 74h

    - Rx channel 5 completion pointer register
        - This register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing. The port uses the value written to determine if the interrupt should be deasserted
- RX6_CP
    - offset
        - 78h
    - Rx channel 6 completion pointer register
        - This register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing. The port uses the value written to determine if the interrupt should be deasserted
- RX7_CP
    - offset
        - 7Ch
    - Rx channel 7 completion pointer register
        - This register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing. The port uses the value written to determine if the interrupt should be deasserted

CPSW_PORT

- base address
    - 0x4A10_0100
- P0_CONTROL
    - offset
        - 0h
    - Port 0 control register
        - enable disable vlan
        - config priorities
- P0_MAX_BLKS
    - offset
        - 8h
    - Port 0 maximum fifo blocks register
        - set the maximum 1kb blocks for fifo
        - keep as default
- P0_BLK_CNT

    - offset
        - Ch
    - Port 0 fifo block usage count
        - read the number of fifo blocks are in use
- P0_TX_IN_CTL
    - offset
        - 10h
    - Port 0 transmit fifo control
        - select queue type
- P0_PORT_VLAN
    - offset
        - 14h
    - Port 0 vlan register
        - vlan config for the port
- P0_TX_PRI_MAP
    - offset
        - 18h
    - Port 0 Tx header prio to switch prio mapping register
        - no clue
        - keep default
- P0_CPDMA_TX_PRI_MAP
    - offset
        - 1Ch
    - CPDMA Tx (port 0 Rx) pkt priority to header priority
        - no clue
        - keep default
- P0_CPDMA_RX_CH_MAP
    - offset
        - 20h
    - CPDMA Rx (port 0 Tx) pkt priority to dma channel
        - no clue
        - keep default
- P0_RX_DSCP_PRI_MAP0
    - offset
        - 30h

    - PORT 0 RX DSCP PRIORITY TO RX PACKET MAPPING REG 0
- P0_RX_DSCP_PRI_MAP1
    - offset
        - 34h
    - PORT 0 RX DSCP PRIORITY TO RX PACKET MAPPING REG 1
- P0_RX_DSCP_PRI_MAP2
    - offset
        - 38h
    - PORT 0 RX DSCP PRIORITY TO RX PACKET MAPPING REG 2
- P0_RX_DSCP_PRI_MAP3
    - offset
        - 3Ch
    - PORT 0 RX DSCP PRIORITY TO RX PACKET MAPPING REG 3
- P0_RX_DSCP_PRI_MAP4
    - offset
        - 40h
    - PORT 0 RX DSCP PRIORITY TO RX PACKET MAPPING REG 4
- P0_RX_DSCP_PRI_MAP5
    - offset
        - 44h
    - PORT 0 RX DSCP PRIORITY TO RX PACKET MAPPING REG 5
- P0_RX_DSCP_PRI_MAP6
    - offset
        - 48h
    - PORT 0 RX DSCP PRIORITY TO RX PACKET MAPPING REG 6
- P0_RX_DSCP_PRI_MAP7
    - offset
        - 4Ch
    - PORT 0 RX DSCP PRIORITY TO RX PACKET MAPPING REG 7
- P1_CONTROL
    - offset
        - 100h
    - Port 1 control registe
        - enable disable vlan
        - confing priority

        - enable disable time sync
- P1_MAX_BLKS
    - offset
        - 108h
    - Port 1 maximum fifo blocks register
        - set the maximum 1kb blocks for fifo
        - keep as default
- P1_BLK_CNT
    - offset
        - 10Ch
    - Port 1 fifo block usage count
        - read the number of fifo blocks are in use
- P1_TX_IN_CTL
    - offset
        - 110h
    - Port 1 transmit fifo control
        - select queue type
- P1_PORT_VLAN
    - offset
        - 114h
    - Port 1 vlan register
        - vlan config for the port
- P1_TX_PRI_MAP
    - offset
        - 118h
    - PORT 1 TX HEADER PRIORITY TO SWITCH PRI MAPPING REGISTER
        - keep default
- P1_TS_SEQ_MTYPE
    - offset
        - 11Ch
    - PORT 1 TIME SYNC SEQUENCE ID OFFSET AND MSG TYPE.
        - time sync sequence id
- P1_SA_LO
    - offset

        - 120h
    - CPGMAC_SL1 SOURCE ADDRESS LOW REGISTER
        - use to set mac addr of port
- P1_SA_HI
    - offset
        - 124h
    - CPGMAC_SL1 SOURCE ADDRESS HIGH REGISTER
        - use to set mac addr of port
- P1_SEND_PERCENT
    - offset
        - 128h
    - PORT 1 TRANSMIT QUEUE SEND PERCENTAGES
        - set transmit percentage of each priority queue
- P1_RX_DSCP_PRI_MAP0
    - offset
        - 130h
    - PORT 1 RX DSCP PRIORITY TO RX PACKET MAPPING REG 0
- P1_RX_DSCP_PRI_MAP1
    - offset
        - 134h
    - PORT 1 RX DSCP PRIORITY TO RX PACKET MAPPING REG 1
- P1_RX_DSCP_PRI_MAP2
    - offset
        - 138h
    - PORT 1 RX DSCP PRIORITY TO RX PACKET MAPPING REG 2
- P1_RX_DSCP_PRI_MAP3
    - offset
        - 13Ch
    - PORT 1 RX DSCP PRIORITY TO RX PACKET MAPPING REG 3
- P1_RX_DSCP_PRI_MAP4
    - offset
        - 140h
    - PORT 1 RX DSCP PRIORITY TO RX PACKET MAPPING REG 4
- P1_RX_DSCP_PRI_MAP5
    - offset

        - 144h
    - PORT 1 RX DSCP PRIORITY TO RX PACKET MAPPING REG 5
- P1_RX_DSCP_PRI_MAP6
    - offset
        - 148h
    - PORT 1 RX DSCP PRIORITY TO RX PACKET MAPPING REG 6
- P1_RX_DSCP_PRI_MAP7
    - offset
        - 14Ch
    - PORT 1 RX DSCP PRIORITY TO RX PACKET MAPPING REG 7
- P2_CONTROL
    - offset
        - 200h
    - PORT 2 CONTROL REGISTER
        - enable disable vlan
        - config priority
        - enable disable time sync
- P2_MAX_BLKS
    - offset
        - 208h
    - PORT 2 MAXIMUM FIFO BLOCKS REGISTER
        - set the maximum 1kb blocks for fifo
        - keep as default
- P2_BLK_CNT
    - offset
        - 20Ch
    - PORT 2 FIFO BLOCK USAGE COUNT
        - read the number of fifo blocks are in use
- P2_TX_IN_CTL
    - offset
        - 210h
    - PORT 2 TRANSMIT FIFO CONTROL
        - select queue type
- P2_PORT_VLAN
    - offset

        - 214h
    - PORT 2 VLAN REGISTER
        - vlan config for the port
- P2_TX_PRI_MAP
    - offset
        - 218h
    - PORT 2 TX HEADER PRIORITY TO SWITCH PRI MAPPING REGISTER
        - keep default
- P2_TS_SEQ_MTYPE
    - offset
        - 21Ch
    - PORT 2 TIME SYNC SEQUENCE ID OFFSET AND MSG TYPE.
        - time sync sequence id
- P2_SA_LO
    - offset
        - 220h
    - CPGMAC_SL2 SOURCE ADDRESS LOW REGISTER
        - use to set MAC addr
- P2_SA_HI
    - offset
        - 224h
    - CPGMAC_SL2 SOURCE ADDRESS HIGH REGISTER
        - use to set MAC addr
- P2_SEND_PERCENT
    - offset
        - 228h
    - PORT 2 TRANSMIT QUEUE SEND PERCENTAGES
        - set transmit percentage of each priority queue
- P2_RX_DSCP_PRI_MAP0
    - offset
        - 230h
    - PORT 2 RX DSCP PRIORITY TO RX PACKET MAPPING REG 0
- P2_RX_DSCP_PRI_MAP1
    - offset

        - 234h
    - PORT 2 RX DSCP PRIORITY TO RX PACKET MAPPING REG 1
- P2_RX_DSCP_PRI_MAP2
    - offset
        - 238h
    - PORT 2 RX DSCP PRIORITY TO RX PACKET MAPPING REG 2
- P2_RX_DSCP_PRI_MAP3
    - offset
        - 23Ch
    - PORT 2 RX DSCP PRIORITY TO RX PACKET MAPPING REG 3
- P1_RX_DSCP_PRI_MAP4
    - offset
        - 240h
    - PORT 2 RX DSCP PRIORITY TO RX PACKET MAPPING REG 4
- P1_RX_DSCP_PRI_MAP5
    - offset
        - 244h
    - PORT 2 RX DSCP PRIORITY TO RX PACKET MAPPING REG 5
- P1_RX_DSCP_PRI_MAP6
    - offset
        - 248h
    - PORT 2 RX DSCP PRIORITY TO RX PACKET MAPPING REG 6
- P1_RX_DSCP_PRI_MAP7
    - offset
        - 24Ch
    - PORT 2 RX DSCP PRIORITY TO RX PACKET MAPPING REG 7

CPSW_SL

- base address
    - port 1
        - 0x4A10_0D80
    - port 2
        - 0x4A10_0DC0
- IDVER

    - offset
        - 0h
    - CPGMAC_SL ID/VERSION REGISTER
        - Rx id numbers
- MACCONTROL
    - offset
        - 4h
    - CPGMAC_SL MAC CONTROL REGISTER
        - enable what frames are transferred to memory
        - modes of operation
        - GMII enable
- MACSTATUS
    - offset
        - 8h
    - CPGMAC_SL MAC STATUS REGISTER
        - various status flags
- SOFT_RESET
    - offset
        - Ch
    - CPGMAC_SL SOFT RESET REGISTER
        - used to initiate software reset
- RX_MAXLEN
    - offset
        - 10h
    - CPGMAC_SL RX MAXIMUM LENGTH REGISTER
        - maximum length of a received frame
- BOFFTEST
    - offset
        - 14h
    - CPGMAC_SL BACKOFF TEST REGISTER
        - contains bit to see number of collisions the current frame has experienced
- RX_PAUSE
    - offset
        - 18h
    - CPGMAC_SL RECEIVE PAUSE TIMER REGISTER

        - no clue
- TX_PAUSE
    - offset
        - 1Ch
    - CPGMAC_SL TRANSMIT PAUSE TIMER REGISTER
        - no clue
- EMCONTROL
    - offset
        - 20h
    - CPGMAC_SL EMULATION CONTROL REGISTER
        - enabling emulation
- RX_PRI_MAP
    - offset
        - 24h
    - CPGMAC_SL RX PKT PRIORITY TO HEADER PRIORITY MAPPING REGISTER
        - keep default
- TX_GAP
    - offset
        - 28h
    - TRANSMIT INTER-PACKET GAP REGISTER
        - no clue
        - keep default

CPSW_SS

- base address
    - 0x4A10_0000
- ID_VER
    - offset
        - 0h
    - ID VERSION REGISTER
        - 3g identification values
- CONTROL
    - offset

        - 4h
    - SWITCH CONTROL REGISTER
        - set vlan and loopback mdoe
- SOFT_RESET
    - offset
        - 8h
    - SOFT RESET REGISTER
        - use to software reset switch
- STATE_PORT_EN
    - offset
        - Ch
    - STATISTICS PORT ENABLE REGISTER
        - enable statistics on ports
- PTYPE
    - offset
        - 10h
    - TRANSMIT PRIORITY TYPE REGISTER
        - enabling transmit shape on ports
        - enabling priority escalate on ports
- SOFT_IDLE
    - offset
        - 14h
    - SOFTWARE IDLE
        - uses this to stop forwarding packets
- THRU_RATE
    - offset
        - 18h
    - THROUGHPUT RATE
        - set maximum throughput of ethernet ports
- GAP_THRESH
    - offset
        - 1Ch
    - CPGMAC_SL SHORT GAP THRESHOLD
        - short gap threshold

- TX_START_WDS
  - offset
    - 20h
  - TRANSMIT START WORDS
    - recommended decimal 32
- FLOW_CONTROL
  - offset
    - 24h
  - FLOW CONTROL
    - enable flow control on ports
- VLAN_LTYPE
  - offset
    - 28h
  - LTYPE1 AND LTYPE 2 REGISTER
- TS_LTYPE
  - offset
    - 2Ch
  - VLAN_LTYPE1 AND VLAN_LTYPE2 REGISTER
- DLR_LTYPE
  - offset
    - 30h
  - DLR LTYPE REGISTER

CPSW_WR

- base address
  - 0x4A10_1200
- IDVER
  - offset
    - 0h
  - SUBSYSTEM ID VERSION REGISTER
    - subsystem version
- SOFT_RESET
  - offset
    - 4h

    - SUBSYSTEM SOFT RESET REGISTER
        - software rest the sub system
- CONTROL
    - offset
        - 8h
    - SUBSYSTEM CONTROL REGISTER
        - set idle and standby state
- INT_CONTROL
    - offset
        - Ch
    - SUBSYSTEM INTERRUPT CONTROL
        - for interrupt pacing
- C0_RX_THRESH_EN
    - offset
        - 10h
    - SUBSYSTEM CORE 0 RECEIVE THRESHOLD INT ENABLE REGISTER
        - enable receive threshold
- C0_RX_EN
    - offset
        - 14h
    - SUBSYSTEM CORE 0 RECEIVE INTERRUPT ENABLE REGISTER
        - enable receive interrupt
- C0_TX_EN
    - offset
        - 18h
    - SUBSYSTEM CORE 0 TRANSMIT INTERRUPT ENABLE REGISTER
        - enable transmit interrupt
- C0_MISC_EN
    - offset
        - 1Ch
    - SUBSYSTEM CORE 0 MISC INTERRUPT ENABLE REGISTER
        - enable misc. interrupts
- C1_RX_THRESH_EN
    - offset
        - 20h

