EMAC8, EMAC8_MD
Ethernet Media Access Controller

Summary

This document provides detailed reference information with respect to the non-Wishbone EMAC peripheral components, EMAC8 and EMAC8_MD.

The 8-bit Standard and Extended Ethernet Media Access Controller components (EMAC8 and EMAC8_MD respectively) provide an interface between a processor and a standard Physical Layer device (PHY) through support of the IEEE802.3 Media Independent Interface (MII).

Important Notice: Supply of these soft cores under the terms and conditions of the Altium End-User License Agreement does not convey nor imply any patent rights to the supplied technologies. Users are cautioned that a license may be required for any use covered by such patent rights.

Features

- 10/100Mbps at Full/Half Duplex
- Independent single message buffering – both send and receive
- MII compatible interface for connecting external 10/100Mbps PHY transceivers
- Ethernet/IEEE802.3 compatible
- PHY register access (**EMAC8_MD** only)
- No lower limit for external (system) clock frequency

Availability

The following variants of the Controller are available:

- **EMAC8** – Standard version of the Ethernet Media Access Controller, providing an 8-bit host interface
- **EMAC8_MD** – Extended version of the Ethernet Media Access Controller, providing an 8-bit host interface. This variant enables write/read of the registers internal to the connected PHY device.

Both components can be found in the FPGA Peripherals integrated library (`FPGA Peripherals.IntLib`), located in the `\Library\Fpga` folder of the installation.

Ethernet Protocol

Data Packaging Format

Data is transmitted over the Ethernet in frames. Figure 1 shows the constitution of such a frame, the fields of which are described in the sections that follow.

Preamble	SFD	Destination Address	Source Address	Length/ Type	Data	Padding	FCS

Figure 1. Standard IEEE802.3 Ethernet message frame format

Preamble

This field of the frame is used for synchronization purposes and contains 7 Bytes, all of which have the bit pattern 10101010. The 7 Bytes are transmitted from left to right, with the individual bits in each byte transmitted from right to left (LSB to MSB).

The Preamble field is automatically created and inserted into the frame by the Controller, upon transmission. Upon reception of a frame from a connected PHY device, the Preamble field is stripped from the frame and discarded.

If, while transmitting the Preamble portion of a frame a collision is detected, transmission will continue until transmission of the Preamble and subsequent SFD fields are completed.

SFD

The SFD (Start of Frame Delimiter) field is used to mark the beginning of the frame. It contains a single byte with the bit pattern 10101011. The bits are transmitted from left to right (MSB to LSB).

The SFD field is automatically created and inserted into the frame by the Controller, upon transmission. Upon reception of a frame from a connected PHY device, the SFD field is stripped from the frame and discarded.

If, while transmitting the SFD portion of a frame a collision is detected, transmission will continue until the field has completed.

Destination Address

This field of the frame is used to specify the address of the PHY device to which you wish to transmit data. The field is 6 Bytes long, resulting in a 48-bit address, the least significant bit of which is used to determine whether the transmitted data is for a single, specific PHY device (‘0’) or for multiple PHY devices (‘1’).

If all 48-bits of the address are set High, then the transmission will be ‘broadcast’ to all connected PHY devices.

The Destination field is transmitted from right to left (LSB to MSB).

Note: The EMAC8 and EMAC8_MD support unicast and broadcast modes of communication only.

Source Address

This field of the frame is used to specify the address of the Controller itself (i.e. the source of the transmission). This field is also 6 Bytes in length, equating to a 48-bit address.

Note: The Source address for the Controller included in a frame to be transmitted is not the same as the address written to the MAC_ADDR register on initialization. The latter is used by the receiver to determine whether a message frame on the Ethernet bus should be received or not.

The Source field is transmitted from right to left (LSB to MSB).

Length/Type

This field of the frame can either be used to represent the length, in Bytes, of the Data field, or the frame type. The 2 Byte value in this field is translated as follows:

- If the value in the Length/Type field is less than 0600h, the value is representing the length of the data field (in Bytes). This value does not include any bytes contained in the Padding field.
- If the value in the Length/Type field is greater than or equal to 0600h, the value is representing the type of frame being sent/received.

The two bytes are transmitted from left to right with individual bit transmission from right to left (LSB to MSB).

Note: This field is not used by any of the Controller variants, in any way.

Data

This field contains the main data for the frame. The Data field can be anywhere from 0 to 1500 Bytes in length and is transmitted with the least significant bit first.

