A **network** is a collection of interfaces that share a physical medium and a data protocol.

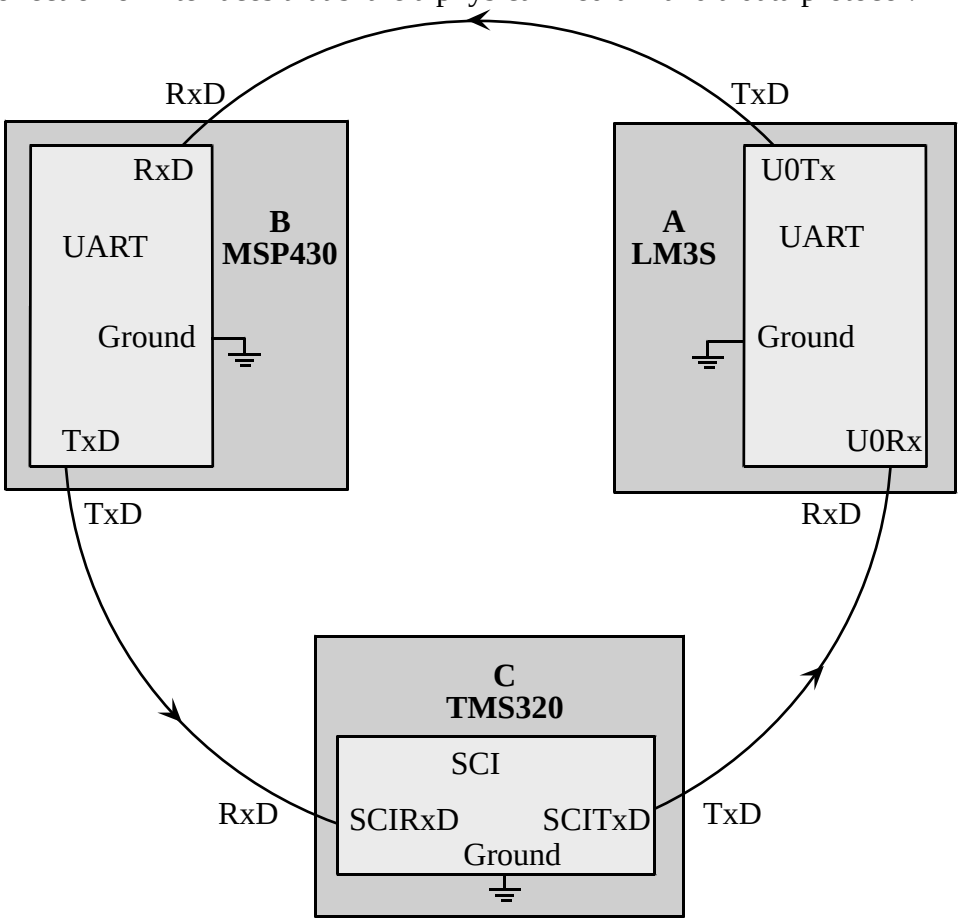

Figure 9.7. A simple ring network with three nodes, linked using the serial ports.

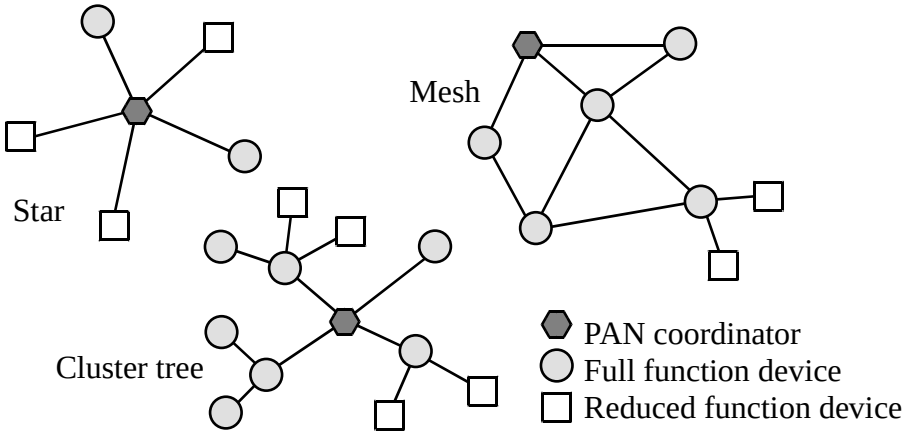

Figure 9.8. ZigBee wireless networks communicate by hopping between nodes.