    - SUBSYSTEM CORE 1 RECEIVE THRESHOLD INT ENABLE REGISTER
- C1_RX_EN
    - offset
        - 24h
    - SUBSYSTEM CORE 1 RECEIVE INTERRUPT ENABLE REGISTER
- C1_TX_EN
    - offset
        - 28h
    - SUBSYSTEM CORE 1 TRANSMIT INTERRUPT ENABLE REGISTER
- C1_MISC_EN
    - offset
        - 2Ch
    - SUBSYSTEM CORE 1 MISC INTERRUPT ENABLE REGISTER
- C2_RX_THRESH_EN
    - offset
        - 30h
    - SUBSYSTEM CORE 2 RECEIVE THRESHOLD INT ENABLE REGISTER
- C2_RX_EN
    - offset
        - 34h
    - SUBSYSTEM CORE 2 RECEIVE INTERRUPT ENABLE REGISTER
- C2_TX_EN
    - offset
        - 38h
    - SUBSYSTEM CORE 2 TRANSMIT INTERRUPT ENABLE REGISTER
- C2_MISC_EN
    - offset
        - 3Ch
    - SUBSYSTEM CORE 2 MISC INTERRUPT ENABLE REGISTER
- C0_RX_THRESH_STAT
    - offset
        - 40h
    - SUBSYSTEM CORE 0 RX THRESHOLD MASKED INT STATUS REGISTER
        - interrupt status for receive threshold interrupt
- C0_RX_STAT

    - offset
        - 44h
    - SUBSYSTEM CORE 0 RX INTERRUPT MASKED INT STATUS REGISTER
        - receive interrupt status
- C0_TX_STAT
    - offset
        - 48h
    - SUBSYSTEM CORE 0 TX INTERRUPT MASKED INT STATUS REGISTER
        - transmit interrupt status
- C0_MISC_STAT
    - offset
        - 4Ch
    - SUBSYSTEM CORE 0 MISC INTERRUPT MASKED INT STATUS REGISTER
        - misc interrupt status
- C1_RX_THRESH_STAT
    - offset
        - 50h
    - SUBSYSTEM CORE 1 RX THRESHOLD MASKED INT STATUS REGISTER
- C1_RX_STAT
    - offset
        - 54h
    - SUBSYSTEM CORE 1 RECEIVE MASKED INTERRUPT STATUS REGISTER
- C1_TX_STAT
    - offset
        - 58h
    - SUBSYSTEM CORE 1 TRANSMIT MASKED INTERRUPT STATUS REGISTER
- C1_MISC_STAT
    - offset
        - 5Ch
    - SUBSYSTEM CORE 1 MISC MASKED INTERRUPT STATUS REGISTER
- C2_RX_THRESH_STAT
    - offset
        - 60h
    - SUBSYSTEM CORE 2 RX THRESHOLD MASKED INT STATUS REGISTER
- C2_RX_STAT

    - offset
        - 64h
    - SUBSYSTEM CORE 2 RECEIVE MASKED INTERRUPT STATUS REGISTER
- C2_TX_STAT
    - offset
        - 68h
    - SUBSYSTEM CORE 2 TRANSMIT MASKED INTERRUPT STATUS REGISTER
- C2_MISC_STAT
    - offset
        - 6Ch
    - SUBSYSTEM CORE 2 MISC MASKED INTERRUPT STATUS REGISTER
- C0_RX_IMAX
    - offset
        - 70h
    - SUBSYSTEM CORE 0 RECEIVE INTERRUPTS PER MILLISECOND
        - set receive interrupts per millisecond
        - if pacing is enabled
- C0_TX_IMAX
    - offset
        - 74h
    - SUBSYSTEM CORE 0 TRANSMIT INTERRUPTS PER MILLISECOND
        - set transmit interrupts per millisecond
        - if pacing is enabled
- C1_RX_IMAX
    - offset
        - 78h
    - SUBSYSTEM CORE 1 RECEIVE INTERRUPTS PER MILLISECOND
- C1_TX_IMAX
    - offset
        - 7Ch
    - SUBSYSTEM CORE 1 TRANSMIT INTERRUPTS PER MILLISECOND
- C2_RX_IMAX
    - offset
        - 80h

        - SUBSYSTEM CORE 2 RECEIVE INTERRUPTS PER MILLISECOND
    - C2_TX_IMAX
        - offset
            - 84h
        - SUBSYSTEM CORE 2 TRANSMIT INTERRUPTS PER MILLISECOND
    - RGMII_CTL
        - offset
            - 88h
        - RGMII CONTROL SIGNAL REGISTER
            - config if in RGMII mode

MDIO Registers

- base address
    - 0x4A10_1000
- MDIOVER
    - offset
        - 0h
    - MDIO Version Register
        - mdio version and id of peripheral
- MDIOCONTROL
    - offset
        - 4h
    - MDIO Control Register
        - check if state machine is idle
        - enable mdio
        - enable fault detection
        -
- MDIOALIVE
    - offset
        - 8h
    - PHY Alive Status Register
        - Each of the 32 bits of this register is set if the most recent access to the PHY with address corresponding to the register bit number was acknowledged by the PHY

- MDIOLINK
  - offset
    - Ch
  - PHY Link Status Register
    - This register is updated after a read of the Generic Status Register of a PHY. The bit is set if the PHY with the corresponding address has link and the PHY acknowledges the read transaction
- MDIOLINKINTRAW
  - offset
    - 10h
  - MDIO Link Status Change Interrupt Register
- MDIOLINKINTMASKED
  - offset
    - 14h
  - MDIO Link Status Change Interrupt Register (Masked Value)
- MDIOUSERINTRAW
  - offset
    - 20h
  - MDIO User Command Complete Interrupt Register (Raw Value)
- MDIOUSERINTMASKED
  - offset
    - 24h
  - MDIO User Command Complete Interrupt Register (Masked Value)
- MDIOUSERINTMASKSET
  - offset
    - 28h
  - MDIO User Command Complete Interrupt Mask Set Register
    - enable user commands interrupt
- MDIOUSERINTMASKCLR
  - offset
    - 2Ch
  - MDIO User Interrupt Mask Clear Register
    - disable user commands interrupt
- MDIOUSERACCESS0
  - offset

        - 80h
    - MDIO User Access Register 0
        - cmd register
- MDIOUSERPHYSEL0
    - offset
        - 84h
    - MDIO User PHY Select Register 0
- MDIOUSERACCESS1
    - offset
        - 88h
    - MDIO User Access Register 1
        - cmd register
- MDIOUSERPHYSEL1
    - offset
        - 8Ch
    - MDIO User PHY Select Register 1

CONTROL_MODULE

- base address
    - 0x44E1_0000
- GMII_SEL
    - offset
        - 650h
    - set interface for ports
    -
- MAC_ID0_LO
    - offset
        - 630h
    -
- MAC_ID0_HI
    - offset
        - 634h
    -
- MAC_ID1_LO

    - offset
        - 638h
    -
- MAC_ID1_HI
    - offset
        - 63Ch
    -

CLOCK_MODULE

- base address
    - 0x44E0_0000
- CM_PER_CPGMAC0_CLKCTRL
    - offset
        - 14h
    - This register manages the CPSW clocks
        - enable clock module
- CM_PER_CPSW_CLKSTCTRL
    - offset
        - 144h
    - This register enables the clock domain state transition. It controls the SW supervised clock domain state transition between ON-ACTIVE and ON-INACTIVE states. It also hold one status bit per clock input of the domain.
        - used to wake up clock
        - check clock state

PIN_REGS

- base address
    - 0x44E1_0000
- conf_mii1_col
    - 908h
- conf_mii1_crs
    - 90Ch
- conf_mii1_rx_er
    - 910h

- conf_mii1_tx_en
  - 914h
- conf_mii1_rx_dv
  - 918h
- conf_mii1_txd3
  - 91Ch
- conf_mii1_txd2
  - 920h
- conf_mii1_txd1
  - 924h
- conf_mii1_txd0
  - 928h
- conf_mii1_tx_clk
  - 92Ch
- conf_mii1_rx_clk
  - 930h
- conf_mii1_rxd3
  - 934h
- conf_mii1_rxd2
  - 938h
- conf_mii1_rxd1
  - 93Ch
- conf_mii1_rxd0
  - 940h
- conf_rmii1_ref_clk
  - 944h
- conf_mdio
  - 948h
- conf_mdc
  - 94Ch

Driver Design

Initialization

Select interface

Table 9-40. gmii_sel Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	Reserved	R	0h	
7	rmii2_io_clk_en	R/W	1h	0: RMII Reference Clock Output mode. Enable RMII clock to be sourced from PLL. 1: RMII Reference Clock Input mode. Enable RMII clock to be sourced from chip pin. See "Silicon Revision Functional Differences and Enhancements" for differences in operation based on AM335x silicon revision.
6	rmii1_io_clk_en	R/W	1h	0: RMII Reference Clock Output mode. Enable RMII clock to be sourced from PLL 1: RMII Reference Clock Input mode. Enable RMII clock to be sourced from chip pin See "Silicon Revision Functional Differences and Enhancements" for differences in operation based on AM335x silicon revision.
5	rgmii2_idmode	R/W	1h	RGMII2 Internal Delay Mode 0: Reserved 1: No Internal Delay See "Silicon Revision Functional Differences and Enhancements" for differences in operation based on AM335x silicon revision.
4	rgmii1_idmode	R/W	1h	RGMII1 Internal Delay Mode 0: Reserved 1: No Internal Delay See "Silicon Revision Functional Differences and Enhancements" for differences in operation based on AM335x silicon revision.
3-2	gmii2_sel	R/W	0h	00: Port2 GMII/MII Mode 01: Port2 RMII Mode 10: Port2 RGMII Mode 11: Not Used
1-0	gmii1_sel	R/W	0h	00: Port1 GMII/MII Mode 01: Port1 RMII Mode 10: Port1 RGMII Mode 11: Not Used

-
- config gmii_sel register
  - gmii/mii mode for port 1 and 2
  - set all bits to 0
  - reg = 0x0
- using gmii/mii interface

Pin Muxing

- https://github.com/beagleboard/linux/blob/master/arch/arm/boot/dts/ti/omap/am335x-phycore-som.dtsi

```
/* Ethernet */
&am33xx_pinmux {
        ethernet0_pins: ethernet0-pins {
                pinctrl-single,pins = <
                        AM33XX_PADCONF(AM335X_PIN_MII1_CRS, PIN_INPUT_PULLDOWN, MUX_MODE1)
                        AM33XX_PADCONF(AM335X_PIN_MII1_RX_ER, PIN_INPUT_PULLDOWN, MUX_MODE1)
                        AM33XX_PADCONF(AM335X_PIN_MII1_TX_EN, PIN_OUTPUT, MUX_MODE1)
                        AM33XX_PADCONF(AM335X_PIN_MII1_TXD1, PIN_OUTPUT, MUX_MODE1)
                        AM33XX_PADCONF(AM335X_PIN_MII1_TXD0, PIN_OUTPUT, MUX_MODE1)
                        AM33XX_PADCONF(AM335X_PIN_MII1_RXD1, PIN_INPUT_PULLDOWN, MUX_MODE1)
                        AM33XX_PADCONF(AM335X_PIN_MII1_RXD0, PIN_INPUT_PULLDOWN, MUX_MODE1)
                        AM33XX_PADCONF(AM335X_PIN_RMII1_REF_CLK, PIN_INPUT_PULLDOWN, MUX_MODE0)
                >;
        };
```

Table 9-60. conf_<module>_<pin> Register Field Descriptions

Bit	Field	Type	Reset	Description
31-20	Reserved	R	0h	
19-7	Reserved	R	0h	
6	conf_<module>_<pin>_slewctrl	R/W	X	Select between faster or slower slew rate 0: Fast 1: Slow Reset value is pad-dependent.
5	conf_<module>_<pin>_rxactive	R/W	1h	Input enable value for the PAD 0: Receiver disabled 1: Receiver enabled
4	conf_<module>_<pin>_putypesel	R/W	X	Pad pullup/pulldown type selection 0: Pulldown selected 1: Pullup selected Reset value is pad-dependent.
3	conf_<module>_<pin>_puden	R/W	X	Pad pullup/pulldown enable 0: Pullup/pulldown enabled 1: Pullup/pulldown disabled Reset value is pad-dependent.
2-0	conf_<module>_<pin>_mmode	R/W	X	Pad functional signal mux select. Reset value is pad-dependent.