Padding

This field of the frame is used to ensure that the frame is at least the minimum length required for successful Half Duplex communications using the Carrier Sense Multiple Access with Collision Detection (CSMA/CD) bus access method. This minimum frame length is 64 Bytes, not including the Preamble and SFD fields.

The length of the Padding field can vary between 0 and 46 Bytes and depends on the number of Bytes contained in the Data field:

- Data field length = 0……………………..Padding field length = 46
- 0< Data field length < 46………………..Padding field length = 46 - Data field length
- Data field length >= 46………………….Padding field length = 0

The Padding field is automatically created with the necessary number of Bytes and inserted into the frame by the Controller, if required, upon transmission. Each padding byte inserted will have the pattern 00000000.

FCS

This field of the frame, is used to provide frame checking, which is ultimately used by the receiver device to ensure that the frame has not corrupted. Frame checking is implemented using a 32-bit Cyclic Redundancy Check (CRC). The value in the field is calculated by performing a modulo 2 division of the frame content (with the exception of the Preamble and SFD fields) by the following polynomial expression:

```
G(x) = x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + x1 + 1
```

The remainder of the division is used as the 4 Byte value for the FCS field. The most significant bit of the remainder is place as the left-most bit in the left-most byte of the field.

The FCS field is automatically created and inserted into the frame by the Controller, upon transmission. The bytes are transmitted from left to right, with individual bits transmitted also from left to right (MSB to LSB). Upon reception of a message frame, the Controller checks the CRC value. If the check reveals an error, the message is skipped. If the CRC check returns no error, the message – including CRC is loaded into the Receive Buffer. The FCS field will always be the last 6 Bytes of a received message.

Interframe Gap

The IEEE802.3 Standard specifies a minimum length of time that must be observed by a device connected to the Ethernet bus, between the end of a current frame transmission and the start of a new one. This time-slot between transmitted frames is referred to as the 'Interframe Gap'. The actual gap will depend on the mode of communications:

- when communicating at 10 Mbps, the Interframe Gap is 9.6 us
- when communicating at 100 Mbps, the Interframe Gap is 0.96 us.

Frame Deference

When configured for Half Duplex communications, the Controller constantly monitors the Ethernet bus, 'listening' for any traffic (passing frames transmitted by other devices connected to the bus). It does this using a carrier sense signal (PHY_CRS) supplied by the external PHY device. If the bus is currently busy, this signal will go High and the Controller will delay any transmission it was ready to make – that is, it defers to the passing frame already in transmission from some other device.

When the last bit of the frame currently being transmitted on the bus is sent, the carrier sense signal goes Low. At this point, the Controller starts to time the Interframe Gap. If another transmission is made by another device in the meantime, the carrier sense signal will go High again and the Controller will stop its timing of the Interframe Gap until the carrier sense signal becomes Low again.

When configured for Full Duplex communications, the Controller is not sharing the bus with any other devices – it is point-to-point connected with the PHY device. It therefore does not need to listen for traffic on the bus and hence does not use the carrier sense signal. In this mode, all it needs to do is time the Interframe Gap after completing the last bit in a current frame transmission. When the time is reached, it can transmit a subsequent frame.

Collisions

When configured in Full Duplex mode, simultaneous transmission and reception of data frames can be carried out without fear of collision, as there are only two devices connected to each other. However, when configured for Half Duplex communication and many devices are connected to the Ethernet bus, there is potential for collision of data frames – i.e. when one device starts to transmit without having detected another transmission already in progress.

With all devices connected to the bus operating normally (no faulty devices), collisions will normally only occur inside what is referred to as the ‘Collision Window’. This is the window of time between a device starting to transmit a message frame and the furthest connected device on the bus detecting this transmission.

The collision window equates to the round-trip propagation time of the bus and is 64 byte times for a 10/100 Mbps configuration (corresponding to the minimum frame length of 64 Bytes).

After this ‘window’ there is a very low chance of collision, as all other devices on the bus should have detected the transmission and defer to it.

If a collision does occur during the collision window, the PHY device will take the collision signal (PHY_COL) High and the Controller will then continue to transmit the nibble pattern `1010` for a predefined period (3.2 us for 10 Mbps, 0.32 us for 100 Mbps) in order to ensure that the collision is propagated through the system and that the destination device(s) are aware that a collision has occurred and any received data should be discarded. This continued transmission period is known as ‘Jamming’.