9.3. Embedded Internet

W. Richard Stevens TCP/IP Illustrated, Volume 1: The Protocols.

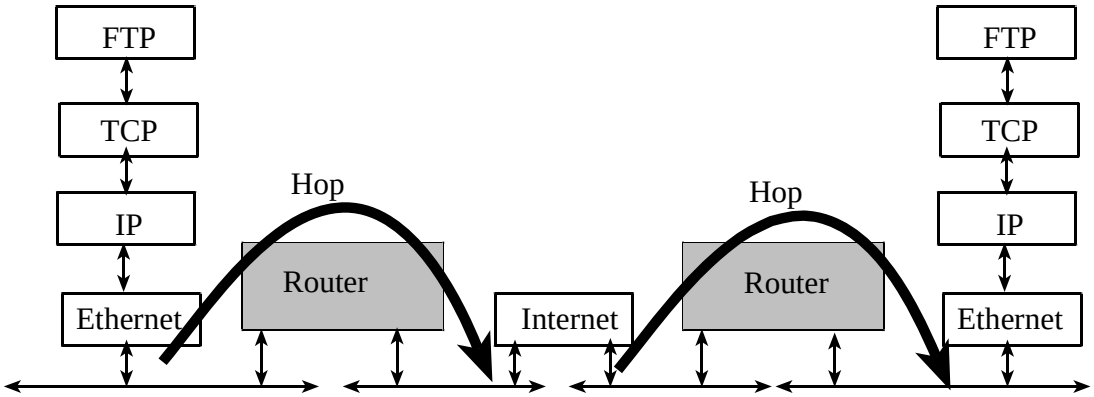

Figure 9.9. Packets on the internet hop from one network to another.

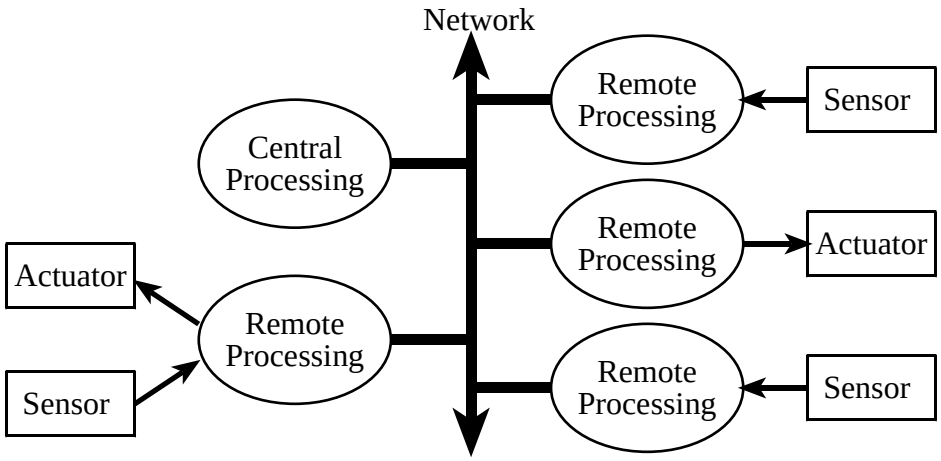

Figure 9.10. Distributed processing places input, output and processing at multiple locations connected together with a network.

abstraction
International Standards Organization (ISO)
Open Systems Interconnection (OSI),