- base set all to 0
- set bit 5 to enable receive
- set bit 4 to enable pullup
- default = set all bits to 0
- Pins
  - conf_mii1_col
    - default
    - set bit 5
  - conf_mii1_crs
    - default
    - set bit 5
  - conf_mii1_rx_er

    - default
    - set bit 5
- conf_mii1_tx_en
    - default
- conf_mii1_rx_dv
    - default
    - set bit 5
- conf_mii1_txd3
    - default
- conf_mii1_txd2
    - default
- conf_mii1_txd1
    - default
- conf_mii1_txd0
    - default
- conf_mii1_tx_clk
    - default
    - set bit 5
- conf_mii1_rx_clk
    - default
    - set bit 5
- conf_mii1_rxd3
    - default
    - set bit 5
- conf_mii1_rxd2
    - default
    - set bit 5
- conf_mii1_rxd1
    - default
    - set bit 5
- conf_mii1_rxd0
    - default
    - set bit 5
- conf_rmii1_ref_clk

    - keep at reset value
- conf_mdio
    - default
    - set bit 5
    - set bit 4
- conf_mdc
    - default
    - set bit 4

Enable and Configure Clocks

Table 8-34. CM_PER_CPGMAC0_CLKCTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-19	Reserved	R	0h	
18	STBYST	R	1h	Module standby status. This bit is warm reset insensitive when CPSW RESET_ISO is enabled 0x0 = Func : Module is functional (not in standby) 0x1 = Standby : Module is in standby
17-16	IDLEST	R	3h	Module idle status. This bit is warm reset insensitive when CPSW RESET_ISO is enabled 0x0 = Func : Module is fully functional, including OCP 0x1 = Trans : Module is performing transition: wakeup, or sleep, or sleep abortion 0x2 = Idle : Module is in Idle mode (only OCP part). It is functional if using separate functional clock 0x3 = Disable : Module is disabled and cannot be accessed
15-2	Reserved	R	0h	
1-0	MODULEMODE	R/W	0h	Control the way mandatory clocks are managed. This bit is warm reset insensitive when CPSW RESET_ISO is enabled 0x0 = DISABLED : Module is disable by SW. Any OCP access to module results in an error, except if resulting from a module wakeup (asynchronous wakeup). 0x1 = RESERVED_1 : Reserved 0x2 = ENABLE : Module is explicitly enabled. Interface clock (if not used for functions) may be gated according to the clock domain state. Functional clocks are guarantied to stay present. As long as in this configuration, power domain sleep transition cannot happen. 0x3 = RESERVED : Reserved

-
- CM_PER_CPGMAC0_CLKCTRL
    - enable module
    - set bit 0-1 to 0x2
    - wait until module is functional by reading bit 16-17 and waiting for it to equal 0

Table 8-85. CM_PER_CPSW_CLKSTCTRL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-5	Reserved	R	0h	
4	CLKACTIVITY_CPSW_125MHz_GCLK	R	0h	This field indicates the state of the CPSW 125 MHz OCP clock in the domain. 0x0 = Inact 0x1 = Act
3-2	Reserved	R	0h	
1-0	CLKTRCTRL	R/W	2h	Controls the clock state transition of the CPSW OCP clock domain. 0x0 = NO_SLEEP : NO_SLEEP: Sleep transition cannot be initiated. Wakeup transition may however occur. 0x1 = SW_SLEEP : SW_SLEEP: Start a software forced sleep transition on the domain. 0x2 = SW_WKUP : SW_WKUP: Start a software forced wake-up transition on the domain. 0x3 = Reserved : Reserved.

- CM_PER_CPSW_CLKSTCTRL
  - wake up clocks
  - set bit 0-1 to 0x2
  - wait for clocks to be active be reading bit 4 until it equals 1

Software reset

- reset each module that can be reset
- CPDMA_SOFT_RESET

Bit	Field	Type	Reset	Description
31-1	Reserved	R	0h	
0	SOFT_RESET	R/W	0h	Software reset - Writing a one to this bit causes the CPDMA logic to be reset. Software reset occurs when the RX and TX DMA Controllers are in an idle state to avoid locking up the VBUSP bus. After writing a one to this bit, it may be polled to determine if the reset has occurred. If a one is read, the reset has not yet occurred. If a zero is read then reset has occurred.

  - write 1 to bit 0
  - when 0 is read back from bit 0 then reset is done
- CSPW_SL SOFT_RESET * 2

Bit	Field	Type	Reset	Description
31-1	RESERVED	Rreturns0s	0h	
0	SOFT_RESET	R/W	0h	Software reset - Writing a one to this bit causes the CPGMAC_SL logic to be reset. After writing a one to this bit, it may be polled to determine if the reset has occurred. If a one is read, the reset has not yet occurred. If a zero is read then reset has occurred.

  - write 1 to bit 0
  - when 0 is read back from bit 0 then reset is done
- CPSW_SS SOFT_RESET

Bit	Field	Type	Reset	Description
0	SOFT_RESET	R/W-0	0	Software reset - Writing a one to this bit causes the 3G logic to be reset. After writing a one to this bit, it may be polled to determine if the reset has occurred. If a one is read, the reset has not yet occurred. If a zero is read then reset has occurred.

- write 1 to bit 0
- when 0 is read back from bit 0 then reset is done

- CPSW_WR SOFT_RESET

Bit	Field	Type	Reset	Description
31-1	Reserved	R	0h	
0	SOFT_RESET	R/W	0h	Software reset - Writing a one to this bit causes the CPGMACSS_R logic to be reset (INT, REGS, CPPI). Software reset occurs on the clock following the register bit write.

- write 1 to bit 0
- read bit 0 until it is not 1 anymore

Initializing CPDMA Pointers

- this should be done after CPDMA software reset so maybe reset CPDMA last in software reset section
- TX HDP

Table 14-102. TX0_HDP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	TX_HDP	R/W	0h	TX Channel (0..7) DMA Head Descriptor Pointer - Writing a TX DMA Buffer Descriptor address to a head pointer location initiates TX DMA operations in the queue for the selected channel. Writing to these locations when they are non-zero is an error (except at reset). Host software must initialize these locations to zero on reset.

- TX CP

Table 14-118. TX0_CP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	TX_CP	R/W	0h	Tx Completion Pointer Register - This register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing. The port uses the value written to determine if the interrupt should be deasserted.

- RX HDP

Table 14-110. RX0_HDP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RX_HDP	R/W	0h	RX DMA Head Descriptor Pointer - Writing an RX DMA Buffer Descriptor address to this location allows RX DMA operations in the selected channel when a channel frame is received. Writing to these locations when they are non-zero is an error (except at reset). Host software must initialize these locations to zero on reset.

- RX CP

Table 14-126. RX0_CP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RX_CP	R/W	0h	Rx Completion Pointer Register - This register is written by the host with the buffer descriptor address for the last buffer processed by the host during interrupt processing. The port uses the value written to determine if the interrupt should be deasserted. Note: The value read is the completion pointer (interrupt acknowledge) value that was written by the CPDMA DMA controller (port). The value written to this register by the host is compared with the value that the port wrote to determine if the interrupt should remain asserted. The value written is not actually stored in the location. The interrupt is deasserted if the two values are equal.

Configure control register

Table 14-201. CONTROL Register Field Descriptions

Bit	Field	Type	Reset	Description
3	DLR_EN	R/W-0	0	DLR enable 0 - DLR is disabled. DLR packets will not be moved to queue priority 3 and will not be separated out onto dlr_cpdma_ch. 1 - DLR is disabled. DLR packets be moved to destination port transmit queue priority 3 and will be separated out onto dlr_cpdma_ch when packet is to egress on port 0.
2	RX_VLAN_ENCAP	R/W-0	0	Port 0 VLAN Encapsulation (egress): 0 - Port 0 receive packets (from 3G) are not VLAN encapsulated. 1 - Port 0 receive packets (from 3G) are VLAN encapsulated.
1	VLAN_AWARE	R/W-0	0	VLAN Aware Mode: 0 - 3G is in the VLAN unaware mode. 1 - 3G is in the VLAN aware mode.
0	FIFO_LOOPBACK	R/W-0	0	FIFO Loopback Mode 0 - Loopback is disabled 1 - FIFO Loopback mode enabled. Each packet received is turned around and sent out on the same port's transmit path. Port 2 receive is fixed on channel zero. The RXSOFOVERRUN statistic will increment for every packet sent in FIFO loopback mode.

- at reset everything we ant is set to the right thing so no need to do anything

Configuring the ALE

- ADDRESS LOOKUP ENGINE CONTROL REGISTER

Table 14-25. CONTROL Register Field Descriptions

Bit	Field	Type	Reset	Description
31	ENABLE_ALE	R/W-0	0	Enable ALE - 0 - Drop all packets 1 - Enable ALE packet processing
30	CLEAR_TABLE	R/W-0	0	Clear ALE address table - Setting this bit causes the ALE hardware to write all table bit values to zero. Software must perform a clear table operation as part of the ALE setup/configuration process. Setting this bit causes all ALE accesses to be held up for 64 clocks while the clear is performed. Access to all ALE registers will be blocked (wait states) until the 64 clocks have completed. This bit cannot be read as one because the read is blocked until the clear table is completed at which time this bit is cleared to zero.
29	AGE_OUT_NOW	R/W-0	0	Age Out Address Table Now - Setting this bit causes the ALE hardware to remove (free up) any ageable table entry that does not have a set touch bit. This bit is cleared when the age out process has completed. This bit may be read. The age out process takes 4096 clocks best case (no ale packet processing during ageout) and 66550 clocks absolute worst case.
8	EN_P0_UNI_FLOOD	R/W-0	0	Enable Port 0 (Host Port) unicast flood 0 - do not flood unknown unicast packets to host port (p0) 1 - flood unknown unicast packets to host port (p0)
7	LEARN_NO_VID	R/W-0	0	Learn No VID - 0 - VID is learned with the source address 1 - VID is not learned with the source address (source address is not tied to VID).
6	EN_VID0_MODE	R/W-0	0	Enable VLAN ID = 0 Mode 0 - Process the packet with VID = PORT_VLAN[11:0]. 1 - Process the packet with VID = 0.
5	ENABLE_OUI_DENY	R/W-0	0	Enable OUI Deny Mode - When set this bit indicates that a packet with a non OUI table entry matching source address will be dropped to the host unless the destination address matches a multicast table entry with the super bit set.

Table 14-25. CONTROL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
4	BYPASS	R/W-0	0	ALE Bypass - When set, all packets received on ports 0 and 1 are sent to the host (only to the host).
3	RATE_LIMIT_TX	R/W-0	0	Rate Limit Transmit mode - 0 - Broadcast and multicast rate limit counters are received port based 1 - Broadcast and multicast rate limit counters are transmit port based.
2	VLAN_AWARE	R/W-0	0	ALE VLAN Aware - Determines what is done if VLAN not found. 0 - Flood if VLAN not found 1 - Drop packet if VLAN not found
1	ENABLE_AUTH_MODE	R/W-0	0	Enable MAC Authorization Mode - Mac authorization mode requires that all table entries be made by the host software. There are no learned address in authorization mode and the packet will be dropped if the source address is not found (and the destination address is not a multicast address with the super table entry bit set). 0 - The ALE is not in MAC authorization mode 1 - The ALE is in MAC authorization mode
0	ENABLE_RATE_LIMIT	R/W-0	0	Enable Broadcast and Multicast Rate Limit 0 - Broadcast/Multicast rates not limited 1 - Broadcast/Multicast packet reception limited to the port control register rate limit fields.

- set bit 31
- set bit 30

- enable and clear ALE

Configuring the MDIO

- Configure the PREAMBLE and CLKDIV bits in the MDIO control register
- Enable the MDIO module by setting the ENABLE bit in MDIOCONTROL
- The MDIO PHY alive status register (MDIOALIVE) can be read in polling fashion until a PHY connected to the system responded, and the MDIO PHY link status register (MDIOLINK) can determine whether this PHY already has a link

Table 14-251. MDIOCONTROL Register Field Descriptions

Bit	Field	Type	Reset	Description
31	IDLE	R	1	MDIO state machine IDLE. Set to 1 when the state machine is in the idle state. 0 = State machine is not in idle state. 1 = State machine is in idle state.
30	ENABLE	R/W	0	Enable control. If the MDIO state machine is active at the time it is disabled, it will complete the current operation before halting and setting the idle bit. If using byte access, the enable bit has to be the last bit written in this register. 0 = Disables the MDIO state machine. 1 = Enable the MDIO state machine.
29	Reserved	R	0	Reserved.
28-24	HIGHEST_USER_CHANNEL	R	1	Highest user channel. This field specifies the highest user access channel that is available in the module and is currently set to 1. This implies that the MDIOUSERACCESS1 register is the highest available user access channel.
23-21	Reserved	R	0	Reserved.
20	PREAMBLE	R/W	0	Preamble disable. 0 = Standard MDIO preamble is used. 1 = Disables this device from sending MDIO frame preambles.
19	FAULT	R/WC	0	Fault indicator. This bit is set to 1 if the MDIO pins fail to read back what the device is driving onto them. This indicates a physical layer fault and the module state machine is reset. Writing a 1 to it clears this bit. 0 = No failure. 1 = Physical layer fault; the MDIO state machine is reset.
18	FAULTENB	R/W	0	Fault detect enable. This bit has to be set to 1 to enable the physical layer fault detection. 0 = Disables the physical layer fault detection. 1 = Enables the physical layer fault detection.
17	INTTESTENB	R/W	0	Interrupt test enable. This bit can be set to 1 to enable the host to set the USERINT and LINKINT bits for test purposes. 0 = Interrupt bits are not set. 1 = Enables the host to set the USERINT and LINKINT bits for test purposes.
16	Reserved	R	0	Reserved.
15-0	CLKDIV	R/W	255	Clock divider. This field specifies the division ratio between CLK and the frequency of MDIO_CLK. MDIO_CLK is disabled when clkdiv is set to 0. MDIO_CLK frequency = clk frequency/(clkdiv+1).