At the end of the jamming period, the Controller will attempt to restart transmission of that same message frame, if it detects that the bus is free.

Functional Description

Symbol

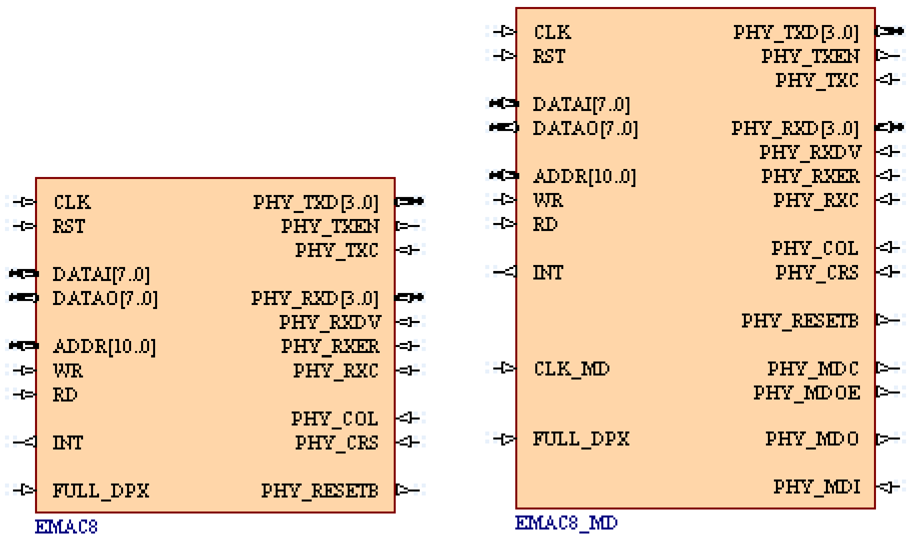

Figure 2. EMAC Controller Symbol – 8-bit non-Wishbone variants (EMAC8, EMAC8_MD)

Pin Description

Table 1. EMAC8, EMAC8_MD pin description

Name	Type	Polarity/ Bus size	Description
Control Signals			
CLK	I	Rise	External (system) clock
RST	I	High	External (system) reset
CLK_MD[1]	I	Rise	External clock signal used for the PHY Register Interface. This signal must be less than 1MHz in frequency.
FULL_DPX	I	Level	Duplex select mode: 0 (i.e. connect to GND) = Half Duplex 1 (i.e. connect to VCC) = Full Duplex **Note**: Using the Controller in Full Duplex mode results in less FPGA resources being used
Host Microcontroller Interface Signals			
DATAI	I	8	Data received from host microcontroller
DATAO	O	8	Data to be sent to host microcontroller
ADDR	I	11	Address bus
WR	I	High	Write enable

[1] EMAC8_MD only.

Name	Type	Polarity/ Bus size	Description
RD	I	High	Read enable
INT	O	High	Interrupt signal. When the corresponding interrupt bits are enabled in the IE register, this signal is used to alert the processor that: • the Transmitter has completed transmission of a frame and is ready to transmit the next frame (provided itx bit is set (IE.0)) • reception of a frame has completed and the Receive Buffer contains valid data to be read (provided irx bit is set (IE.1)) • the MD interface is ready for the next action (provided imd bit is set (IE.2)). This interrupt is only available in the EMAC8_MD
Media Independent Interface (MII) Signals			
PHY_TXD	O	4	Data to be transmitted to the PHY
PHY_TXEN	O	High	Transmit data enable. This signal goes High when transmitting data to the PHY
PHY_TXC	I	Rise	Transmit clock. Used to clock the data that is transmitted to the PHY
PHY_RXD	I	4	Data received from the PHY
PHY_RXDV	I	High	Receive Data Valid signal. This signal goes High if data on the PHY_RXD bus is valid
PHY_RXER	I	High	Receive Error signal. This signal goes High if the PHY detects a receive error
PHY_RXC	I	Rise	Receive clock. Used to clock the data received from the PHY into the Controller
PHY_COL	I	High	Used to flag the detection of a collision
PHY_CRS	I	High	Carrier Sense signal. This signal goes high if the carrier is detected at the network side of the PHY
PHY_RESETB	O	Low	Resets the PHY. This signal is the logical NOT of the RST signal. Therefore resetting the EMACx will cause a reset of the connected PHY device
PHY Register Interface Signals (EMAC8_MD only)			
PHY_MDC	O	Rise	PHY register clock
PHY_MDOE	O	Low	Enable signal for the address/data transmitted from the PHY_MDO pin
PHY_MDO	O	1	Address/Data transmitted to PHY registers
PHY_MDI	I	1	Data received from PHY registers