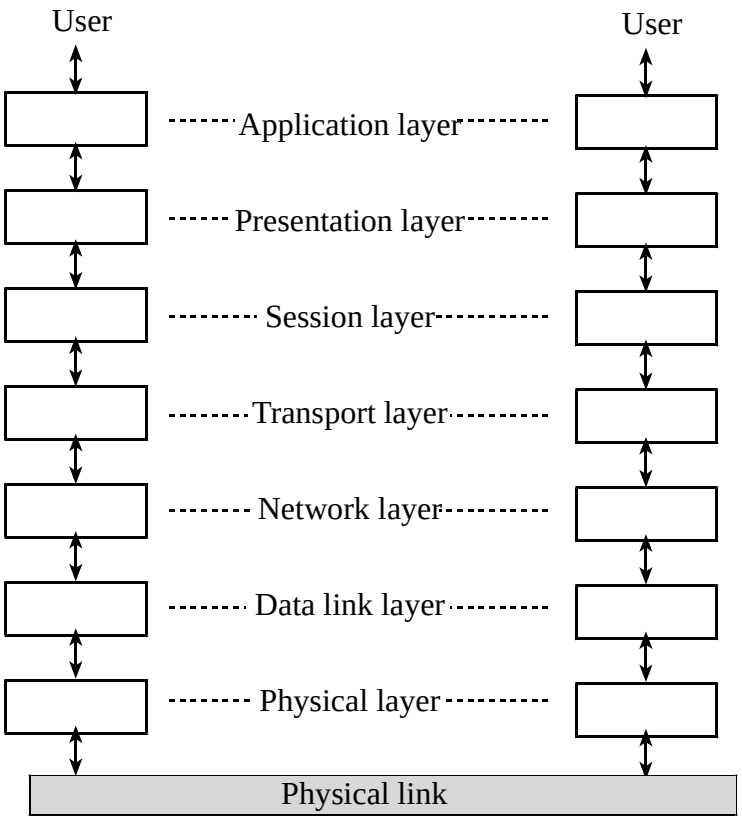

Figure 9.11. The Open Systems Interconnection model has seven layers.

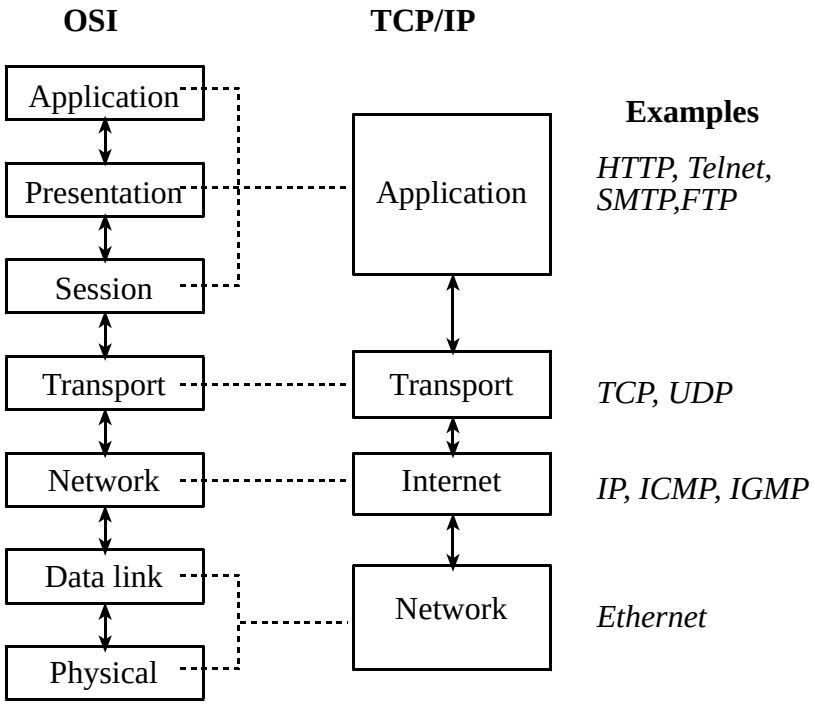

Figure 9.12. The TCP/IP model has four layers.