-
  - set bit 30
  - CLK div to 124
  - disable preamble
    - set bit 20 to 1
  - set bit 18 to 1

Table 14-252. MDIOALIVE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ALIVE	R/WC	0	MDIO alive. Each of the 32 bits of this register is set if the most recent access to the PHY with address corresponding to the register bit number was acknowledged by the PHY, the bit is reset if the PHY fails to acknowledge the access. Both the user and polling accesses to a PHY will cause the corresponding alive bit to be updated. The alive bits are only meant to be used to give an indication of the presence or not of a PHY with the corresponding address. Writing a 1 to any bit will clear it, writing a 0 has no effect.

-
    - read to see if PHY is connected

Configuring the Statistics Port Enable register

- configure STAT_PORT_EN

Table 14-203. STAT_PORT_EN Register Field Descriptions

Bit	Field	Type	Reset	Description
2	P2_STAT_EN	R/W-0	0	Port 2 (GMII2 and Port 2 FIFO) Statistics Enable 0 - Port 2 statistics are not enabled. 1 - Port 2 statistics are enabled.
1	P1_STAT_EN	R/W-0	0	Port 1 (GMII1 and Port 1 FIFO) Statistics Enable 0 - Port 1 statistics are not enabled. 1 - Port 1 statistics are enabled.
0	P0_STAT_EN	R/W-0	0	Port 0 Statistics Enable 0 - Port 0 statistics are not enabled 1 - Port 0 statistics are enabled. FIFO overruns (SOFOVERRUNS) are the only port 0 statistics that are enabled to be kept.

-
- set all to 0
- not enabling statistics right now

Configure Ports

- set port states in ALE
    - PORTCTL0
    - PORTCTL1
    - PORTCTL2

Table 14-32. PORTCTL0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	BCAST_LIMIT	R/W-0	0	Broadcast Packet Rate Limit - Each prescale pulse loads this field into the port broadcast rate limit counter. The port counters are decremented with each packet received or transmitted depending on whether the mode is transmit or receive. If the counters decrement to zero, then further packets are rate limited until the next prescale pulse. Broadcast rate limiting is enabled by a non-zero value in this field
23-16	MCAST_LIMIT	R/W-0	0	Multicast Packet Rate Limit - Each prescale pulse loads this field into the port multicast rate limit counter. The port counters are decremented with each packet received or transmitted depending on whether the mode is transmit or receive. If the counters decrement to zero, then further packets are rate limited until the next prescale pulse. Multicast rate limiting is enabled by a non-zero value in this field.
5	NO_SA_UPDATE	R/W-0	0	No Souce Address Update - When set the port is disabled from updating the source port number in an ALE table entry.
4	NO_LEARN	R/W-0	0	No Learn Mode - When set the port is disabled from learning an address.
3	VID_INGRESS_CHECK	R/W-0	0	VLAN ID Ingress Check - If VLAN not found then drop the packet. Packets with an unknown (default) VLAN will be dropped.
2	DROP_UNTAGGED	R/W-0	0	Drop Untagged Packets - Drop non-VLAN tagged ingress packets.
1-0	PORT_STATE	R/W-0	0	Port State 0 - Disabled 1 - Blocked 2 - Learn 3 - Forward

- set state to forward

- configure the ports for normal priority switch, VLAN unaware
  - set ALE entries for ports
    - TBLCTL

Table 14-28. TBLCTL Register Field Descriptions

Bit	Field	Type	Reset	Description
31	WRITE_RDZ	R/W-0	0	Write Bit - This bit is always read as zero. Writing a 1 to this bit causes the three table word register values to be written to the entry_pointer location in the address table. Writing a 0 to this bit causes the three table word register values to be loaded from the entry_pointer location in the address table so that they may be subsequently read. A read of any ALE address location will be stalled until the read or write has completed.
9-0	ENTRY_POINTER	R/W-0	0	Table Entry Pointer - The entry_pointer contains the table entry value that will be read/written with accesses to the table word registers.

    - TBLWn

Bit	Field	Type	Reset	Description
7-0	ENTRY71-64	R/W-0	0	Table entry bits 71:64

Table 14-30. TBLW1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ENTRY63_32	R/W	0h	Table entry bits 63:32

Table 14-31. TBLW0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ENTRY31_0	R/W	0h	Table entry bits 31:0

- get entry pointer
  - check which entry in table is free
  - write ALE index in first 10 bits
  - write 0 to read that index into the 3 TBLW registers
- multicast entry for ALE
  - ALE entry has 3 words

14.3.2.7.1.2 Multicast Address Table Entry

Table 14-11. Multicast Address Table Entry Bit Values

71:70	68:66	65	64	63:62	61:60	59:48	47:0
Reserved	Port Mask	Super	Reserved	Mcast Fwd State	Entry Type (01)	Reserved	Multicast Address

- unicast entry for ALE
  - ALE entry has 3 words

14.3.2.7.1.4 Unicast Address Table Entry

Table 14-13. Unicast Address Table Entry Bit Values

71:70	69	68	67:66	65	64	63:62	61:60	59:48	47:0
Reserved	DLR Unicast	Reserved	Port Number	Block	Secure	Unicast Type (00) or (X1)	Entry Type (01)	Reserved	Unicast Address

- set address for ports
  - Pn_SA_LO

Bit	Field	Type	Reset	Description
31-16	RESERVED	R/W	0h	
15-8	MACSRCADDR_7_0	R/W	0h	Source Address Lower 8 bits (byte 0)
7-0	MACSRCADDR_15_8	R/W	0h	Source Address bits 15:8 (byte 1)

  - Pn_SA_HI

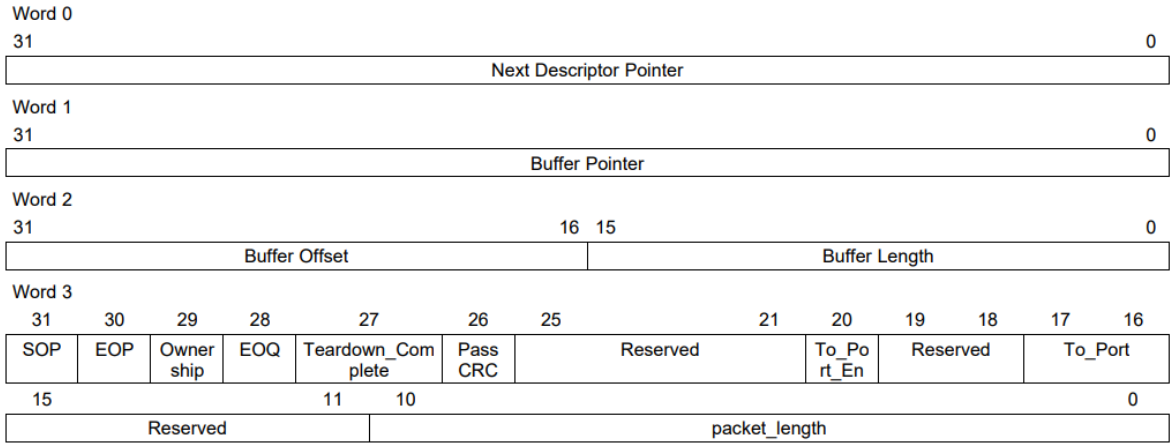
Bit	Field	Type	Reset	Description
31-24	MACSRCADDR_23_16	R/W	0h	Source Address bits 23:16 (byte 2)
23-16	MACSRCADDR_31_24	R/W	0h	Source Address bits 31:24 (byte 3)
15-8	MACSRCADDR_39_32	R/W	0h	Source Address bits 39:32 (byte 4)
7-0	MACSRCADDR_47_40	R/W	0h	Source Address bits 47:40 (byte 5)

Configure the CPPI Tx and Rx Descriptors

- need to setup DMA buffers following descriptor protocol
- buffers are located in CPPI ram
  - 4A10_2000
- half for TX and Half for RX
- TX

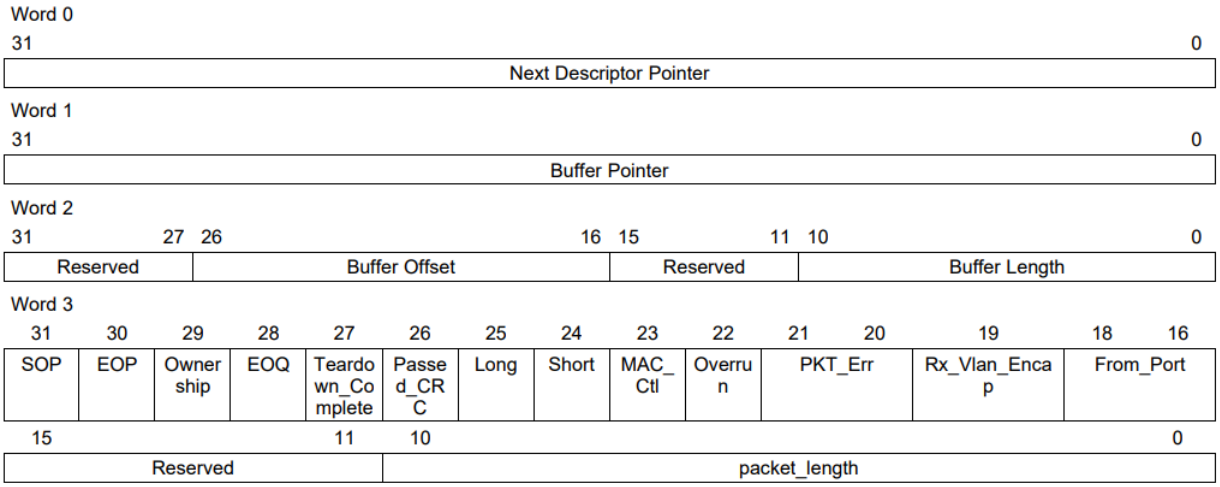
A TX buffer descriptor is a contiguous block of four 32-bit data words aligned on a 32-bit word boundary.

Figure 14-7. Tx Buffer Descriptor Format



- RX

Figure 14-8. Rx Buffer Descriptor Format



Configure the CPDMA receive DMA controller

- TX_CONTROL

Table 14-40. TX_CONTROL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	Reserved	R	0h	
0	TX_EN	R/W	0h	TX Enable 0 - Disabled 1 - Enabled

- enable TX

Configure the CPDMA transmit DMA controller

- RX_CONTROL

Table 14-43. RX_CONTROL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-1	Reserved	R	0h	
0	RX_EN	R/W	0h	RX DMA Enable 0 - Disabled 1 - Enabled

- enable RX

Configuring interrupts

- unmask interrupts
  - 41
    - RX DMA completion
    - clear bit 9
  - 42
    - TX DMA completion
    - clear bit 10
  - maybe 40
    - RX buffer threshold
      - when rx runs out of buffers but we prob wont be running out
- unmask interrupts for chosen channels
  - enable required channel interrupts for RX and TX by setting bits in
    - TX_INTMASK_SET

Table 14-60. TX_INTMASK_SET Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	Reserved	R	0h	
7	TX7_MASK	R/W	0h	TX Channel 7 Mask - Write one to enable interrupt.
6	TX6_MASK	R/W	0h	TX Channel 6 Mask - Write one to enable interrupt.
5	TX5_MASK	R/W	0h	TX Channel 5 Mask - Write one to enable interrupt.
4	TX4_MASK	R/W	0h	TX Channel 4 Mask - Write one to enable interrupt.
3	TX3_MASK	R/W	0h	TX Channel 3 Mask - Write one to enable interrupt.
2	TX2_MASK	R/W	0h	TX Channel 2 Mask - Write one to enable interrupt.
1	TX1_MASK	R/W	0h	TX Channel 1 Mask - Write one to enable interrupt.
0	TX0_MASK	R/W	0h	TX Channel 0 Mask - Write one to enable interrupt.

    - RX_INTMASK_SET

Table 14-66. RX_INTMASK_SET Register Field Descriptions

Bit	Field	Type	Reset	Description
31-16	Reserved	R	0h	
15	RX7_THRESH_PEND_MASK	R/W	0h	RX Channel 7 Threshold Pending Int. Mask - Write one to enable Int.
14	RX6_THRESH_PEND_MASK	R/W	0h	RX Channel 6 Threshold Pending Int. Mask - Write one to enable Int.
13	RX5_THRESH_PEND_MASK	R/W	0h	RX Channel 5 Threshold Pending Int. Mask - Write one to enable Int.
12	RX4_THRESH_PEND_MASK	R/W	0h	RX Channel 4 Threshold Pending Int. Mask - Write one to enable Int.
11	RX3_THRESH_PEND_MASK	R/W	0h	RX Channel 3 Threshold Pending Int. Mask - Write one to enable Int.
10	RX2_THRESH_PEND_MASK	R/W	0h	RX Channel 2 Threshold Pending Int. Mask - Write one to enable Int.
9	RX1_THRESH_PEND_MASK	R/W	0h	RX Channel 1 Threshold Pending Int. Mask - Write one to enable Int.
8	RX0_THRESH_PEND_MASK	R/W	0h	RX Channel 0 Threshold Pending Int. Mask - Write one to enable Int.
7	RX7_PEND_MASK	R/W	0h	RX Channel 7 Pending Int. Mask - Write one to enable Int.
6	RX6_PEND_MASK	R/W	0h	RX Channel 6 Pending Int. Mask - Write one to enable Int.
5	RX5_PEND_MASK	R/W	0h	RX Channel 5 Pending Int. Mask - Write one to enable Int.
4	RX4_PEND_MASK	R/W	0h	RX Channel 4 Pending Int. Mask - Write one to enable Int.
3	RX3_PEND_MASK	R/W	0h	RX Channel 3 Pending Int. Mask - Write one to enable Int.
2	RX2_PEND_MASK	R/W	0h	RX Channel 2 Pending Int. Mask - Write one to enable Int.
1	RX1_PEND_MASK	R/W	0h	RX Channel 1 Pending Int. Mask - Write one to enable Int.