Hardware Description

Block Diagram

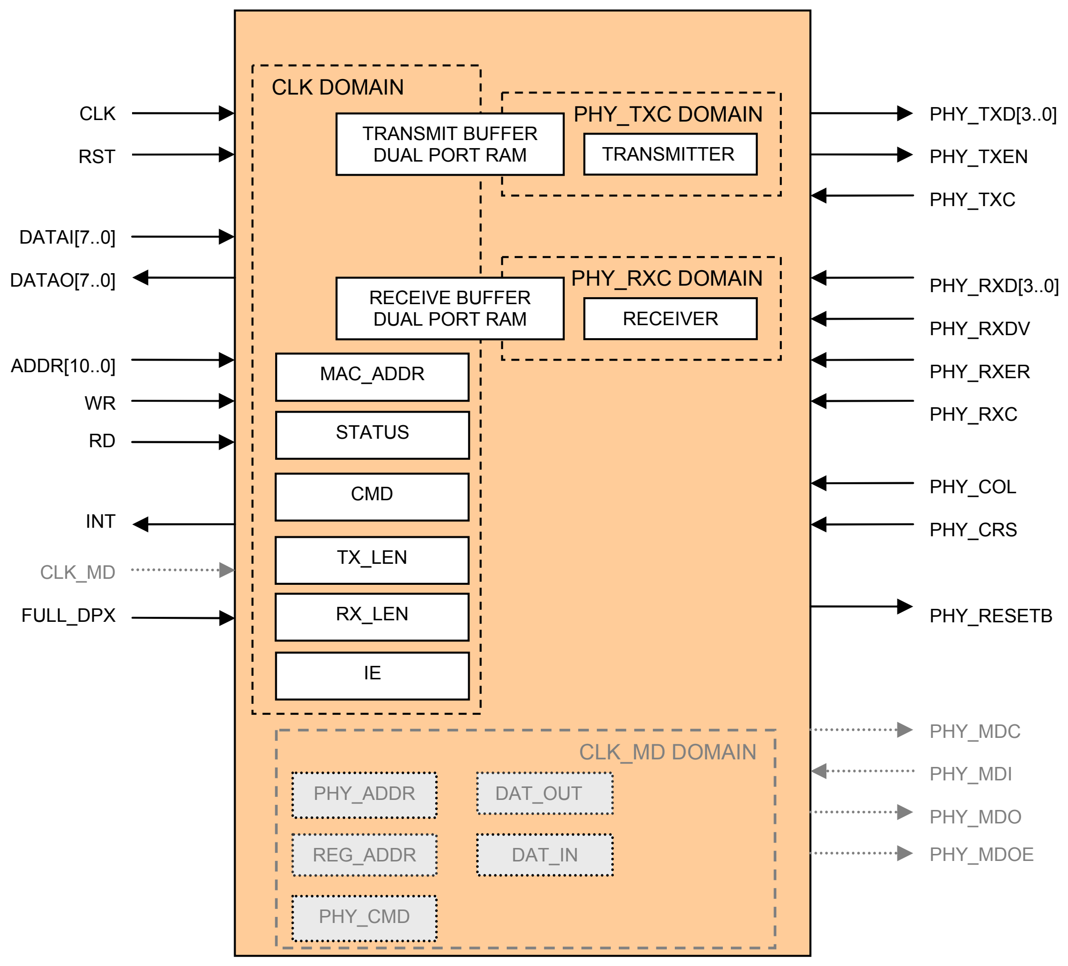

Figure 3. EMAC8, EMAC8_MD block diagram

Figure 3 is drawn primarily to represent the EMAC8 device. The greyed-out areas of the diagram apply to the EMAC8_MD device only.

The internal structure of the Controller consists of memory-mapped registers and two distinct dual port RAM blocks, used for the Transmit and Receive message buffers respectively.

Internal Registers

The following sections detail the internal registers for the EMAC8/EMAC8_MD Controllers.

MAC Address Register (MAC_ADDR)

Address: 5FAh to 5FFh

Access: Write-only

Value after Reset: 000000000000h

This 48-bit register is addressed as six 8-bit locations in memory space. It is used to store the receiver MAC address, which is used by the Receiver to determine if a message on the bus is addressed to it or not (contained within the Destination Address field of the message frame).