9.3.2. Message Protocols

46 to 1500 bytes of payload,
Padding
error checking (CRC).
32-bit destination IP address,
Domain Name System (DNS)

Start	*End*	*Number of addresses*
10.0.0.0	10.255.255.255	2^{24}
172.16.0.0	172.31.255.255	2^{20}
192.168.0.0	192.168.255.255	2^{16}

Table 9.1. Private IP addresses.

Because of the growth of the internet, the 32-bit IP address (IPv4) is being replaced with a 128-bit address (IPv6), which will provide for about $3 \cdot 10^{38}$ addresses.

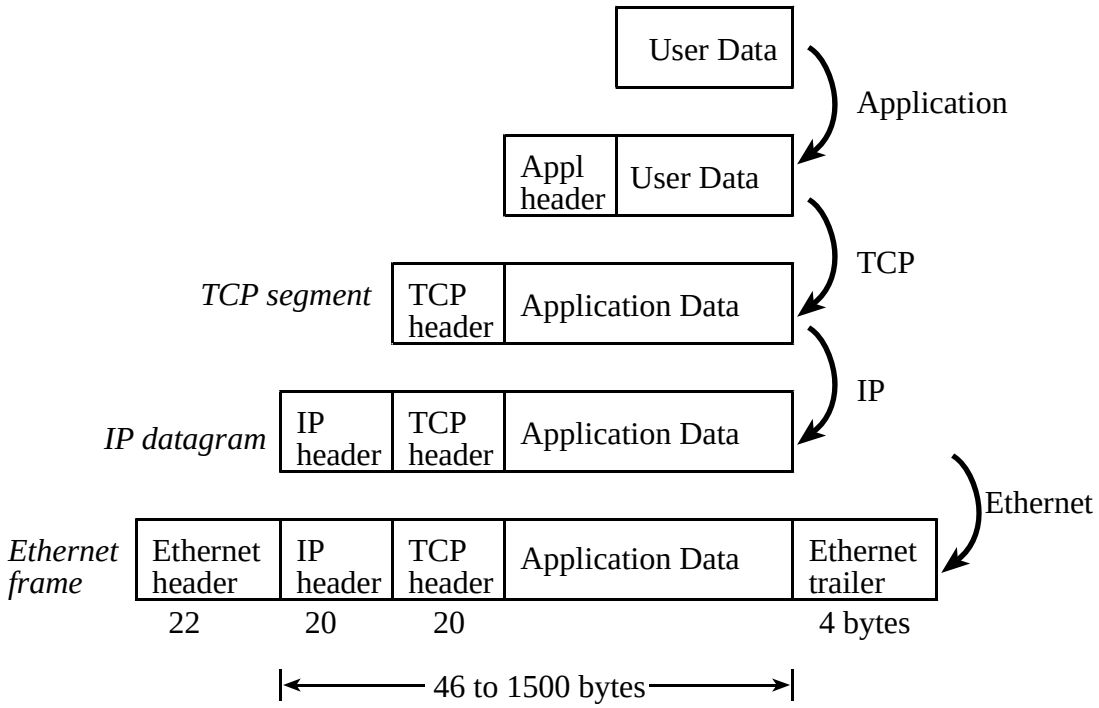

Figure 9.13. Overview of message packets used at various layers.

9.3.3. Ethernet Physical Layer

10BASE-T
100BASE-TX,
1000BASE-T.
Ethernet switch

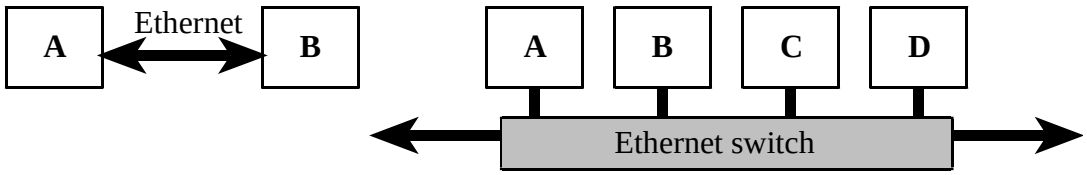

Figure 9.14. Ethernet has a bus-based topology.