Table 14-66. RX_INTMASK_SET Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
0	RX0_PEND_MASK	R/W	0h	RX Channel 0 Pending Int. Mask - Write one to enable Int.

- unmask core interrupts
  - Cn_RX_EN

Table 14-219. C0_RX_EN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	Reserved	R	0h	
7-0	C0_RX_EN	R/W	0h	Core 0 Receive Enable - Each bit in this register corresponds to the bit in the rx interrupt that is enabled to generate an interrupt on C0_RX_PULSE.

    - set bit CHANNEL to 1
  - Cn_TX_EN

Table 14-220. C0_TX_EN Register Field Descriptions

Bit	Field	Type	Reset	Description
31-8	Reserved	R	0h	
7-0	C0_TX_EN	R/W	0h	Core 0 Transmit Enable - Each bit in this register corresponds to the bit in the tx interrupt that is enabled to generate an interrupt on C0_TX_PULSE.

        - set bit CHANNEL to 1
- Acknowledge interrupts

Start up RX and TX DMA and wait for completion

- writing to HDP will cause TX and RX to start

Table 14-102. TX0_HDP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	TX_HDP	R/W	0h	TX Channel (0..7) DMA Head Descriptor Pointer - Writing a TX DMA Buffer Descriptor address to a head pointer location initiates TX DMA operations in the queue for the selected channel. Writing to these locations when they are non-zero is an error (except at reset). Host software must initialize these locations to zero on reset.

Table 14-110. RX0_HDP Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	RX_HDP	R/W	0h	RX DMA Head Descriptor Pointer - Writing an RX DMA Buffer Descriptor address to this location allows RX DMA operations in the selected channel when a channel frame is received. Writing to these locations when they are non-zero is an error (except at reset). Host software must initialize these locations to zero on reset.

Interrupt Service routines

RX DMA ISR

- Upon reception of an interrupt, software should perform the following:
    - Read the RX_STAT register to determine which channel(s) caused the interrupt
        - Write the 3PSW completion pointer(s) (RXn_CP)
            - The data written by the host (buffer descriptor address of the last processed buffer) is compared to the data in the register written by the subsystem (address of last buffer descriptor used by the subsystem). If the two values are not equal (which means that the 3PSW has received more packets than the CPU has processed), the receive packet completion interrupt signal remains asserted
            - If the two values are equal (which means that the host has processed all packets that the system has received), the pending interrupt is de-asserted.
            - The value that the 3PSW is expecting is found by reading the receive channeln completion pointer register (RXn_CP).
        - Write the value 1h to the CPDMA_EOI_VECTOR register

TX DMA ISR

- Upon receiving an interrupt, software should perform the following
    - Read the TX_STAT register to determine which channel(s) caused the interrupt
        - Process received packets for the interrupting channel
        - Write the 3PSW completion pointer(s) (TXn_CP)
            - The data written by the host (buffer descriptor address of the last processed buffer) is compared to the data in the register written by the 3PSW (address of last buffer descriptor used by the 3PSW).
            - If the two values are not equal (which means that the 3PSW has transmitted more packets than the CPU has processed), the transmit packet completion interrupt signal remains asserted.
            - If the two values are equal (which means that the host has processed all packets that the subsystem has transferred), the pending interrupt is cleared.
            - The value that the 3PSW is expecting is found by reading the transmit channeln completion pointer register (TXn_CP).
            - Write the 2h to the CPDMA_EOI_VECTOR register

Send Packet

- only going to support packet sizes of 1520 bytes
- polling work around cause interrupts dont work

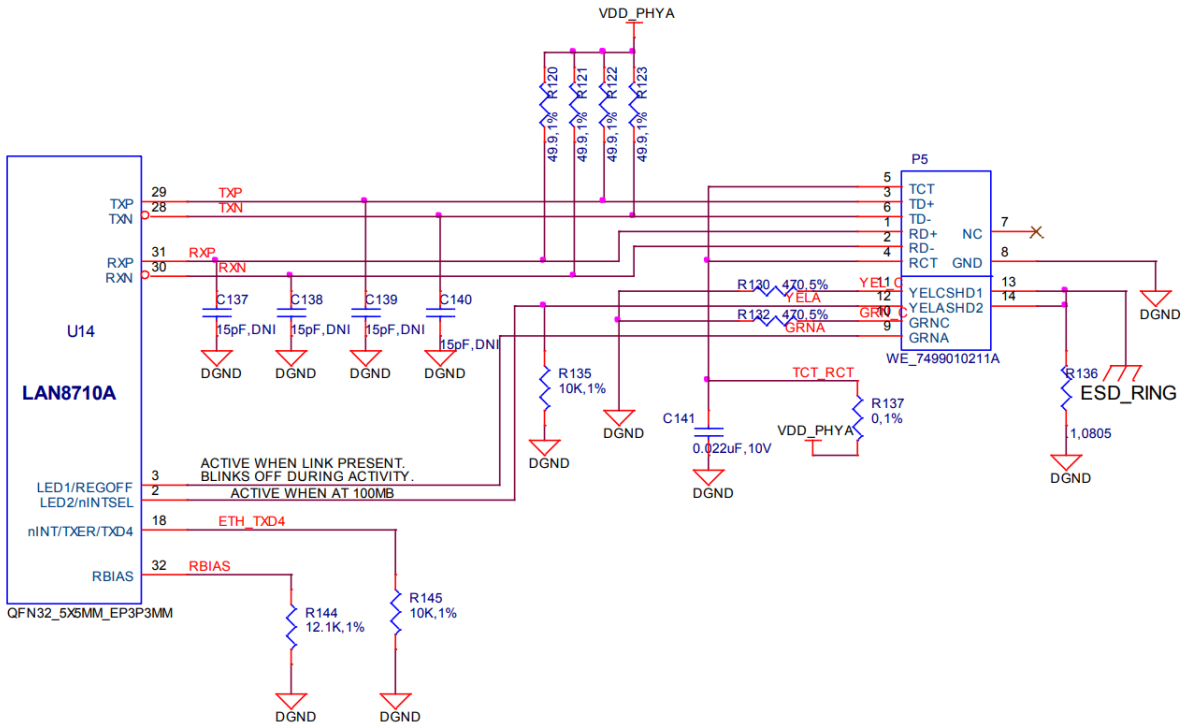
Receive Packet

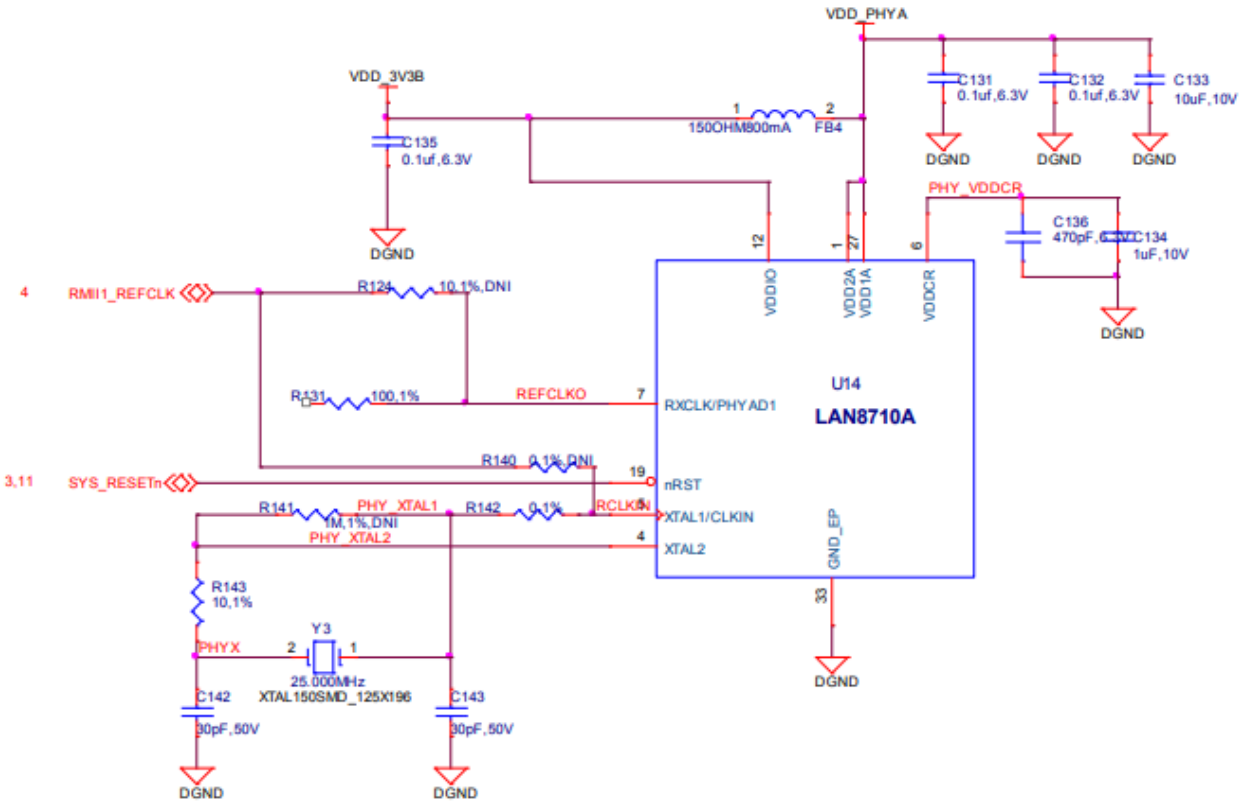
- only going to support packet sizes of 1520 bytes
- polling work around cause interrupts dont work

PHY

- https://www.ti.com/lit/ds/symlink/dp83620.pdf?ts=1742117239151&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FDP83620
    - different PHY not on board
- https://ww1.microchip.com/downloads/aemDocuments/documents/UNG/ProductDocuments/DataSheets/LAN8710A-LAN8710Ai-Data-Sheet-DS00002164.pdf

    - this one is the one we need
- SMSC Ethernet PHY is the physical interface to the network





- need to power on/reset PHY, this is not mentioned any where
    - https://github.com/beagleboard/linux/blob/master/arch/arm/boot/dts/ti/omap/am33

5x-bone-common.dtsi

```
ethphy0: ethernet-phy@0 {
        reg = <0>;
        /* Support GPIO reset on revision C3 boards */
        reset-gpios = <&gpio1 8 GPIO_ACTIVE_LOW>;
        reset-assert-us = <300>;
        reset-deassert-us = <50000>;
};
```

- GPIO1_8 can be used to reset PHY

- registers

TABLE 4-2: SMI REGISTER MAP

Register Index (Decimal)	Register Name	Group
0	Basic Control Register	Basic
1	Basic Status Register	Basic
2	PHY Identifier 1	Extended
3	PHY Identifier 2	Extended
4	Auto-Negotiation Advertisement Register	Extended
5	Auto-Negotiation Link Partner Ability Register	Extended
6	Auto-Negotiation Expansion Register	Extended
17	Mode Control/Status Register	Vendor-specific
18	Special Modes	Vendor-specific
26	Symbol Error Counter Register	Vendor-specific
27	Control / Status Indication Register	Vendor-specific
29	Interrupt Source Register	Vendor-specific
30	Interrupt Mask Register	Vendor-specific
31	PHY Special Control/Status Register	Vendor-specific

- this is for the one on the beagle bone black
- LAN8710

Table 10-1. Register Map

Offset		Access	Tag	Description
Hex	Decimal			
00h	0	RW	BMCR	Basic Mode Control Register
01h	1	RO	BMSR	Basic Mode Status Register
02h	2	RO	PHYIDR1	PHY Identifier Register #1
03h	3	RO	PHYIDR2	PHY Identifier Register #2
04h	4	RW	ANAR	Auto-Negotiation Advertisement Register
05h	5	RW	ANLPAR	Auto-Negotiation Link Partner Ability Register
06h	6	RW	ANER	Auto-Negotiation Expansion Register
07h	7	RW	ANNPTR	Auto-Negotiation Next Page TX Register
08h-0Fh	8-15		RESERVED	RESERVED
10h	16	RO	PHYSTS	PHY Status Register
11h	17	RW	MICR	MII Interrupt Control Register
12h	18	RW	MISR	MII Interrupt Status and Event Control Register
13h	19	RW	PAGESEL	Page Select Register

- reading and writing

14.4.4 Writing Data to a PHY Register

The MDIO module includes a user access register (MDIOUSERACCESS*n*) to directly access a specified PHY device.To write a PHY register, perform the following:

1. Ensure the GO bit in the MDIO user access register (MDIOUSERACCESS*n*) is cleared.

2. Write to the GO, WRITE, REGADR, PHYADR, and DATA bits in MDIOUSERACCESS*n* corresponding to the PHY and PHY register you want to write.
3. The write operation to the PHY is scheduled and completed by the MDIO module. Completion of the write operation can be determined by polling the GO bit in MDIOUSERACCESS*n* for a 0.
4. Completion of the operation sets the corresponding USERINTRAW bit (0 or 1) in the MDIO user command complete interrupt register (MDIOUSERINTRAW) corresponding to USERACCESS*n* used. If interrupts have been enabled on this bit using the MDIO user command complete interrupt mask set register (MDIOUSERINTMASKSET), then the bit is also set in the MDIO user command complete interrupt register (MDIOUSERINTMASKED) and an interrupt is triggered on the CPU.