This address is not used for transmission. Instead, a separate (and quite possibly different) MAC address must be entered into the Source Address field of a frame to be transmitted.

When writing to the MAC_ADDR register, the most significant byte of the MAC address should be loaded into the least significant byte of the register (at address 5FAh). The least significant byte of the MAC address will therefore be stored in the most significant byte of the register (at address 5FFh).

Status Register (STATUS)

Address: 5F9h

Access: Read-only

Value after Reset: 01h for the EMAC8 and 05h for the EMAC8_MD

This 8-bit register is used to determine the current state of the Controller.

Table 2. The STATUS register

MSB							LSB
-	-	-	-	-	mdr	rxv	txr

Note: Bit 2 of the STATUS register (mdr) is only used for the EMAC8_MD.

Table 3. STATUS register bit functions

Bit	Symbol	Function
STATUS.7	-	Not used. Returns 0 when read
STATUS.6	-	Not used. Returns 0 when read
STATUS.5	-	Not used. Returns 0 when read
STATUS.4	-	Not used. Returns 0 when read
STATUS.3	-	Not used. Returns 0 when read
STATUS.2	mdr	MD Ready flag. Reflects the state of communications with the internal registers of the connected PHY device. 0 = an action is currently being carried out and the interface is not ready for use 1 = the interface is ready for another action to be issued
STATUS.1	rxv	Receive Valid flag. Reflects the current state of the Receiver. 0 = A message frame has not yet completed reception into the Receive Buffer 1 = Reception of a message frame has completed and the Receive Buffer is loaded will valid data to be read
STATUS.0	txr	Transmit Ready flag. Reflects the current state of the Transmitter. 0 = The Transmitter is still sending the current message frame 1 = The Transmitter has completed transmission of the previously loaded message frame and is waiting, ready to transmit a the next frame

Note: For the EMAC8, bit 2 of the STATUS register (mdr) is not used and will return zero when read.

Command Register (CMD)

Address: 5F8h

Access: Read and Write

Value after Reset: 00h

This 8-bit register is used to control the start of transmission and reception for the Controller.

Table 4. The CMD register

MSB							LSB
-	-	-	-	-	-	srx	stx

Table 5. CMD register bit functions

Bit	Symbol	Function
CMD.7	-	Not used. Returns 0 when read
CMD.6	-	Not used. Returns 0 when read
CMD.5	-	Not used. Returns 0 when read
CMD.4	-	Not used. Returns 0 when read
CMD.3	-	Not used. Returns 0 when read
CMD.2	-	Not used. Returns 0 when read
CMD.1	srx	Start Receive bit. Set High to start reception of the next message frame
CMD.0	stx	Start Transmit bit. Set High to start transmission of the next message frame

Transmit Message Length Register (TX_LEN)

Address: 5F6h to 5F7h

Access: Write-only

Value after Reset: 0000h

This 16-bit register is addressed as two 8-bit locations in memory space. It is used to store the length (in Bytes) of the message frame to be transmitted. This value is used by the Transmitter to determine the number of bytes of data to be read from the Transmitter Buffer, when creating the message frame. The value written to this register is the total of the following individual field lengths:

- Destination Address field – constant value of 6 Bytes
- Source Address field – constant value of 6 Bytes
- Length/Type field – constant value of 2 Bytes
- Data field – variable value in the range 0 to 1500 Bytes.

The Preamble, SFD, Padding and FCS field lengths are not considered as part of this value for the Transmit length.

The maximum value for the Transmit length is 1514 Bytes, which equates to only 11 bits of the TX_LEN register being used. Bits 7..0 of the length value is written to memory location 5F7h, while bits 10..8 are written to the three least significant bits of memory address 5F6h.

Note: The TX_LEN register is mapped to the same address as the RX_LEN register. Provided you are performing a write (WR input High), you will access the TX_LEN register.

Receive Message Length Register (RX_LEN)

Address: 5F6h to 5F7h

Access: Read-only

Value after Reset: 0000h

This 16-bit register is addressed as two 8-bit locations in memory space. It is used to store the length (in Bytes) of the received message frame. This value is used by the processor to determine the number of bytes of data to be read from the Receive Buffer. The value read from this register is the total of the following individual field lengths:

- Destination Address field – constant value of 6 Bytes
- Source Address field – constant value of 6 Bytes
- Length/Type field – constant value of 2 Bytes
- Data field – variable value in the range 0 to 1500 Bytes
- Padding field – variable value in the range 0 to 46 Bytes

The Preamble and SFD fields are automatically stripped from the frame upon reception and the FCS field length is not considered as part of this value for the Receive length.