Hubs switches **router**

Pin	*Color*	*Pair*	*Description*
1	white/orange	2	TxData +
2	orange	2	TxData -
3	white/green	3	RecvData +
4	blue	1	Unused
5	white/blue	1	Unused
6	green	3	RecvData -
7	white/brown	4	Unused
8	brown	4	Unused

Table 9.2. Pin assignments on a 568-B Ethernet connector.

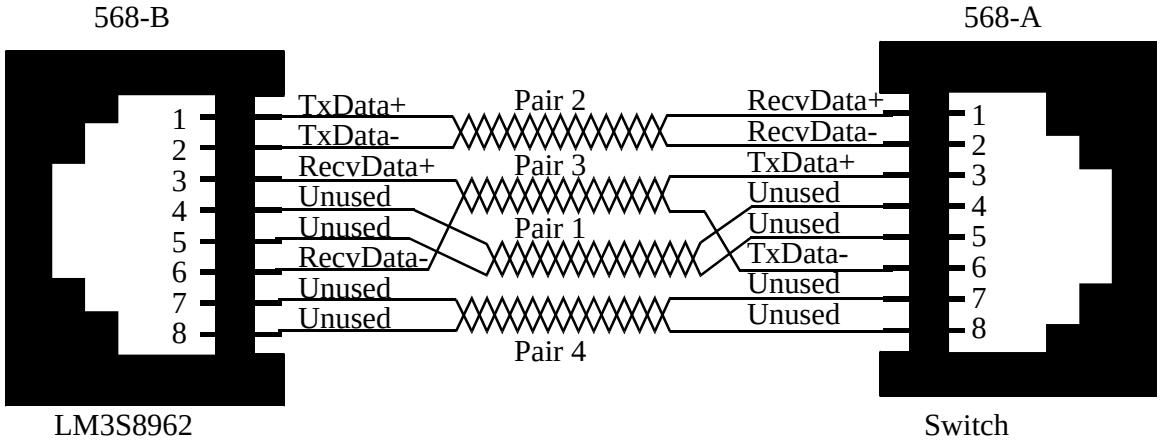

Figure 9.15. Ethernet cable between a microcontroller and a switch.

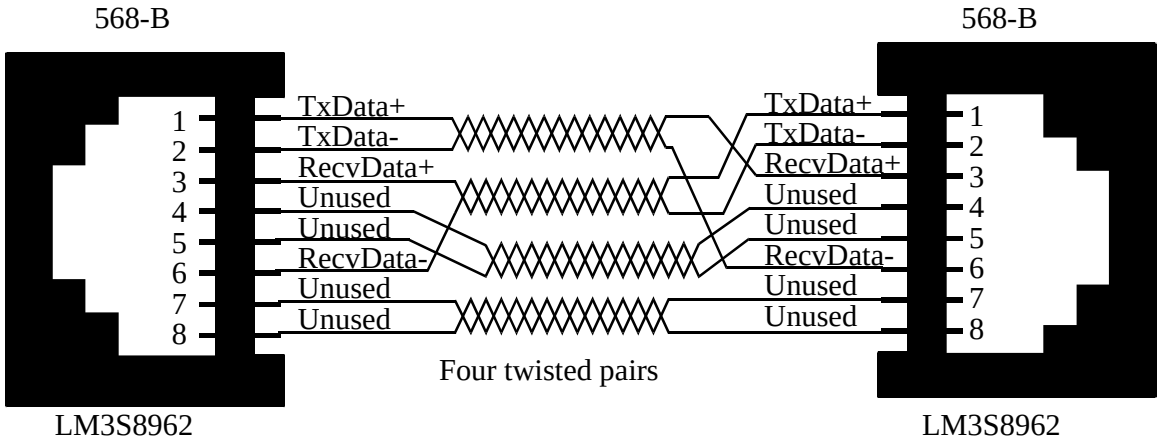

Figure 9.16. Ethernet cable between two microcontrollers.

Pin	*Left color*	*Left signal*	*Cable*	*Right color*	*Right signal*
1	white/orange	TxData +		white/green	TxData +
2	orange	TxData -		green	TxData -
3	white/green	RecvData +		white/orange	RecvData +
4	blue	Unused		blue	Unused
5	white/blue	Unused		white/blue	Unused
6	green	RecvData -		orange	RecvData -
7	white/brown	Unused		white/brown	Unused
8	brown	Unused		brown	Unused

Table 9.3. Pin assignments for a crossover Ethernet cable.

9.3.4. Ethernet on the LM3S8962

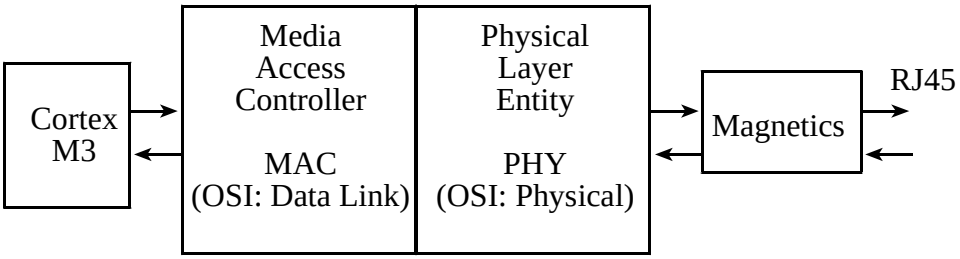

Figure 9.17. The Ethernet port on the Stellaris implements the MAC and PHY layers.

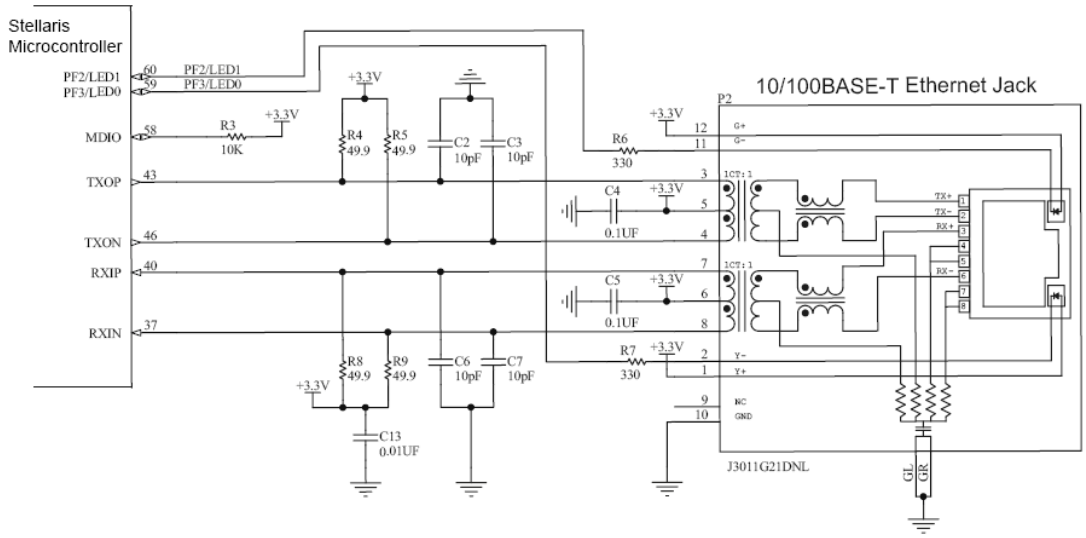

Figure 9.18. Electrical interface between the Stellaris® and the Ethernet cable.

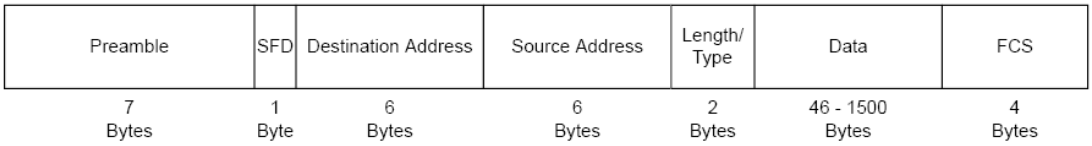

Figure 9.19. An Ethernet frame can hold 46 to 1500 bytes.

Autonegotiation

	31-7	6	5	4	3	2	1	0	Name
$4004.8000		PHYINT	MDINT	RXER	FOV	TXEMP	TXER	**RXINT**	MAC_RIS_R
$4004.8004		PHYINTM	MDINTM	RXERM	FOVM	TXEMPM	TXERM	RXINTM	MAC_IM_R
$4004.8008				**RSTFIFO**	BADCRC	PRMS	AMUL	**RXEN**	MAC_RCTL_R
$4004.800C				DUPLEX		CRC	PADEN	TXEN	MAC_TCTL_R
$4004.801C			THRESH						MAC_THR_R
$4004.8034			NPR						MAC_NP_R
$4004.8038								NEWTX	MAC_TR_R
$4004.803C								TSEN	MAC_TS_R

	31-0	Name
$4004.8010	Data	MAC_DATA_R

	31-24	23-16	15-8	7-0	Name
$4004.8014	MACOCT4	MACOCT3	MACOCT2	MACOCT1	MAC_IA0_R
$4004.8018			MACOCT6	MACOCT5	MAC_IA1_R

	31-8	7-3	2	1	0	Name
$4004.8020		REGADR		WRITE	START	MAC_MCTL_R

	31-8	7-0	Name
$4004.8024		DIV	MAC_MDV_R

	31-16	15-0	Name
$4004.802C		MDTX	MAC_MTXD_R
$4004.8030		MDRX	MAC_MRXD_R

Table 9.4. Ethernet registers. Each register is 32 bits wide. Shaded bits are reserved zero. The bits shown in bold will be used in this section.

```
  MAC_RIS = 0x12;  // clears RXER TXER bits (bits 4,1)
```

Four bytes in **MAC_IA0_R** and two bytes in **MAC_IA1_R** comprise the 48-bit MAC address used to filter incoming frames.

AC-DE-48-00-00-80

first three bytes (AC-DE-48) are the Organizationally Unique Identifier (OUI).

4096 MAC addresses is $625 (http://standards.ieee.org/develop/regauth/iab/)
Individual Address Block (IAB).

To purchase the 2^{24} addresses in an OUI costs $1,825.

FLASH_USERREG0_R and **FLASH_USERREG1_R**)

Let *DIV* be the 8-bit value in **MAC_MDV_R**

$$F_{mdc} = F_{bus}/(2*(DIV+1)$$

```
unsigned long PHYRead(unsigned char ucRegAddr){
  while(MAC_MCTL_R & 0x0001){}; // wait for previous command
  MAC_MCTL_R = (((ucRegAddr<<3) & 0x00F8) | 0x0001);
  while(MAC_MCTL_R & 0x0001){}; // wait for this command
  return(MAC_MRXD_R & 0xFFFF);  // read the result
}
```

```
  do{
    status = PHYRead(1);
  }
  while((status & 0x04) == 0);   // check the Link bit
```

```
void PHYWrite(unsigned char ucRegAddr, unsigned long data){
  while(MAC_MCTL_R & 0x0001){}; // wait for previous command
  MAC_MTXD_R = data & 0xFFFF;   // data to be written
  MAC_MCTL_R = (((ucRegAddr << 3) & 0x00F8) | 0x0003);
  while(MAC_MCTL_R & 0x0001){}; // wait for this command
}
```

The following steps (Program 9.1) can then be used to configure the Ethernet Controller for basic operation.