14.4.5 Reading Data from a PHY Register

The MDIO module includes a user access register (MDIOUSERACCESS*n*) to directly access a specified PHY device. To read a PHY register, perform the following:

1. Ensure the GO bit in the MDIO user access register (MDIOUSERACCESS*n*) is cleared.
2. Write to the GO, REGADR, and PHYADR bits in MDIOUSERACCESS*n* corresponding to the PHY and PHY register you want to read.
3. The read data value is available in the DATA bits in MDIOUSERACCESS*n* after the module completes the read operation on the serial bus. Completion of the read operation can be determined by polling the GO and ACK bits in MDIOUSERACCESS*n*. After the GO bit has cleared, the ACK bit is set on a successful read.
4. Completion of the operation sets the corresponding USERINTRAW bit (0 or 1) in the MDIO user command complete interrupt register (MDIOUSERINTRAW) corresponding to USERACCESS*n* used. If interrupts have been enabled on this bit using the MDIO user command complete interrupt mask set register (MDIOUSERINTMASKSET), then the bit is also set in the MDIO user command complete interrupt register (MDIOUSERINTMASKED) and an interrupt is triggered on the CPU.

Table 14-260. MDIOUSERACCESS0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31	GO	R/W/S	0	Go. Writing a 1 to this bit causes the MDIO state machine to perform an MDIO access when it is convenient for it to do so, this is not an instantaneous process. Writing a 0 to this bit has no effect. This bit is write able only if the MDIO state machine is enabled. This bit will self clear when the requested access has been completed. Any writes to the MDIOUSERACCESS0 register are blocked when the GO bit is 1. If byte access is being used, the GO bit should be written last.
30	WRITE	R/W	0	Write enable. Setting this bit to a 1 causes the MDIO transaction to be a register write, otherwise it is a register read.
29	ACK	R/W	0	Acknowledge. This bit is set if the PHY acknowledged the read transaction.
28-26	Reserved	R	0	Reserved.
25-21	REGADR	R/W	0	Register address. Specifies the PHY register to be accessed for this transaction.
20-16	PHYADR	R/W	0	PHY address. Specifies the PHY to be accesses for this transaction.
15-0	DATA	R/W	0	User data. The data value read from or to be written to the specified PHY register.

-
- check if PHY is alive

Table 14-252. MDIOALIVE Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ALIVE	R/WC	0	MDIO alive. Each of the 32 bits of this register is set if the most recent access to the PHY with address corresponding to the register bit number was acknowledged by the PHY, the bit is reset if the PHY fails to acknowledge the access. Both the user and polling accesses to a PHY will cause the corresponding alive bit to be updated. The alive bits are only meant to be used to give an indication of the presence or not of a PHY with the corresponding address. Writing a 1 to any bit will clear it, writing a 0 has no effect.

-
- https://www.ti.com/lit/an/spracc8a/spracc8a.pdf?ts=1742108678425&ref_url=https%253A%252F%252Fwww.google.com%252F
- establish link
  -
- BCR register

Bits	Description	Type	Default
15	**Soft Reset** 1 = software reset. Bit is self-clearing. When setting this bit do not set other bits in this register. The configuration (as described in Section 3.7.2, MODE[2:0]: Mode Configuration) is set from the register bit values, and not from the mode pins.	R/W SC	0b
14	**Loopback** 0 = normal operation 1 = loopback mode	R/W	0b
13	**Speed Select** 0 = 10Mbps 1 = 100Mbps Ignored if Auto-negotiation is enabled (0.12 = 1).	R/W	Note 4-1
12	**Auto-Negotiation Enable** 0 = disable auto-negotiate process 1 = enable auto-negotiate process (overrides 0.13 and 0.8)	R/W	Note 4-1

  -

Bits	Description	Type	Default
11	**Power Down** 0 = normal operation 1 = General power down mode The Auto-Negotiation Enable must be cleared before setting the Power Down.	R/W	0b
10	**Isolate** 0 = normal operation 1 = electrical isolation of PHY from the MII/RMII	R/W	0b
9	**Restart Auto-Negotiate** 0 = normal operation 1 = restart auto-negotiate process Bit is self-clearing.	R/W SC	0b
8	**Duplex Mode** 0 = half duplex 1 = full duplex Ignored if Auto-Negotiation is enabled (0.12 = 1).	R/W	Note 4-1
7	**Collision Test** 0 = disable COL test 1 = enable COL test	R/W	0b
6:0	**RESERVED**	RO	—

- AUTONEG ADV

Bits	Description	Type	Default
15:14	**RESERVED**	RO	—
13	Remote Fault 0 = no remote fault 1 = remote fault detected	R/W	0b
12	**RESERVED**	RO	—
11:10	**Pause Operation** 00 = No PAUSE 01 = Symmetric PAUSE 10 = Asymmetric PAUSE toward link partner 11 = Advertise support for both Symmetric PAUSE and Asymmetric PAUSE toward local device **Note:** When both Symmetric PAUSE and Asymmetric PAUSE are set, the device will only be configured to, at most, one of the two settings upon auto-negotiation completion.	R/W	00b
9	**RESERVED**	RO	—
8	100BASE-TX Full Duplex 0 = no TX full duplex ability 1 = TX with full duplex	R/W	Note 4-3
7	100BASE-TX 0 = no TX ability 1 = TX able	R/W	1b
6	10BASE-T Full Duplex 0 = no 10Mbps with full duplex ability 1 = 10Mbps with full duplex	R/W	Note 4-3

Bits	Description	Type	Default
5	10BASE-T 0 = no 10Mbps ability 1 = 10Mbps able	R/W	Note 4-3
4:0	Selector Field 00001 = IEEE 802.3	R/W	00001b

- BSR

Bits	Description	Type	Default
15	100BASE-T4 0 = no T4 ability 1 = T4 able	RO	0b
14	100BASE-TX Full Duplex 0 = no TX full duplex ability 1 = TX with full duplex	RO	1b
13	100BASE-TX Half Duplex 0 = no TX half duplex ability 1 = TX with half duplex	RO	1b
12	10BASE-T Full Duplex 0 = no 10Mbps with full duplex ability 1 = 10Mbps with full duplex	RO	1b
11	10BASE-T Half Duplex 0 = no 10Mbps with half duplex ability 1 = 10Mbps with half duplex	RO	1b

Bits	Description	Type	Default
10	100BASE-T2 Full Duplex 0 = PHY not able to perform full duplex 100BASE-T2 1 = PHY able to perform full duplex 100BASE-T2	RO	0b
9	100BASE-T2 Half Duplex 0 = PHY not able to perform half duplex 100BASE-T2 1 = PHY able to perform half duplex 100BASE-T2	RO	0b
8	Extended Status 0 = no extended status information in register 15 1 = extended status information in register 15	RO	0b
7:6	**RESERVED**	RO	—
5	Auto-Negotiate Complete 0 = auto-negotiate process not completed 1 = auto-negotiate process completed	RO	0b
4	Remote Fault 1 = remote fault condition detected 0 = no remote fault	RO/LH	0b
3	Auto-Negotiate Ability 0 = unable to perform auto-negotiation function 1 = able to perform auto-negotiation function	RO	1b
2	Link Status 0 = link is down 1 = link is up	RO/LL	0b
1	Jabber Detect 0 = no jabber condition detected 1 = jabber condition detected	RO/LH	0b
0	Extended Capabilities 0 = does not support extended capabilities registers 1 = supports extended capabilities registers	RO	1b

- partner cap register

Bits	Description	Type	Default
15	Next Page 0 = no next page ability 1 = next page capable **Note:** This device does not support next page ability.	RO	0b
14	Acknowledge 0 = link code word not yet received 1 = link code word received from partner	RO	0b
13	Remote Fault 0 = no remote fault 1 = remote fault detected	RO	0b
12:11	**RESERVED**	RO	—
10	Pause Operation 0 = No PAUSE supported by partner station 1 = PAUSE supported by partner station	RO	0b
9	100BASE-T4 0 = no T4 ability 1 = T4 able **Note:** This device does not support T4 ability.	RO	0b
8	100BASE-TX Full Duplex 0 = no TX full duplex ability 1 = TX with full duplex	RO	0b
7	100BASE-TX 0 = no TX ability 1 = TX able	RO	0b
6	**10BASE-T Full Duplex** 0 = no 10Mbps with full duplex ability 1 = 10Mbps with full duplex	RO	0b

Bits	Description	Type	Default
5	10BASE-T 0 = no 10Mbps ability 1 = 10Mbps able	RO	0b
4:0	Selector Field 00001 = IEEE 802.3	RO	00001b

- SL maccontrol

Table 14-189. MACCONTROL Register Field Descriptions

Bit	Field	Type	Reset	Description
31-25	RESERVED	Rreturns0s	0h	
24	RX_CMF_EN	R/W	0h	RX Copy MAC Control Frames Enable - Enables MAC control frames to be transferred to memory. MAC control frames are normally acted upon (if enabled), but not copied to memory. MAC control frames that are pause frames will be acted upon if enabled in the MacControl register, regardless of the value of rx_cmf_en. Frames transferred to memory due to rx_cmf_en will have the control bit set in their EOP buffer descriptor. 0 - MAC control frames are filtered (but acted upon if enabled). 1 - MAC control frames are transferred to memory.
23	RX_CSF_EN	R/W	0h	RX Copy Short Frames Enable - Enables frames or fragments shorter than 64 bytes to be copied to memory. Frames transferred to memory due to rx_csf_en will have the fragment or undersized bit set in their receive footer. Fragments are short frames that contain CRC/align/code errors and undersized are short frames without errors. 0 - Short frames are filtered 1 - Short frames are transferred to memory.
22	RX_CEF_EN	R/W	0h	RX Copy Error Frames Enable - Enables frames containing errors to be transferred to memory. The appropriate error bit will be set in the frame receive footer. Frames containing errors will be filtered when rx_cef _en is not set. 0 - Frames containing errors are filtered. 1 - Frames containing errors are transferred to memory.
21	TX_SHORT_GAP_LIM_EN	R/W	0h	Transmit Short Gap Limit Enable When set this bit limits the number of short gap packets transmitted to 100ppm. Each time a short gap packet is sent, a counter is loaded with 10,000 and decremented on each wireside clock. Another short gap packet will not be sent out until the counter decrements to zero. This mode is included to preclude the host from filling up the FIFO and sending every packet out with short gap which would violate the maximum number of packets per second allowed.

Table 14-189. MACCONTROL Register Field Descriptions (continued)

Bit	Field	Type	Reset	Description
20-19	RESERVED	Rreturns0s	0h	
18	EXT_EN	R/W	0h	Mode of operation. 0 - Forced mode of operation. 1 - In-band mode of operation.
17	GIG_FORCE	R/W	0h	Gigabit Mode Force - This bit is used to force the CPGMAC_SL into gigabit mode if the input GMII_MTCLK has been stopped by the PHY.
16	IFCTL_B	R/W	0h	Connects to the speed_in input of the respective RMII gasket. When using RMII mode: 0 - 10Mbps operation 1 - 100Mbps operation
15	IFCTL_A	R/W	0h	Connects to the speed_in input of the respective RMII gasket. When using RMII mode: 0 - 10Mbps operation 1 - 100Mbps operation
14-12	RESERVED	Rreturns0s	0h	
11	CMD_IDLE	R/W	0h	Command Idle 0 - Idle not commanded 1 - Idle Commanded (read idle in MacStatus)
10	TX_SHORT_GAP_EN	R/W	0h	Transmit Short Gap Enable 0 - Transmit with a short IPG is disabled 1 - Transmit with a short IPG (when TX_SHORT_GAP input is asserted) is enabled.
9-8	RESERVED	Rreturns0s	0h	
7	GIG	R/W	0h	Gigabit Mode - 0 - 10/100 mode 1 - Gigabit mode (full duplex only) The GIG_OUT output is the value of this bit.
6	TX_PACE	R/W	0h	Transmit Pacing Enable 0 - Transmit Pacing Disabled 1 - Transmit Pacing Enabled
5	GMII_EN	R/W	0h	GMII Enable Bit. This bit must be set before the MAC will transmit or receive data in any of the supported interface modes (such as MII and RMII). This bit does not select the interface mode, but rather holds or releases the MAC TX and RX state machines from reset. 0 - The MAC RX and TX state machines are held in reset 1 - The MAC RX and TX state machines are released from reset and transmit/receive are enabled.
4	TX_FLOW_EN	R/W	0h	Transmit Flow Control Enable - Determines if incoming pause frames are acted upon in full-duplex mode. Incoming pause frames are not acted upon in half-duplex mode regardless of this bit setting. The RX_MBP_Enable bits determine whether or not received pause frames are transferred to memory. 0 - Transmit Flow Control Disabled. Full-duplex mode - Incoming pause frames are not acted upon. 1 - Transmit Flow Control Enabled . Full-duplex mode - Incoming pause frames are acted upon.
3	RX_FLOW_EN	R/W	0h	Receive Flow Control Enable - 0 - Receive Flow Control Disabled Half-duplex mode - No flow control generated collisions are sent. Full-duplex mode - No outgoing pause frames are sent. 1 - Receive Flow Control Enabled Half-duplex mode - Collisions are initiated when receive flow control is triggered. Full-duplex mode - Outgoing pause frames are sent when receive flow control is triggered.
2	MTEST	R/W	0h	Manufacturing Test mode - This bit must be set to allow writes to the Backoff_Test and PauseTimer registers.