The maximum value for the Receive length is 1514 Bytes, which equates to only 11 bits of the RX_LEN register being used. Bits 7..0 of the length value are written to memory location 5F7h, while bits 10..8 are written to the three least significant bits of memory address 5F6h. The upper five bits of the RX_LEN register (bits 7..3 of address 5F6h) are filled with zeros.

Note: The RX_LEN register is mapped to the same address as the TX_LEN register. Provided you are performing a read (RD input High), you will access the RX_LEN register.

Interrupt Enable Register (IE)

Address: 5F5h

Access: Read and Write

Value after Reset: 00h

This 8-bit register is used to individually enable interrupt generation from the Transmitter, Receiver and, in the case of the EMAC8_MD, the MD interface of the Controller.

Table 6. The IE register

MSB							LSB
-	-	-	-	-	imd	irx	itx

Note: Bit 2 of the IE register (imd) is only used for the EMAC8_MD.

Table 7. IE register bit functions

Bit	Symbol	Function
IE.7	-	Not used. Returns 0 when read
IE.6	-	Not used. Returns 0 when read
IE.5	-	Not used. Returns 0 when read
IE.4	-	Not used. Returns 0 when read
IE.3	-	Not used. Returns 0 when read
IE.2	imd	MD Interrupt Enable. Set this bit High to enable generation of an interrupt when the MD interface becomes ready
IE.1	irx	Receive Interrupt Enable. Set this bit High to enable generation of an interrupt when reception of a message frame has completed and the Receive Buffer is full of valid data, ready to be read
IE.0	itx	Transmit Interrupt Enable. Set this bit High to enable generation of an interrupt when the Transmitter has completed transmission of the previously loaded message frame and is waiting ready to transmit the next frame

Note: For the EMAC8, bit 2 of the IE register (imd) is not used and will return zero when read.

PHY Address Register (PHY_ADDR) – EMAC8_MD only

Address: 5F0h

Access: Read and Write

Value after Reset: 01h

This 8-bit register is used to load the address of the PHY Controller device connected to the EMACx Controller. The PHY address is five bits in length. The high-order three bits of this register are therefore unused and will return ‘0’ when read.

PHY Internal Register Address Register (REG_ADDR) – EMAC8_MD only

Address: 5F1h

Access: Read and Write

Value after Reset: 00h

This 8-bit register is used to load the unique address of an internal register of the connected PHY device that you wish to write to/read from. The internal register address is five bits in length. The high-order three bits of this register are therefore unused and will return ‘0’ when read.

PHY Command Register (PHY_CMD) – EMAC8_MD only

Address: 5F4h

Access: Read and Write

Value after Reset: 00h

This 8-bit register is used to control the writing to/reading from internal registers of the connected PHY Controller device. **Note**: This register can only be written to if no other action is pending.

Table 8. The PHY_CMD register

MSB							LSB
-	-	-	-	-	-	rrd	rwr

Table 9. PHY_CMD register bit functions

Bit	Symbol	Function
PHY_CMD.7	-	Not used. Returns 0 when read
PHY_CMD.6	-	Not used. Returns 0 when read
PHY_CMD.5	-	Not used. Returns 0 when read
PHY_CMD.4	-	Not used. Returns 0 when read
PHY_CMD.3	-	Not used. Returns 0 when read
PHY_CMD.2	-	Not used. Returns 0 when read
PHY_CMD.1	rrd	Register Read bit. Set High to read from the addressed internal PHY register
PHY_CMD.0	rwr	Register Write bit. Set High to write to the addressed internal PHY register

PHY Data Output Register (DAT_OUT) – EMAC8_MD only

Address: 5F2h to 5F3h

Access: Write-only

Value after Reset: 0000h

This 16-bit register is addressed as two 8-bit locations in memory space. It is used to store the data to be written to the accessed internal register of the connected PHY Controller device.

Note: The DAT_OUT register is mapped to the same address as the DAT_IN register. Provided you are performing a write (WR input High), you will access the DAT_OUT register.