1) Program the **MAC_DIV_R** register to obtain a 2.5 MHz clock (or less) on the internal MII. Assuming a 50-MHz system clock, the **MAC_DIV_R** value should be 9 or greater.

2) Program the **MAC_IA0_R** and **MAC_IA1_R** register for address filtering.

3) Program the **MAC_TCTL_R** register for Auto CRC generation, padding, and full-duplex operation using a value of 0x16.

4) Program the **MAC_RCTL_R** register to flush the receive FIFO and reject frames with bad FCS using a value of 0x18.

5) Enable both the Transmitter and Receive by setting the LSB in both the **MAC_TCTL** and **MAC_RCTL_R** registers.

```
void MAC_Init(unsigned char *pucMACAddr){
  volatile unsigned long delay; unsigned long ulTemp;
  unsigned char *pucTemp = (unsigned char *)&ulTemp;
  SYSCTL_RCGC2_R |= SYSCTL_RCGC2_GPIOF+
        SYSCTL_RCGC2_EPHY0+SYSCTL_RCGC2_EMAC0; // F and Ethernet
  delay = SYSCTL_RCGC2_R;
  GPIO_PORTF_AFSEL_R |= 0x0C;       // PF3-2, Ethernet LEDs
  GPIO_PORTF_DEN_R |= 0x0C;         // digital I/O on PF3-2
  MAC_RCTL_R &= ~MAC_RCTL_RXEN;     // Disable the receiver
  MAC_TCTL_R &= ~MAC_TCTL_TXEN;     // Disable transmitter
  MAC_MDV_R = (MAC_MDV_R & MAC_MDV_DIV_M)+10; // 2.27 MHz
  pucTemp[0] = pucMACAddr[0];     // set 48-bit MAC address
  pucTemp[1] = pucMACAddr[1];
  pucTemp[2] = pucMACAddr[2];
  pucTemp[3] = pucMACAddr[3];
  MAC_IA0_R = ulTemp;
  ulTemp = 0;
  pucTemp[0] = pucMACAddr[4];
  pucTemp[1] = pucMACAddr[5];
  MAC_IA1_R  = ulTemp;
  MAC_TCTL_R |=  MAC_TCTL_DUPLEX    // duplex mode
                +MAC_TCTL_CRC       // create CRC automatically
                +MAC_TCTL_PADEN;    // zero pad if too short
  MAC_RCTL_R |= MAC_RCTL_RSTFIFO;  // flush receiver
  MAC_RCTL_R |= MAC_RCTL_BADCRC;   // Reject bad FCS
  MAC_RCTL_R |= MAC_RCTL_RXEN;     // Enable the receiver
  MAC_TCTL_R |= MAC_TCTL_TXEN;     // Enable transmitter
  MAC_RCTL_R |= MAC_RCTL_RSTFIFO;  // flush receiver again
}
```

Program 9.1. Ethernet initialization.

To transmit a frame, write the frame into the TX FIFO using the Ethernet MAC Data **(MAC_DATA_R)** register. Then set the NEWTX bit in the Ethernet MAC Transmission Request

(MAC_TR_R) register to initiate the transmit process. When the NEWTX bit has been cleared, the TX FIFO is available for the next transmit frame.

To receive a frame, wait for the NPR field in the Ethernet MAC Number of Packets **(MAC_NP_R)** register to be non-zero. Then begin reading the frame from the RX FIFO by using the **MAC_DATA_R** register. To ensure that the entire packet is received, read the length field from the frame and perform that many reads from the RX FIFO to make sure the entire packet has been read.

This system can be found on the book web page as **Ethernet_8962.zip**.