Table 14-189. MACCONTROL Register Field Descriptions (continued)

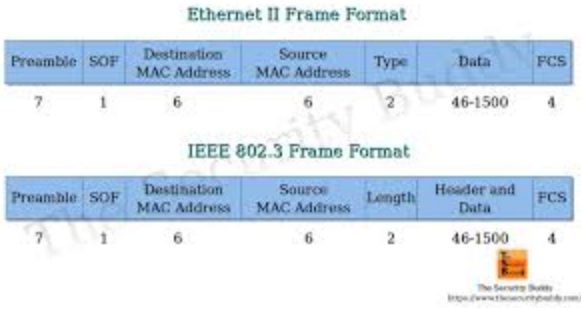
Bit	Field	Type	Reset	Description
1	LOOPBACK	R/W	0h	Loop Back Mode - Loopback mode forces internal fullduplex mode regardless of whether the fullduplex bit is set or not. The loopback bit should be changed only when GMII_en is deasserted. 0 - Not looped back 1 - Loop Back Mode enabled
0	FULLDUPLEX	R/W	0h	Full Duplex mode - Gigabit mode forces fullduplex mode regardless of whether the fullduplex bit is set or not. The FULLDUPLEX_OUT output is the value of this register bit 0 - half duplex mode 1 - full duplex mode

- MDIOLINK

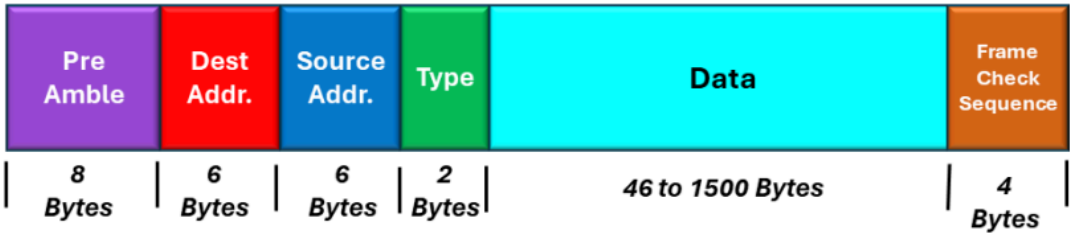
Table 14-253. MDIOLINK Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	LINK	R	0	MDIO link state. This register is updated after a read of the Generic Status Register of a PHY. The bit is set if the PHY with the corresponding address has link and the PHY acknowledges the read transaction. The bit is reset if the PHY indicates it does not have link or fails to acknowledge the read transaction. Writes to the register have no effect. In addition, the status of the two PHYs specified in the MDIOUSERPHYSEL*n* registers can be determined using the MLINK input pins. This is determined by the LINKSEL bit in the MDIOUSERPHYSEL*n* register.

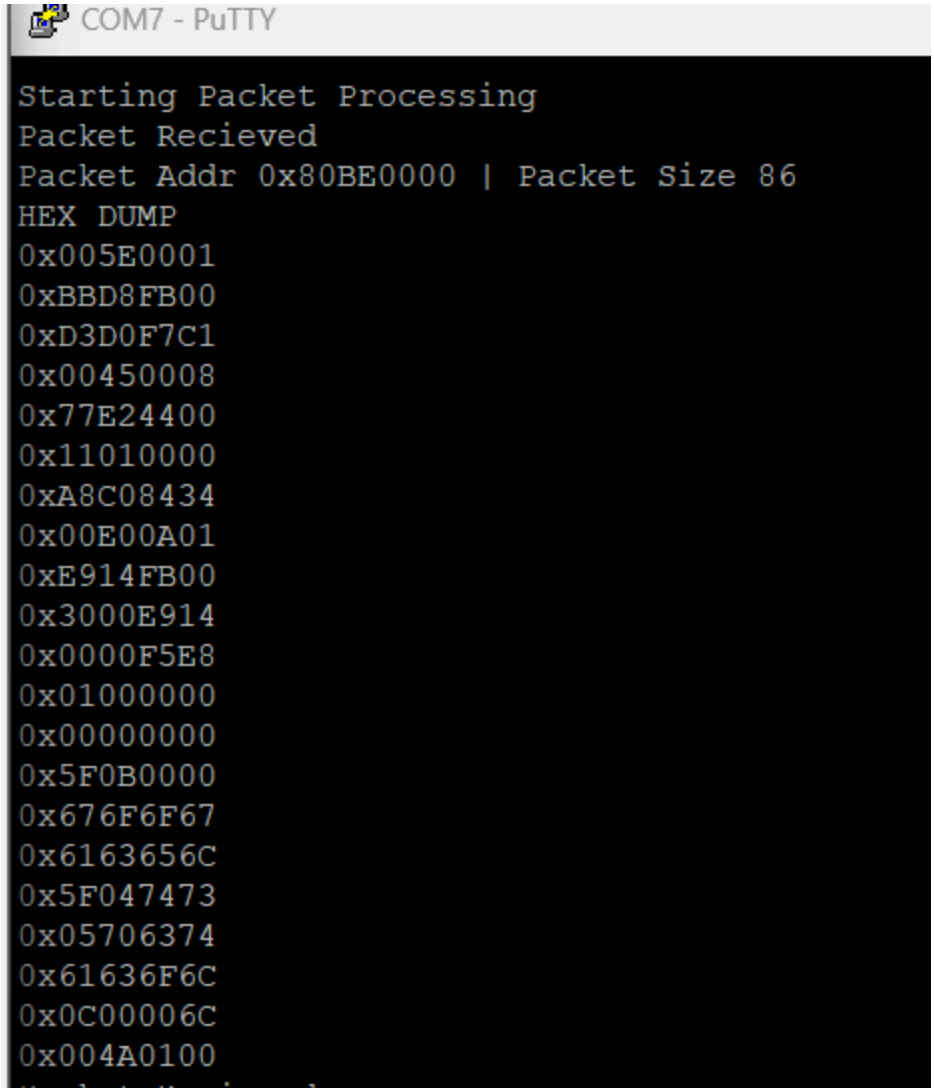
Ethernet Layer

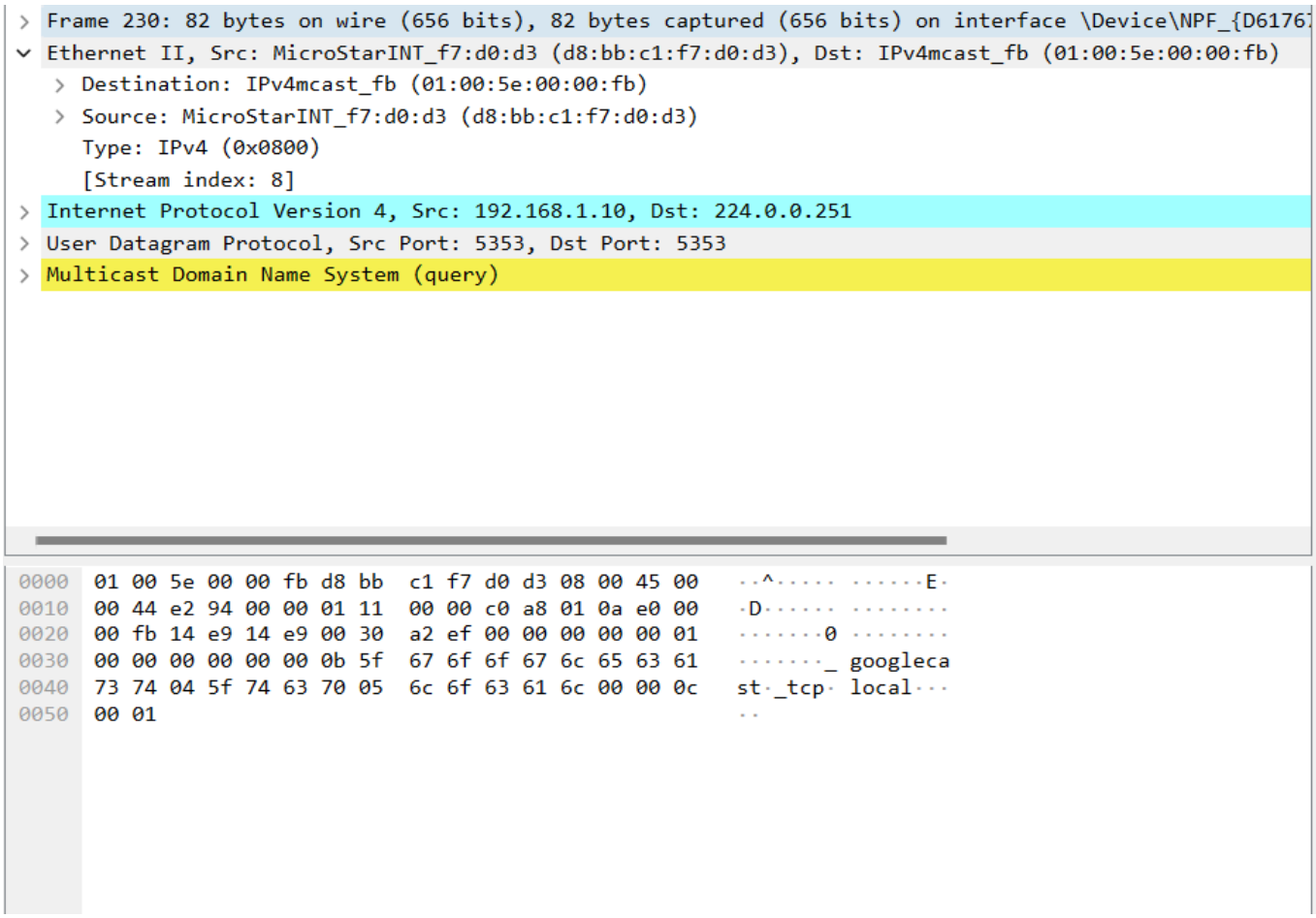


- supporting Ethernet 2

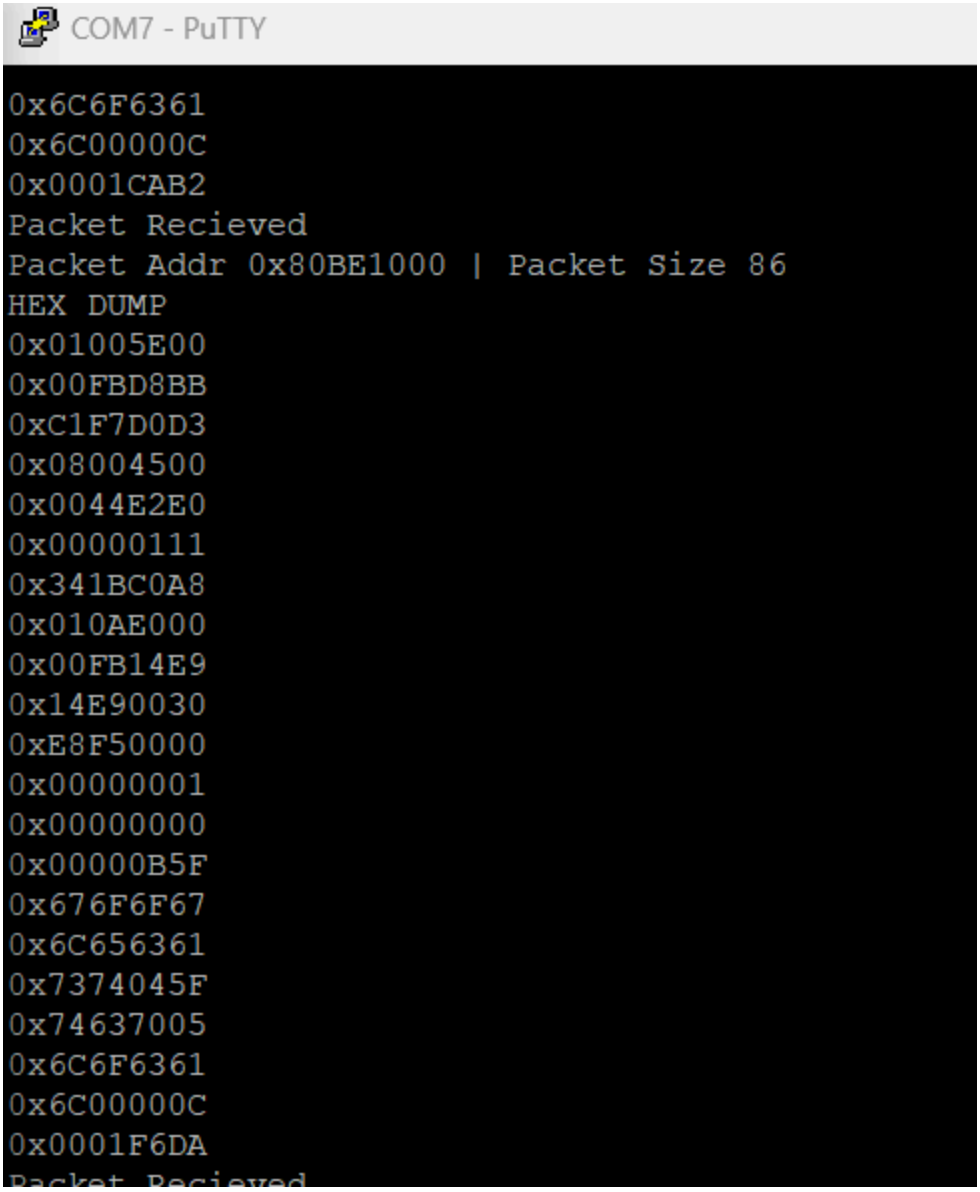


Minimum Frame Size = 64 Byte, Max = 1518 Bytes + Preamble


COM7 - PuTTY
Starting Packet Processing
Packet Recieved
Packet Addr 0x80BE0000 | Packet Size 86
HEX DUMP
0x005E0001
0xBBD8FB00
0xD3D0F7C1
0x00450008
0x77E24400
0x11010000
0xA8C08434
0x00E00A01
0xE914FB00
0x3000E914
0x0000F5E8
0x01000000
0x00000000
0x5F0B0000
0x676F6F67
0x6163656C
0x5F047473
0x05706374
0x61636F6C
0x0C00006C
0x004A0100