PHY Data Input Register (DAT_IN) – EMAC8_MD only

Address: 5F2h to 5F3h

Access: Read-only

Value after Reset: 0000h

This 16-bit register is addressed as two 8-bit locations in memory space. It is used to store the data received from the accessed internal register of the connected PHY Controller device.

Note: The DAT_IN register is mapped to the same address as the DAT_OUT register. Provided you are performing a read (RD input High), you will access the DAT_IN register.

Internal Memory Buffers

The following sections detail the memory buffers used for the Transmit and Receive sections of the EMACx Controller.

Transmit Buffer

Address: 000h to 5EDh

Access: Write-only

Value after Reset: Unknown

The message data to be transmitted is written to and stored in, FPGA Dual Port Block RAM. The size of the RAM will depend on the physical FPGA device used.

The message to be sent is written using one port of the RAM block, configured for byte-wide addressing. The message must consist of the following components, written to the specified (and contiguous) memory locations within the buffer:

- 000h to 005h – 6-Byte Destination Address
- 006h to 00Bh – 6-Byte Source Address
- 00Ch to 00Dh – 2-Byte Length/Type field
- 00Eh to 5EDh – the main message Data (from 0-1500 Bytes)

The message written to the Transmit Buffer does not include the Preamble, SFD, Padding or CRC fields, as these will be automatically generated by the Controller.

When transmitting, the stored message data is read from the second port of the RAM block, which is configured for nibble-wide addressing (since transmission to the connected PHY device is in nibbles).

Receive Buffer

Address: 000h to 5EDh

Access: Read-only

Value after Reset: Unknown

Received message data is written to and stored in, additional FPGA Dual Port Block RAM. Again, the size of RAM will depend on the particular FPGA device to which the design is targeted.

The received message from the connected PHY device is written using one port of the RAM block, configured for nibble-wide addressing (since data is received from the PHY in nibbles).

When the processor reads the message, the stored message data is read using the second port of the RAM block, which is configured for byte-wide addressing. The message consists of the following components, read from the specified (and contiguous) memory locations within the buffer:

- 000h to 005h – 6-Byte Destination Address
- 006h to 00Bh – 6-Byte Source Address
- 00Ch to 00Dh – 2-Byte Length/Type field
- 00Eh to 5EDh – the main message Data (from 0-1500 Bytes)

The received message will also contain Padding bytes (if applicable) and the CRC.

Writing to the Internal Registers and Transmit Buffer

To write to an internal register or the Transmit Buffer, put the desired value on the DATAI bus and the 11-bit address for the register/buffer on the ADDR input bus. Assert the WR signal for a minimum of one period of the external system clock signal (CLK). The value for the register/buffer at that address will be updated.

Reading from the Internal Registers and Receive Buffer

To read from an internal register or the Receive Buffer, put the 11-bit address for the register/buffer on the ADDR input bus and assert the RD signal for a minimum of one period of the external system clock (CLK). The content of the register/buffer at that address will appear on the Controller’s DATAO bus.

Operation

The following sections summarize the steps involved in sending and receiving messages using the EMAC8 and EMAC8_MD variants. For the EMAC8_MD, a section is also included which summarizes the steps required to write to, or read from, the internal PHY device registers.

Initializing the Controller

You will need to re-initialize the Controller, ready for reception of messages, after each external reset. To initialize the Controller:

- write the 6-Byte MAC Address for the Controller to the MAC_ADDR register. Remember that the most significant byte of the address must be loaded into the least significant byte of the register.
- start the receiver by setting the srx bit of the CMD register (CMD.1).

Transmitting a Message

To send a message to the connected PHY device:

- check the txr bit of the STATUS register (STATUS.0). If the Controller is ready to transmit the next message frame, this bit will be High. It is very important to check that the Transmitter has finished sending any previous message frame and is in the ready state, as writing a new message to the Transmit Buffer before the previous message has completed transmission will result in corruption of the previous message
- write to the Transmit Buffer with the message that you wish to transmit. Remember that this should include Destination Address, Source Address, Length/Type and Data fields, with all corresponding bytes loaded contiguously into memory
- write the length of the message to the TX_LEN register. This is the length, in Bytes, of the message you have just written into the Transmit Buffer
- start the Transmitter by setting the stx bit of the CMD register (CMD.0)

The Controller will construct the frame for the message, in accordance with the Ethernet protocol. If configured in Half-Duplex Mode, the Controller will listen to see whether it can transmit – by monitoring its PHY_CRS input. Provided this input is Low, there is no current activity on the Ethernet bus and transmission can go ahead. Transmission is carried out, one nibble at a time, from the PHY_TXD output on each rising edge of the transmit clock provided by the PHY device (arriving at the PHY_TXC input).

When the Controller has completed the transmission, the PHY_TXEN signal will go Low and the txr bit in the STATUS register will go High, signifying that the Transmitter is ready to send the next message frame. If you have enabled the itx bit in the Interrupt Enable register (IE.0), an internal interrupt flag for the transmitter will go High, generating an active interrupt output at the INT pin of the device (INT_O for Wishbone-compliant versions). The interrupt will be cleared when the STATUS register is read or the device is reset.

Receiving a Message

To receive a message that has been transmitted from the connected PHY device to the EMACx Controller:

- check the rxv bit of the STATUS register (STATUS.1). If the Controller has completed reception of a valid message frame, that message will be loaded into the Receive Buffer and this bit will be High
- read the value for the length of the received message from the RX_LEN register. This is the length, in Bytes, for the message currently loaded into the Receive Buffer and includes the Destination Address, Source Address, Length/Type, Data and Padding fields (where applicable)
- read the message from the Receive Buffer. This will comprise of RX_LEN Bytes of contiguous message data
- start the receiver by setting the srx bit of the CMD register (CMD.1). It is important to ‘release’ the Receive Buffer after the processor has read the message, otherwise no further messages will be received.

When the Receiver has completed reception of a message frame, the PHY_RXDV signal will go Low and the rxv bit in the STATUS register will go High, signifying that the Receiver is ready to receive the next message frame. If you have enabled the irx bit in the Interrupt Enable register (IE.1), an internal interrupt flag for the receiver will go High, generating an active interrupt output at the INT pin of the device (INT_O for Wishbone-compliant versions). The interrupt will be cleared when the STATUS register is read or the device is reset.

Communicating with Internal PHY Registers

The following two sections detail how to write to and read from, an internal register in the connected PHY device respectively, when using the EMAC8_MD device.

Note: Only one action – Read or Write – can be started at a time. If you give the command to both write to and read from an internal PHY register, the read command will be accepted and proceed, while the write command will be ignored.

Writing to a PHY Register

To write to an internal register of the PHY device:

- check the mdr bit of the STATUS register (STATUS.2). If the MD interface is ready for another action, this bit will be High. This check is an important one, as writing to either of the address registers (PHY_ADDR, REG_ADDR) or the data register (DAT_OUT) while a command is pending can give unexpected results
- write the address of the PHY Controller device to the PHY_ADDR register
- write the address of the internal PHY register that you wish to write to, to the REG_ADDR register
- write the data that you wish to send to the addressed internal PHY register, into the DAT_OUT register
- start the Write command by setting the rwr bit of the PHY_CMD register (PHY_CMD.0).

At the end of the transmission, the mdr bit in the STATUS register will go High, signifying that the MD interface is ready for the next operation. If you have enabled the imd bit in the Interrupt Enable register (IE.2), an internal interrupt flag for the MD interface will go High, generating an active interrupt output at the INT pin of the device (INT_O for Wishbone-compliant versions). The interrupt will be cleared when the STATUS register is read or the device is reset.

Reading from a PHY Register

To read from an internal register of the PHY device:

- check the mdr bit of the STATUS register (STATUS.2). If the MD interface is ready for another action, this bit will be High. This check is an important one, as writing to either of the address registers (PHY_ADDR, REG_ADDR) while a command is pending can give unexpected results
- write the address of the PHY Controller device to the PHY_ADDR register
- write the address of the internal PHY register that you wish to read from, to the REG_ADDR register
- start the Read command by setting the rrd bit of the PHY_CMD register (PHY_CMD.1)
- read the data received from the addressed internal PHY register, from the DAT_IN register.

At the end of the reception, the mdr bit in the STATUS register will go High, signifying that the MD interface is ready for the next operation. If you have enabled the imd bit in the Interrupt Enable register (IE.2), an internal interrupt flag for the MD interface will go High, generating an active interrupt output at the INT pin of the device (INT_O for Wishbone-compliant versions). The interrupt will be cleared when the STATUS register is read or the device is reset.

Revision History

Date	Version No.	Revision
12-Feb-2009	1.0	Initial document release
30-Aug-2011	-	Updated template.