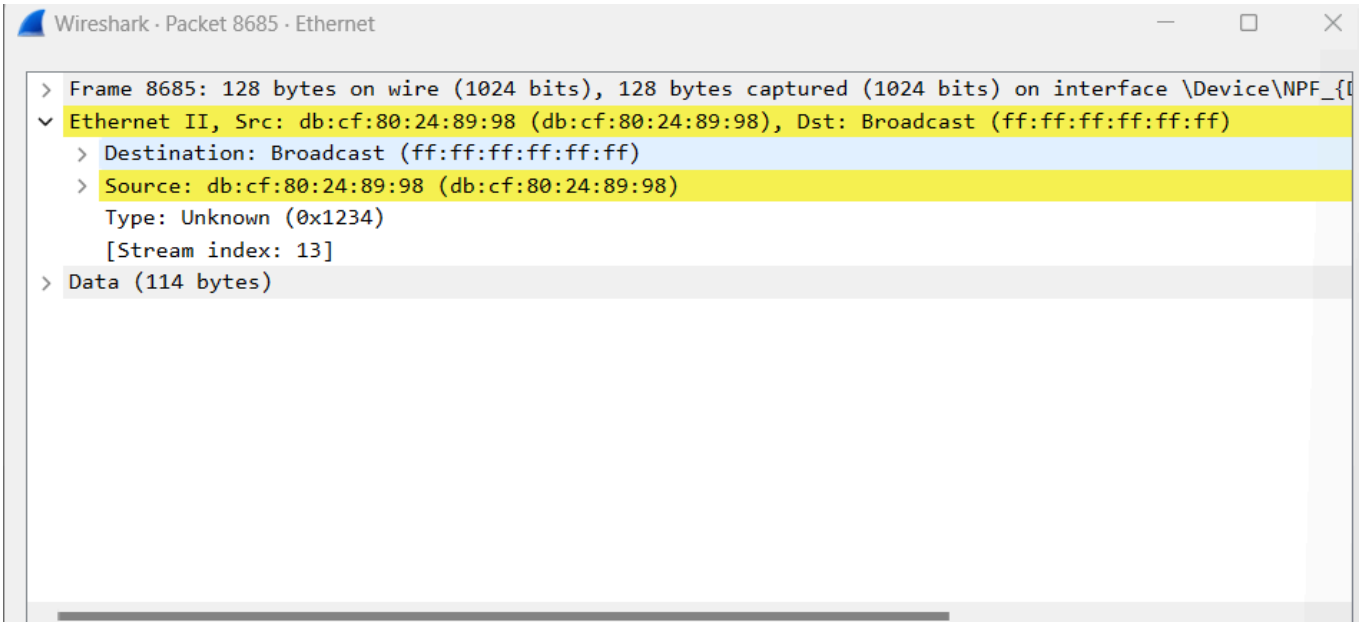

Frame 230: 82 bytes on wire (656 bits), 82 bytes captured (656 bits) on interface \Device\NPF_{D6176
Ethernet II, Src: MicroStarINT_f7:d0:d3 (d8:bb:c1:f7:d0:d3), Dst: IPv4mcast_fb (01:00:5e:00:00:fb)
Destination: IPv4mcast_fb (01:00:5e:00:00:fb)
Source: MicroStarINT_f7:d0:d3 (d8:bb:c1:f7:d0:d3)
Type: IPv4 (0x0800)
[Stream index: 8]
Internet Protocol Version 4, Src: 192.168.1.10, Dst: 224.0.0.251
User Datagram Protocol, Src Port: 5353, Dst Port: 5353
Multicast Domain Name System (query)

0000 01 00 5e 00 00 fb d8 bb c1 f7 d0 d3 08 00 45 00
0010 00 44 e2 94 00 00 01 11 00 00 c0 a8 01 0a e0 00
0020 00 fb 14 e9 14 e9 00 30 a2 ef 00 00 00 00 00 01
0030 00 00 00 00 00 00 0b 5f 67 6f 6f 67 6c 65 63 61
0040 73 74 04 5f 74 63 70 05 6c 6f 63 61 6c 00 00 0c
0050 00 01



https://e2e.ti.com/support/processors-group/processors/f/processors-forum/318448/am335x-cpsw-problem-with-broadcast-transmission

```
Crafting Ethernet Frame
Destination MAC
MAC ADDR: 0x000000FF:0x000000FF:0x000000FF:0x000000FF:0x000000FF:0x000000FF
Source MAC
MAC ADDR: 0x000000DB:0x000000CF:0x00000080:0x00000024:0x00000089:0x00000098
Frame Type 0x00001234
Transmiting Packet
Packet Transmited
```



https://www.ti.com/lit/er/sprz360i/sprz360i.pdf?ts=1742256719716&ref_url=https%253A%252F%252Fwww.google.com%252F

https://www.networxsecurity.org/members-area/glossary/e/ethertype.html

EtherType	Protocol
0x0800	Internet Protocol version 4 (IPv4)
0x0806	Address Resolution Protocol (ARP)

ARP

Internet Protocol (IPv4) over Ethernet ARP packet

Offset	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	*Hardware Type (1)*																*Protocol Type (0x0800)*															
4	32	*Hardware Length (6)*								*Protocol Length (4)*								*Operation*															
8	64	*Sender Hardware Address*																															
12	96																	*Sender Protocol Address*															
16	128	↪ *Sender Protocol Address (cont.)*																*Target Hardware Address*															
20	160																																
24	192	*Target Protocol Address*																															

Gratuitous

Example Traffic

```
Ethernet II, Src: 02:02:02:02:02:02, Dst: ff:ff:ff:ff:ff:ff
    Destination: ff:ff:ff:ff:ff:ff (Broadcast)
    Source: 02:02:02:02:02:02 (02:02:02:02:02:02)
    Type: ARP (0x0806)
    Trailer: 00000000000000000000000000000000000000
Address Resolution Protocol (request/gratuitous ARP)
    Hardware type: Ethernet (0x0001)
    Protocol type: IP (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (0x0001)
    Sender MAC address: 02:02:02:02:02:02 (02:02:02:02:02:02)
    Sender IP address: 192.168.1.1 (192.168.1.1)
    Target MAC address: ff:ff:ff:ff:ff:ff (Broadcast)
    Target IP address: 192.168.1.1 (192.168.1.1)
0000  ff ff ff ff ff ff 02 02 02 02 02 02 08 06 00 01   ..............
0010  08 00 06 04 00 01 02 02 02 02 02 02 c0 a8 01 01   ..............
0020  ff ff ff ff ff ff c0 a8 01 01 00 00 00 00 00 00   ..............
0030  00 00 00 00 00 00 00 00 00 00 00 00               ............
```

192.168.1.20

```c
typedef struct arp {

    uint16_t hardware_type;
    uint16_t protocol_type;
    uint8_t hardware_length;
    uint8_t protocol_length;
    uint16_t operation;
    uint8_t src_mac[MAC_ADDR_LEN];
    uint32_t src_ip;
    uint8_t dest_mac[MAC_ADDR_LEN];
    uint32_t dest_ip;

} __attribute__((packed)) arp_header;
```

https://www.iana.org/assignments/arp-parameters/arp-parameters.xhtml

Number	Operation Code (op)	References
0	Reserved	[RFC5494]
1	REQUEST	[RFC826][RFC5227]
2	REPLY	[RFC826][RFC5227]

Get-NetNeighbor -InterfaceAlias "Ethernet"
netsh interface ip show neighbors
arp -a

Number	Hardware Type (hrd)	Reference
0	Reserved	[RFC5494]
1	Ethernet (10Mb)	[Jon Postel]

need to do arp request arp reply and gratuitous arp

IPV4

IPv4 header format

Offset	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Version (4)				IHL				DSCP						ECN		Total Length															
4	32	Identification																Flags			Fragment Offset											
8	64	Time to Live								Protocol								Header Checksum															
12	96	Source address																															
16	128	Destination address																															
20	160	(Options) (if IHL > 5)																															
⋮	⋮																																
56	448																																

icmp hex dump

```
0000  b8 5e 71 30 ad fc 84 7b  57 43 34 60 08 00 45 00   ·^q0···{ WC4`··E·
0010  00 3c ec 92 00 00 80 01  00 00 0a 00 00 39 0a 00   ·<····· ·····9··
0020  00 01 08 00 4d 5a 00 01  00 01 61 62 63 64 65 66   ····MZ·· ··abcdef
0030  67 68 69 6a 6b 6c 6d 6e  6f 70 71 72 73 74 75 76   ghijklmn opqrstuv
0040  77 61 62 63 64 65 66 67  68 69                    wabcdefg hi
```

```
typedef struct ipv4 {

    uint8_t version;
    uint8_t ihl;
    uint8_t dhsp;
    uint8_t ecn;
    uint16_t total_length;
    uint16_t identification;
    uint8_t flags
    uint16_t fragment_offset;
    uint8_t time_to_live;
    uint8_t protocol;
    uint16_t header_checksum;
    uint32_t src_ip;
    uint32_t dest_ip;

} ipv4_header;
```

ipv4 checksum

https://datatracker.ietf.org/doc/html/rfc1071

Checksum computation [edit]

The method used to compute the checksum is defined in RFC 768, and efficient calculation is discussed in RFC 1071:

> Checksum is the 16-bit ones' complement of the ones' complement sum of a pseudo header of information from the IP header, the UDP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.[7]

ipv4 protocol numbers

https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml

Decimal	Keyword	Protocol	IPv6 Extension Header	Reference
0	HOPOPT	IPv6 Hop-by-Hop Option	Y	[RFC8200]
1	ICMP	Internet Control Message		[RFC792]
2	IGMP	Internet Group Management		[RFC1112]
3	GGP	Gateway-to-Gateway		[RFC823]
4	IPv4	IPv4 encapsulation		[RFC2003]
5	ST	Stream		[RFC1190][RFC1819]
6	TCP	Transmission Control		[RFC9293]
7	CBT	CBT		[Tony Ballardie]
8	EGP	Exterior Gateway Protocol		[RFC888][David Mills]
9	IGP	any private interior gateway (used by Cisco for their IGRP)		[Internet Assigned Numbers Authority]
10	BBN-RCC-MON	BBN RCC Monitoring		[Steve Chipman]
11	NVP-II	Network Voice Protocol		[RFC741][Steve Casner]
12	PUP	PUP		[Boggs, D., J. Shoch, E. Taft, and R. Metcalfe, "PUP: An Internetwork Architecture", XEROX Palo Alto Research Center, CSL-79-10, July 1979; also in IEEE Transactions on Communication, Volume COM-28, Number 4, April 1980.][[XEROX]]
13	ARGUS (deprecated)	ARGUS		[Robert W Scheifler]
14	EMCON	EMCON		[Bich Nguyen]
15	XNET	Cross Net Debugger		[Haverty, J., "XNET Formats for Internet Protocol Version 4", IEN 158, October 1980.][Jack Haverty]
16	CHAOS	Chaos		[J Noel Chiappa]
17	UDP	User Datagram		[RFC768][Jon Postel]

- icmp
  - 1
- tcp
  - 6
- udp
  - 17

ICMP

ICMP header format

Large

Offset	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	*Type*								*Code*								*Checksum*															
4	32	*Rest of Header*																															

Color (beta)

Type	Code	Status	Description
0 – Echo Reply[2]:14	0		Echo reply (used to ping)
8 – Echo Request	0		Echo request (used to ping)

Echo reply

- type = 0
- code = 0

  Echo request
- type = 8
- code = 0

https://www.iana.org/assignments/icmp-parameters/icmp-parameters.xhtml

UDP

UDP header format[7]

Offset	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source Port																Destination Port															
4	32	Length																Checksum															
8	64	Data																															
12	96																																
⋮	⋮																																

https://stackoverflow.com/questions/45908909/my-udp-checksum-calculation-gives-wrong-results-every-time

check sum is calculated using pseudo header

IPv4 pseudo header [edit]

When UDP runs over IPv4, the checksum is computed using a *pseudo header* that contains some of the same information from the real IPv4 header.[7]:2 The pseudo header is not the real IPv4 header used to send an IP packet, it is used only for the checksum calculation. UDP checksum computation is optional for IPv4. If a checksum is not used it should be set to the value zero.

Large
Width
Standard
Wide

UDP pseudo-header for checksum computation (IPv4)

Offset	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	*Source Address*																															
4	32	*Destination Address*																															
8	64	*Zeroes*								*Protocol*								*UDP Length*															
12	96	*Source Port*																*Destination Port*															
16	128	*Length*																*Checksum*															
20	160	*Data*																															
24	192																																
⋮	⋮																																

The checksum is calculated over the following fields: