Freescale Semiconductor
Application Note

AN2692
Rev. 0.2, 9/2004

MC9S12NE64 Integrated Ethernet Controller

By Steven Torres
8/16 Bit System Engineering
Austin, Texas

Introduction

Ethernet connectivity of embedded devices is a growing trend in industrial and consumer applications. Ethernet is a medium of choice because of its competitive performance, relatively low price of implementation, established infrastructure, and interoperability. Ethernet is also easy to use, widely available, and scalable. With Ethernet capability, embedded devices can be connected to the Internet, which allows access to the embedded device from across the world. Figure 1 shows a simplified illustration of an embedded device that is connected, transparently, to a remote host by the Internet.

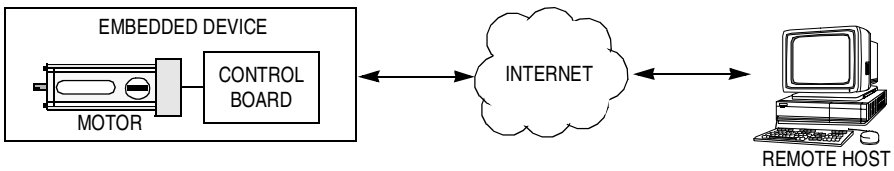


Figure 1. Embedded Device on Internet

This product incorporates SuperFlash® technology licensed from SST.

The MC9S12NE64 is a 16-bit MCU based on Freescale Semiconductor's HCS12 CPU platform. It is the first in a series of low-cost Ethernet-capable MCUs in a small package. The MC9S12NE64 provides a complete, integrated, single-chip Ethernet solution. This application note provides a system overview of the MC9S12NE64 and its integrated Ethernet controller. This discussion also describes how the MC9S12NE64 fits into the network communication model of wired networks. This discussion describes detailed setup for register configuration, initialization, and operation regarding the MC9S12NE64 Ethernet capability; however, Freescale Semiconductor provides a free software driver for the MC9S12NE64 Ethernet controller, which greatly simplifies its setup and use. See EMAC and EPHY sections for details.

Connectivity Example Applications

Embedded devices with Ethernet capability can be implemented in a wide range of applications, including:

- Database data logging or queries
- Web servers for remote embedded devices
- Remote monitoring (data collection/diagnostics)
- Control of remote devices
- Use of email by remote device
- Remote reprogramming of nonvolatile memory (FLASH)

The MC9S12NE64 provides a total solution for systems that:

- Need Ethernet connectivity
- Are end nodes on a network
- Need Ethernet connectivity, but not necessarily the fastest data throughput
- Need a low-cost Ethernet solution
- Need a reduced component count and package size

Ethernet Network Overview

This section provides an overview of Ethernet basics and discusses how the MC9S12NE64 fits into the network communication model of wired Ethernet networks.

Ethernet is commonly used in local area networks (LANs) for device connectivity because it is inexpensive and fast. Ethernet is a technology used in LANs where a group of network devices share a common communications medium. In this application note, the focus is wired LANs because the MC9S12NE64 is designed for wired LANs. The most common communications medium, for wired LANs, is category 5 (cat-5) un-shielded twisted pair (UTP) cable. Using Ethernet technology and the shared communications medium allows the sharing of resources and data among connected devices on the network. Wired Ethernet is becoming a widely used communication tool much like an SCI or USB.

The IEEE 802.3™ standard defines how wired Ethernet works. Ethernet works on the basis that every device connected to the network has a unique hardware address. Because Ethernet uses a shared medium, when one device transmits an Ethernet packet on the network (an Ethernet packet as defined in the IEEE 802.3 standard is shown in Figure 2), every device on the network sees that message. Each device on the network starts to process the packet to determine whether the packet is meant for it by

inspecting the incoming Ethernet packet destination hardware address. Depending on the destination hardware address of the incoming packet, the device will either ignore the message or accept it for further processing.

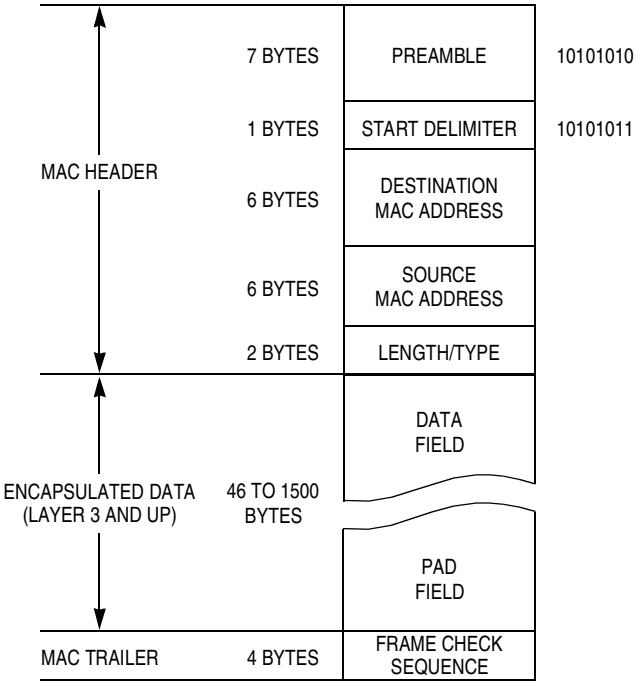


Figure 2. Format and Content of Ethernet Packets

Ethernet Packet

Figure 2 illustrates the specific structure of an Ethernet packet. Figure 2 shows that the data field of an Ethernet packet is encapsulated by a media access controller (MAC) header and frame check sequence (FCS) trailer.

It is important to understand the format and content of Ethernet packets that are transferred between devices in a typical LAN because Ethernet devices operate on these data fields. The data fields are:

- MAC Header
  - Preamble — Seven bytes of an alternating pattern of 1s and 0s that indicates that a frame is coming and provides synchronization.
  - Start of frame delimiter (SFD) — One byte that follows the preamble and consists of an alternating pattern of 1s and 0s that ends with two consecutive 1s.
  - Destination MAC address (or hardware MAC address) — 6-byte value that indicates the MAC hardware address of the network device that should receive the Ethernet packet.
  - Source MAC address (or source hardware address) — 6-byte value that indicates the MAC hardware address of the network device that is sending the Ethernet packet.
  - Length/type — 2-byte value that indicates the number of data bytes that are encapsulated by the Ethernet packet if the length/type value is less-than or equal-to 1500. If the length/type field is greater than 1536, the length/type field identifies the type (or Ethertype) of the packet. The Ethertype identifies the higher-level protocol used to create the encapsulated data portion of an Ethernet packet.

- Encapsulated data portion — 46 to 1500 bytes of user data.
- Frame check sequence trailer (frame check sequence) — 4-byte value that contains a 32-bit cyclic redundancy check (CRC) value.

Ethernet is not the only component of the communication mechanism required for LAN operation. Figure 3 provides a more complete look at the communication model used by a generic user application that uses LAN. The network interface block is where Ethernet is positioned in the LAN communication model. Figure 3 shows that the simplified block version provided is actually derived from a TCP/IP model, which in turn is derived from ISO's (the International Organization for Standardization) OSI (open systems interface model) 7-layer theoretical communications model.

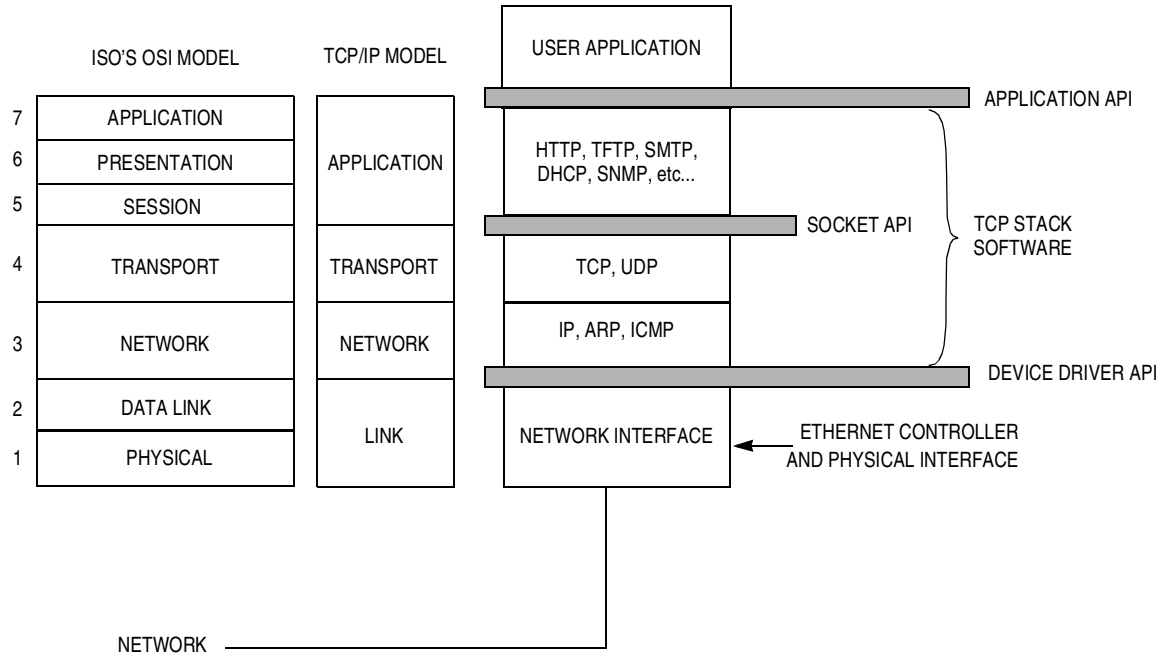


Figure 3. Block Diagram of TCP/IP Stack Model

The functionality that the data link layer and the physical layer provides to the LAN communication model is very important. These layers work together to provide access to the analog world of the UTP cable. The following bullets list some of the basic functions performed by the data link layer and the physical layer.

- Data link layer — MAC
  - Packet error checking
  - Data framing
  - Network access
- Physical layer — PHY
  - Analog signaling

This document will provide a brief overview of TCP/IP stack software, but TCP/IP stack software is not the focus of this document.

MC9S12NE64 Integrated Ethernet Controller

This section will introduce the MC9S12NE64 and provide an overview of the MC9S12NE64 integrated Ethernet controller. The discussion will also include a summary of the minimum number of printed circuit board (PCB) components required in an MC9S12NE64 system to enable its Ethernet capability.

MC9S12NE64

The MC9S12NE64 includes 8K of RAM and 64K of FLASH. It also has other standard on-chip peripherals, including two asynchronous serial communications interface modules (SCIs), a serial peripheral interface (SPI), an inter-integrated circuit module (IIC), a 4-channel/16-bit timer module (TIM), an 8-channel/10-bit analog-to-digital converter (ATD), and pins available as keypad wake-up inputs (KWUs). In addition, in the 112-pin package, an expanded bus that is specified for operation at 16 MHz[1] is available.

Integrated Ethernet Controller

The MC9S12NE64 introduces a new peripheral for the HCS12 CPU platform, an integrated Ethernet controller. The MC9S12NE64 integrates an Ethernet controller that includes a MAC and PHY in one die with the CPU, memory, and other HCS12 standard on-chip peripherals.

The MC9S12NE64 can be targeted at low-throughput connectivity applications that use a 3.3-V external power supply. With an on-chip bandgap-based voltage regulator (VREG), an internal digital supply voltage of 2.5 V (V_{DD}) can also be generated.

A block diagram of the MC9S12NE64 is provided in Figure 4. More information on the MC9S12NE64 is available from the Freescale Semiconductor website: http://freescale.com.

1. At a 16-MHz internal bus speed, the MC9S12NE64 integrated Ethernet controller is limited to 10-Mbps operation. A 25-MHz internal bus speed is required for 100-Mbps operation.

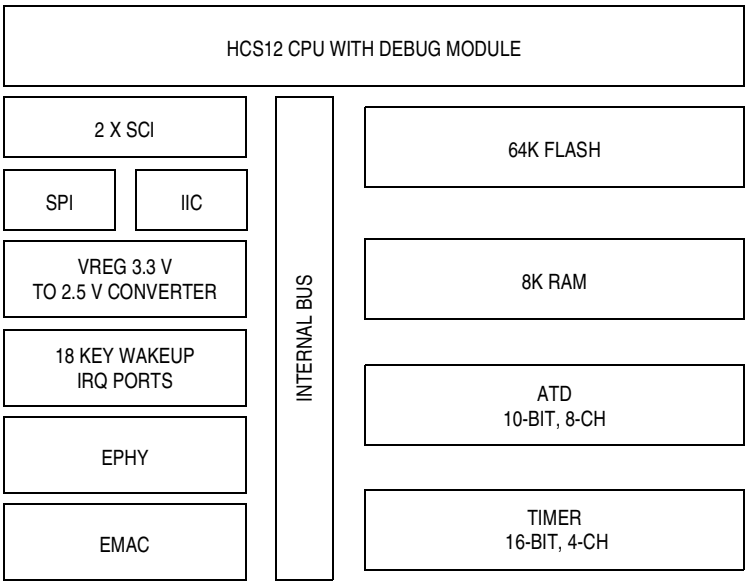


Figure 4. Block Diagram of the MC9S12NE64

MC9S12NE64 System Overview

The MC9S12NE64 is a single-chip Ethernet solution. Having an on-chip CPU, FLASH, RAM, MAC, and PHY reduces the cost of implementing an embedded device with Ethernet connectivity, because no active external components are required. The components required to enable the MC9S12NE64 and its Ethernet interface are:

- MC9S12NE64 MCU
- 25-MHz clock
- 3.3-V power supply
- High-speed LAN magnetics isolation module (available in integrated RJ45 connectors)
- RJ45 connector
- Capacitors and resistors
- Optional: PHY status LEDs (available in integrated RJ45 connectors)
- Optional: BDM connector

Figure 5 provides an illustration of the MC9S12NE64 minimum system circuit implementation using the MC9S12NE64 80-pin package.

NOTE

For basic operation of the MC9S12NE64 Ethernet controller, a 25-MHz input with a tolerance of 25 ppm clock is required. The 25-MHz clock is required to provide the clock input to the integrated PHY for its basic operation. In addition, to operate at 100 Mbps, the internal bus clock must be configured to 25 MHz. For 10 Mbps, an internal bus clock setting of 2.5 MHz is required as a minimum.

To configure the bus clock to 25 MHz with a 25-MHz clock input, the CRG (clock and reset generator) must be configured so that the PLL setting yields the 25-MHz internal bus clock setting. See the Configuring the CRG section for details.

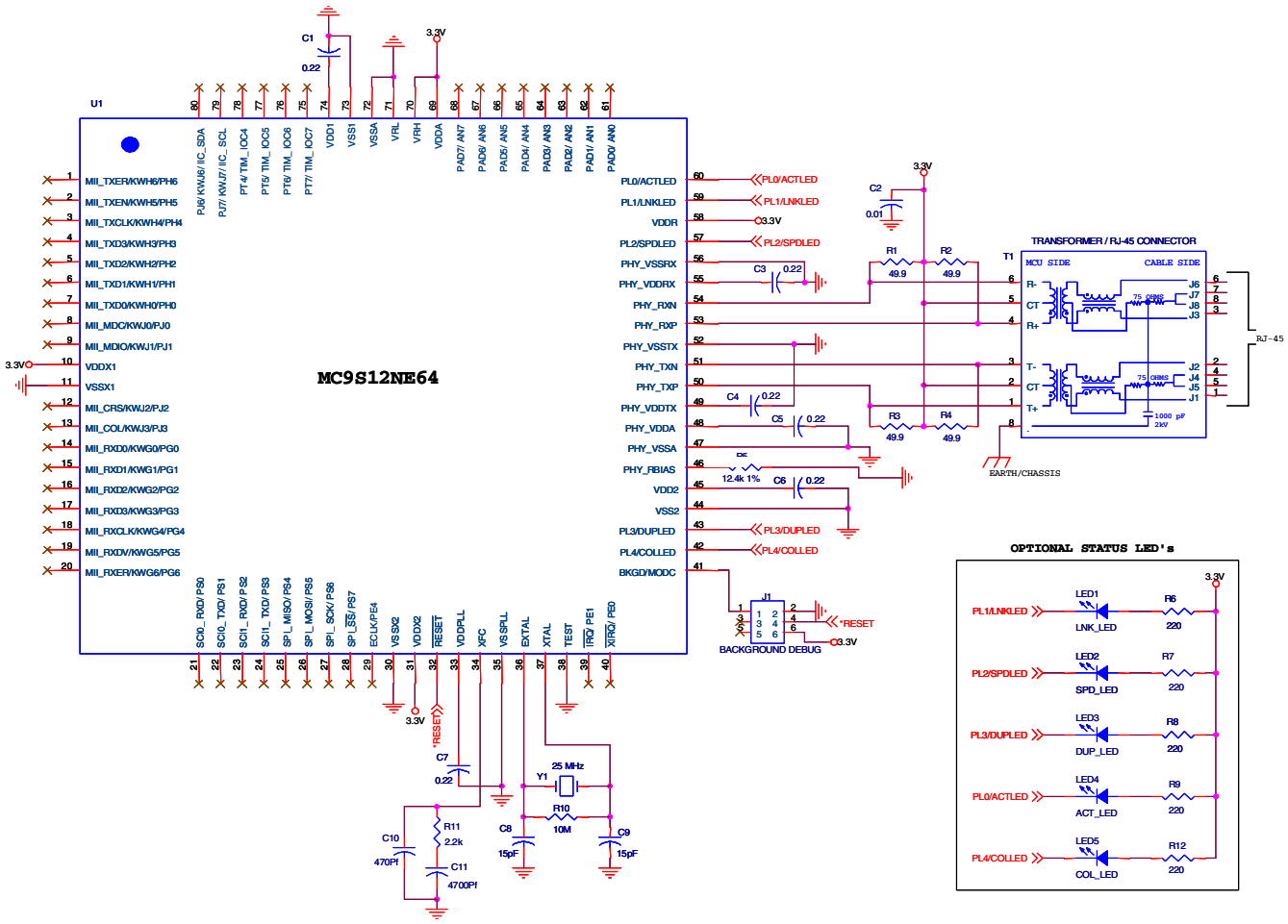


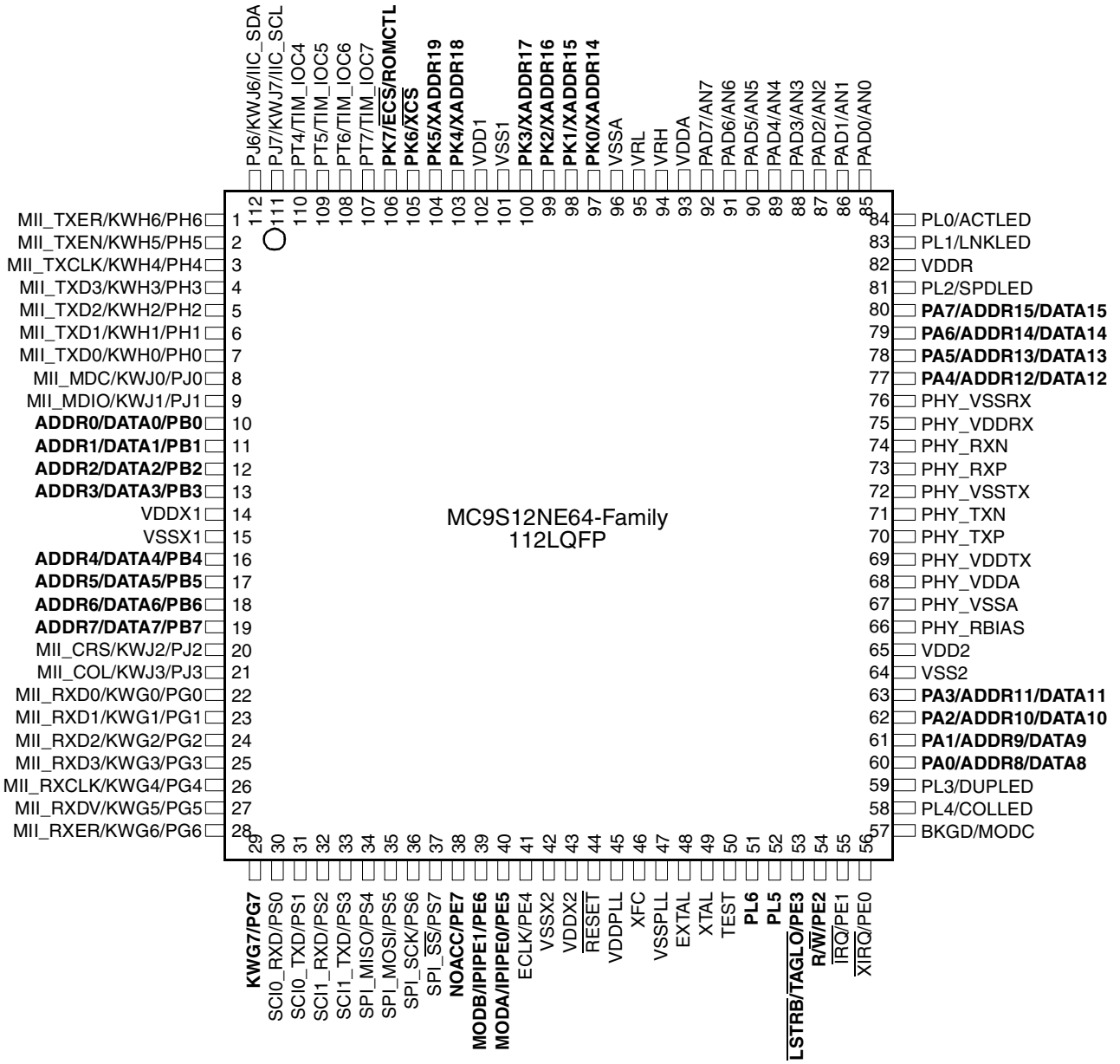
Figure 5. MC9S12NE64 Minimum Web Server Circuit Implementation

MC9S12NE64 Packages

The MC9S12NE64 is available in two packages:

- 112-pin LQFP package — 70 I/O port pins and 10 input-only pins
- 80-pin TQFP-EP package — 38 I/O port pins and 10 input-only pins

The 80-pin TQFP-EP package does not have access to the multiplex address and data bus, and it has an exposed flag for heat dissipation that requires PCB layout accommodation. If the port pins are not bonded out in the chosen package, the user must initialize the registers to be inputs with enabled pull-up resistance to avoid excess current consumption. Figure 6 shows the 112-pin LQFP package. Signals shown in bold are not available on the 80-pin package.



Signals shown in **Bold** are not available on the 80-pin package.

Figure 6. Pinout of MC9S12NE64 112-Pin LQFP Package

CRG and Other Basic MC9S12NE64 Peripherals

An overview of the CRG (clocks and reset generator) and other MC9S12NE64 peripherals is provided in this section. See the MC9S12NE64 block guides and device user guide for details about the CRG and other MC9S12NE64 peripherals. Freescale Semiconductor document AN2552/D also discusses the CRG and several peripherals that are found in the MC9S12E128. In fact, this discussion contains several excerpts from AN2552/D because the MC9S12NE64 and MC9S12E128 share many modules.

Figure 7 illustrates the dependence of the peripherals, core, and memory on the CRG clock outputs, including the core, bus (IP_{Bus}), and oscillator clocks.

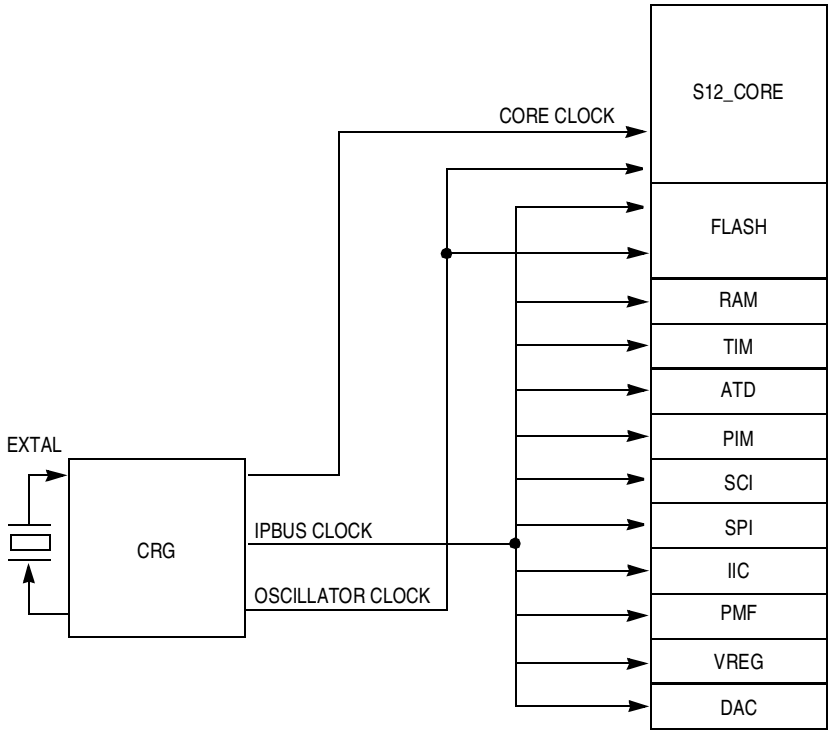


Figure 7. MC9S12NE64 Clock Tree

The modules shown in Figure 7 are discussed in this section. A discussion of how each module uses the clock signals from the CRG is provided.

Configuring the CRG

The source code for configuring the CRG and other peripherals can be created automatically with Processor Expert™, which is a stand-alone utility of Unis that is available as a plug-in for CodeWarrior software. The developer can also manually write code and determine the register settings for a specific configuration. The equations used to determine peripheral settings are provided in this section.

The CRG module provides a set of registers that allows the user to control the operation and behavior of the MCU in its various configurations and modes. The CRG registers that affect clocking include:

- SYNR register — Controls the multiplication factor of the PLL.
- REFDV register — Provides a finer granularity for the PLL multiplier steps.
- CLKSEL register — Configures and controls the clock behavior of the stop and wait modes of the MCU. The PLLSEL bit in this register controls whether the system clocks are derived from the PLLCLK or OSCCLK signal.
- PLLCTL register — Controls PLL functionality and other CRG functions. The PLLON bit in this register turns on the PLL circuitry.

These registers can be used to configure the internal bus clock to 25 MHz, which is required by the MC9S12NE64 when operating at 100 Mbps. To configure the internal bus clock to 25 MHz with a 25-MHz clock input, the PLL must be configured and initialized. This section discusses the initialization of the PLL. The PLL can be configured as a frequency multiplier to run the MCU from a different timebase than that of the incoming OSCCLK signal.

To select the timebase from which the system clock (SYSCLK) will be derived, the PLLSEL bit of the CLKSEL register must be configured. SYSCLK can be either the OSCCLK signal coming from the OSC module or the PLLCLK signal coming from the CRG PLL block. SYSCLK then becomes the source from which both the core and the bus clocks are computed. Equation 1 and Equation 2 show the relationship between the SYSCLK signal and the core and the bus clocks.

Equation 1:

$$f_{Core} = f_{SYSCLK}$$

Equation 2:

$$f_{Bus} = \frac{f_{SYSCLK}}{2}$$

The PLLSEL bit configures the CRG clock PLL switch. If the PLLSEL bit is set, the system clocks are derived from PLLCLK. If the PLLSEL bit is cleared, SYSCLK is derived from OSCCLK. So, depending on the state of the PLLSEL bit, the bus clock can be calculated two different ways:

Equation 3: (In Equation 3, f_{PLL} is the frequency of the PLLCLK signal.)

if PLLSEL = 1, $f_{Core} = f_{PLL}$ and $f_{Bus} = \frac{f_{PLL}}{2}$

if PLLSEL = 0, $f_{Core} = f_{OSCCLK}$ and $f_{Bus} = \frac{f_{OSCCLK}}{2}$

f_{PLL} can be calculated from *Equation 4*.

For MC9S12NE64, the PLLSEL bit must be set. Configuring the PLLCLK signal to the appropriate values to achieve a 25-MHz SYSCLK signal requires configuring the CRG synthesizer register (SYNR) and CRG reference divider register (REFDV). CRG registers include:

- SYNR — 6-bit value that controls the multiplication factor of the PLL. If PLLSEL is asserted, the OSCCLK signal is multiplied by 2 × (SYNR + 1).
- REFDV — 4-bit value that provides a finer granularity for the PLL multiplier steps. If PLLSEL is asserted, the OSCCLK signal is divided by (REFDV + 1).

The SYNR and REFDV values together modify the incoming signal into the PLL block, OSCCLK. The combined contribution of these registers to the value of PLLCLK is shown in Equation 4.

Equation 4:

$$f_{PLL} = \left[\frac{2 \times (SYNR + 1)}{REFDV + 1}\right] \times OSCCLK$$

FLASH Clock

The oscillator clock is the input for the FLASH module. To perform FLASH program or erase operations, the internal FLASH clock frequency ($f_{FLASHCLK}$) must be configured to run between 150 kHz and 200 kHz.

The FLASH uses the PRDIV8 and FDIV[5:0] bits in the FCLKDIV register to divide the oscillator clock down to the required clock range. PRDIV8 is a 1-bit prescaler value that, if set, divides the oscillator clock by 8. If PRDIV8 is clear, the oscillator clock is directly fed into the FCLKDIV divider. The FCLKDIV divider includes a 6-bit value (bits FDIV[5:0]) that generates a divisor from 1 to 64. The divisor is equal-to (FDIV[5:0]+1). Equation 5 is the resulting equation for the FLASH clock frequency ($f_{FLASHCLK}$).

Equation 5:

$$f_{FLASHCLK} = \frac{f_{Bus}}{(8^{PRDIV8} \times (FDIV[5:0]+1))}$$

SCI Clock Baud Rate (BR_{SCI})

The bus clock is the input for the SCI modules. Two asynchronous SCIs are available on the MC9S12NE64. The SCI module version 3.x can be configured to operate in compliance with the IrDA SIR (IrDA) specification.

The SCI uses the SCIxBDH and SCIxBDL. These registers together provide a 13-bit field for SCI baud rate (BR_{SCI}) configuration. SBR[12:0] is used to represent the 13-bit SCI baud rate register value. SBR[12:0] can be set to any value from 0 to 8191. When SBR[12:0] = 0, the SCI baud rate generator is disabled, which reduces system current consumption. For all other SCI baud rate calculations, use Equation 6.

Equation 6:

$$\text{If IREN = 0, } BR_{SCI} = \frac{f_{Bus}}{(16 \times SBR[12:0])}$$

$$\text{If IREN = 1, } BR_{SCIIR} = \frac{f_{Bus}}{(32 \times SBR[12:1])}$$

The MC9S12NE64 SCI also can be operated in IrDA mode. In this mode, the SCI can modulate and demodulate narrow pulse-widths as defined by the IrDA SIR standard. IrDA mode is enabled by setting the IREN bit of the SCIBDH register. Equation 6 shows that the BR_{SCIIR} calculation depends on the IREN bit.

SPI Clock Baud Rate

The synchronous serial peripheral interface (SPI) allows duplex, synchronous serial communication between the MCU and peripheral devices. The only clock source for the SPI module is the bus clock.

SPI baud rate (BR_{SPI}) can be set using the SPI control register 2 (SPCR2). The SPCR2 register has two 3-bit values that contribute to an SPI baud rate divisor, SPPR[2:0] and SPR[2:0]. The first 3-bit value, identified by the SPPR[2:0] bits, can have a value ranging from 1 to 8. The second 3-bit value, identified by the SPR[2:0] bits, can have a value ranging from 2 to 256. The mathematical expression of the SPI baud rate divisor is $[(SPPR[2:0]+1) \times 2^{(SPR[2:0]+1)}]$. Applying this expression to the bus clock yields a formula for the SPI baud rate:

Equation 7:

$$BR_{SPI} = \frac{f_{Bus}}{[(SPPR[2:0]+1) \times 2^{(SPR[2:0]+1)}]}$$

IIC Clock

The IIC bus is a two-wire, bidirectional serial bus that provides a simple, efficient method of data exchange between two devices. The only clock source for the IIC module is the bus clock. The device is designed to operate at speeds up to 100 kbps.

The IIC baud rate is a function of the bus clock divided by the SCL divider. Equation 8 shows how to generate the SCL divider and illustrates the IIC frequency divider register (IICF) relationship to the SCL divider variables.

Equation 8:

$$\text{SCL divider} = \text{mul} \times [2 \times \{2 + \text{SCL2tap} + ((\text{SCL\_TAP} - 1) \times \text{tap2tap})\}]$$

Each of the variables in the SCL divider (MUL, SCI2tap, tap2tap, and SCL_TAP) can be derived from the IICF IBCn bits.

The IIC baud rate is computed by Equation 9.

Equation 9:

$$\text{IIC baud rate} = \frac{f_{Bus}}{(\text{mul} \times [2 \times \{2 + \text{SCL2tap} + (\text{SCL\_TAP} - 1) \times \text{tap2tap})\}]}$$

where: $\text{MUL} = 2^{\text{IBC[7:6]}}$

$\text{tap2tap} = 2^{\text{IBC[5:3]}}$

$\text{SCL\_TAP} = 5 + \text{IBC[2:0]}$, if IBC[2:0] = 6, Tap = 12
if IBC[2:0] = 7, Tap = 15

$\text{SCL2tap} = (2^{\text{IBC[5:3]}} - 2)$, if IBC[5:3] = 0, SCL2tap = 4
if IBC[5:3] = 1, SCL2tap = 4
if IBC[5:3] = 2, SCL2tap = 6

TIM Clock Rate

The only clock source for the TIM module is the bus clock. The TIM clock feeds into the three 4-channel timers with 16-bit counters.

The TIM clock rate, $R_{DesiredTIM}$, is a function of the bus clock divided by the timer prescaler. The timer prescaler includes the PR[2:0] bits of the TSCR2 register. The clock rate is computed by Equation 10.

Equation 10:

$$R_{DesiredTIM} = \frac{f_{Bus}}{2^{(PR[2:0])}}$$

ATD Conversion Clock Frequency

The only clock source for the analog-to-digital (ATD) module is the bus clock. The ATD clock source feeds into a 16-channel ATD converter with 10-bit resolution and sets the ATD conversion clock frequency, $f_{DesiredATD}$. Depending on the power supply voltage applied to the ATD converter, the ATD conversion clock frequency will be in the range from 0.5 MHz to 2 MHz.

The ATD conversion clock frequency is a function of both the bus clock and an ATD prescaler value, PRS[4:0]. The ATD conversion clock frequency can be calculated using Equation 11.

Equation 11:

$$f_{DesiredATD} = \frac{f_{BUS}}{2 \times (PRS[4:0] + 1)}$$

The PRS[4:0] bits are found in the ATD control register 4 (ATDCTL4). PRS[4:0] allows 32 prescaler values to divide the bus clock. Based on the selected bus clock frequency and required system conversion times, the user must choose the prescaler value to achieve the required ATD conversion rate—keeping in mind the minimum and maximum ATD clock range.

RTI Time-Out Period

The real-time interrupt (RTI) function can be used to generate a hardware interrupt at a fixed periodic rate, $f_{DesiredRTI}$. The RTI runs with a gated oscillator clock, OSCCLK. OSCCLK feeds into an RTI prescaler.

The RTI prescaler is controlled using the CRG RTI control register (RTICTL). This register controls the two RTI frequency divide rate components. The RTI prescaler rate select bits (RTR[6:4]) and the RTI modulus counter select bits (RTR[3:0]) are combined to determine the overall RTI frequency divide rate. Use Equation 12 to calculate the RTI periodic rate.

Equation 12:

$$f_{DesiredRTI} = \frac{f_{Bus}}{(RTR[3:0] + 1) \times (2^{(9 + RTR[6:4])})}$$

Note that if RTR[6:4] is equal-to zero, the RTI is disabled. In SCM mode, the RTI operates using f_{SCM}.

COP Time-Out Period

The computer operating properly (COP) is a free-running watchdog timer. It enables the user to check whether a program is running and sequencing correctly. Like the RTI, the COP runs with a gated oscillator clock, OSCCLK. Three control bits, CR[2:0], in the COPCTL register allow the selection of seven COP timeout periods, $t_{DesiredCOP}$. Use Equation 13 to calculate $t_{DesiredCOP}$. Note the special cases provided.

Equation 13:

$$t_{DesiredCOP} = t_{Bus} \times 2^{(12 + 2 \times CR[2:0])}$$

Special cases:

- if CR[2:0] = 6, $t_{DesiredCOP} = t_{Bus} \times 2^{(23)}$
- if CR[2:0] = 7, $t_{DesiredCOP} = t_{Bus} \times 2^{(24)}$

Note that if COP[2:0] is equal-to zero, the RTI is disabled. In SCM mode, the RTI operates using f_{SCM}.

MC9S12NE64 and the IEEE 802.3 Standard

The MC9S12NE64 Ethernet controller is compliant with the IEEE 802.3 standard and the 802.3, 802.3u, and 802.3x specifications, so operation can be either 10 Mbps full-duplex, 10 Mbps half-duplex, 100 Mbps full-duplex, or 100 Mbps half-duplex using a medium-independent interface (MII). Flow control and auto-negotiation are also available. Figure 8 provides an overview of the IEEE 802.3 standard.

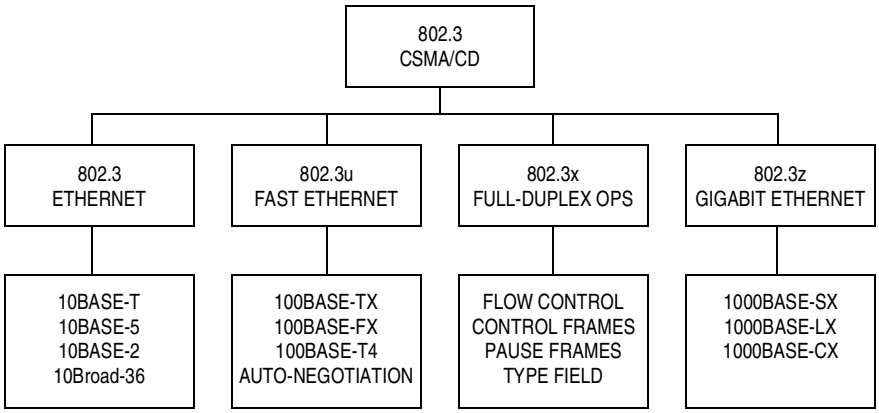


Figure 8. IEEE 802.3 Standard Overview

The IEEE 802.3 specifications that are relevant to the MC9S12NE64 design include:

- MAC — IEEE 802.3, clause 4
- Flow control — IEEE 802.3, clause 31 and annex 31B
- 10BASE-T — IEEE 802.3, clause 14
- 100BASE-TX — IEEE 802.3, clause 24 and 25
- Auto-negotiation — IEEE 802.3, clause 28
- MII — IEEE 802.3, clause 22

The MC9S12NE64 commitment to compliance is also demonstrated through its IEEE 802.3 conformance and interoperability testing at the University of New Hampshire InterOperability Lab's (UNH IOL) Fast Ethernet Consortium. This consortium tested the MC9S12NE64 MAC and PHY against the IEEE 803.2 design specification criteria to ensure that the Ethernet design is both correct and robust. The testing gives assurance that the MC9S12NE64 is fully functional and, more important, interoperates with other network devices from other manufacturers.

MC9S12NE64 Ethernet Media Access Controller (EMAC)

The EMAC is an implementation of a media access controller (MAC) as defined by the IEEE 802.3 Ethernet standard. The EMAC implements the data link layer as described in the communication model shown in Figure 3. The EMAC supports operation at both 10 Mbps and 100 Mbps. The EMAC also implements the IEEE 802.3 medium-independent interface (MII) standard including the MII management interface that can be used to set different PHY options and check PHY status. The EMAC also includes the following features:

- Address recognition and filtering
- Promiscuous mode
- Ethertype filter
- External PHY mode
- Flow control with receive detection
- Two receive buffers with valid frame received detection
- One transmit buffer interface with transmit frame complete detection
- Buffer overrun detection
- Maximum frame size babbling detection
- Frame alignment, CRC, and frame length error detection
- MII serial management transfer detection
- Late collision and excessive collision detection
- Loopback mode

The MC9S12NE64 EMAC provides the mechanism for transferring data from the network layer of one Ethernet-enabled device to another. In the course of these data transfers, the EMAC must also provide data encapsulation, MAC address handling, and error detection. Data encapsulation is the framing, delimiting, and conversion of data from the nibble bit (coming to and from the PHY) to data frames or packets stored in the buffers. In addition, in half-duplex mode, the EMAC provides media access using carrier sense multiple access/collision detect (CSMA/CD) protocol (see the Full Duplex (FDX) section).

Figure 9 is a block diagram of the EMAC module in the MC9S12NE64 with the internal bus clock as the EMAC clock source.

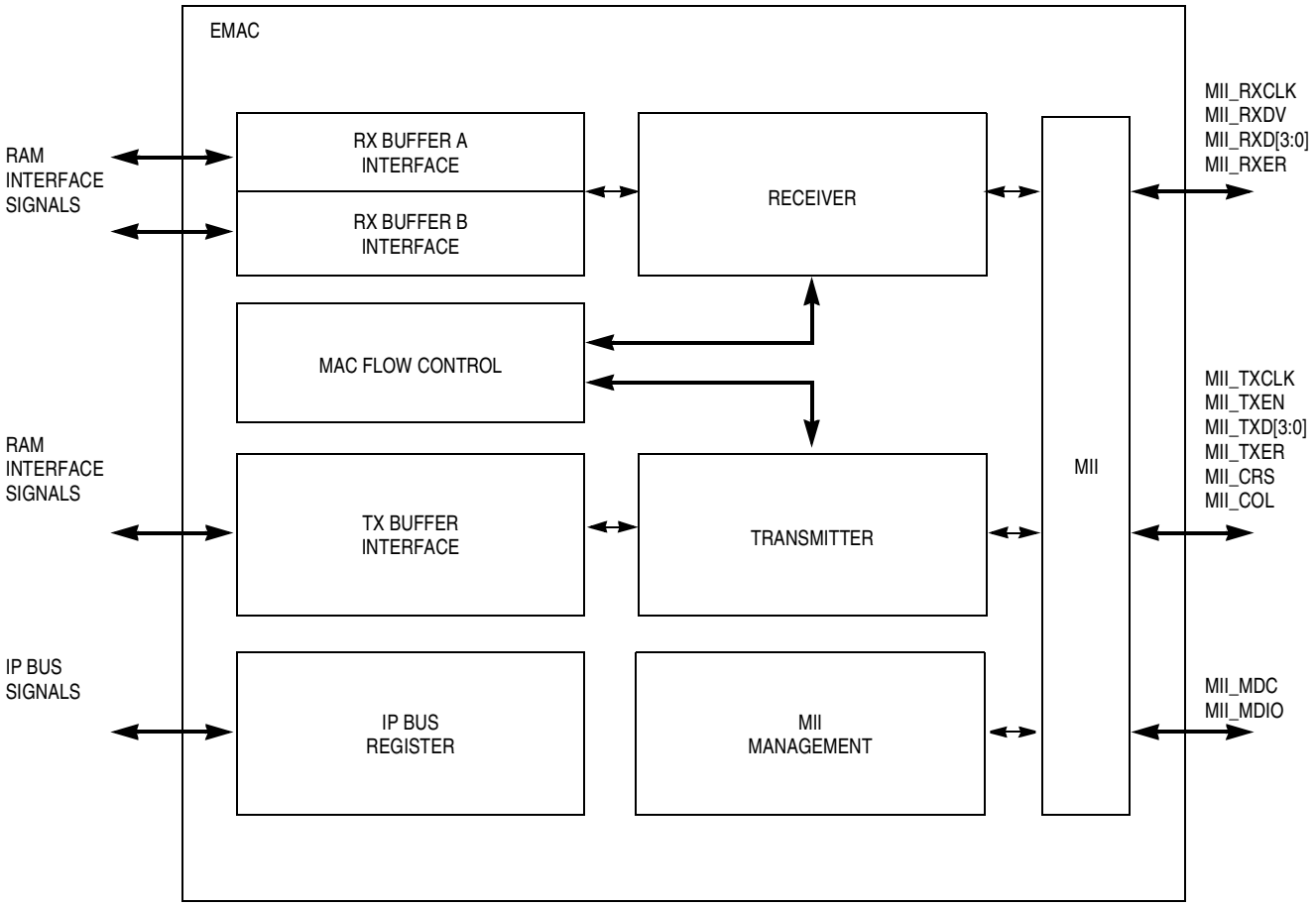


Figure 9. EMAC Block Diagram

The following EMAC options are described in this section:

- Network control
- MAC hardware address
- Address filtering mode
- Ethertype filtering
- Ethernet buffer configuration
- Multicast hash table
- Flow control

Network Control Register (NETCTL)

The network control (NETCTL) register is used mainly to enable the EMAC module and set duplex mode. Figure 10 shows the NETCTL register.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	EMACE	0	0	ESWAI	EXTPHY	MLB	FDX	0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 10. Network Control (NETCTL) Register

EMAC Enable (EMACE)

Setting the EMAC enable (EMACE) bit in the NETCTL register enables the EMAC. Before enabling the EMAC, it is important to configure several other EMAC options. Some of these EMAC options are write-once before the EMAC is enabled; others should be changed only while the EMAC is disabled.

Full Duplex (FDX)

The full duplex (FDX) bit in the NETCTL register configures the EMAC duplex setting. The EMAC can be configured in half-duplex (FDX = 0) or full-duplex (FDX = 1) mode. The duplex setting in the EMAC must be equivalent to the duplex setting in the PHY. If the PHY uses auto-negotiation, the duplex setting of the PHY can be determined when a link is established and auto-negotiation is complete (Figure 22).

In half-duplex mode, the carrier sense multiple access/collision detect (CSMA/CD) protocol is used by the EMAC in half-duplex. Using CSMA/CD provides a mechanism by which two or more network devices can share a common communication medium by allowing only one device to transmit at a time. When more than one network devices transmit simultaneously, their data collide and corrupt the transmission. In full-duplex mode, simultaneous two-way transmissions over point-to-point links are allowed and the CSMA/CD protocol is not required.

External PHY (EXTPHY)

The EXTPHY bit in the NETCTL register can configure the MC9S12NE64 PG[6:0], PH[6:0], and PJ[3:0] pins to operate as an external MII. This allows the user to bypass the internal EPHY of the MC9S12NE64 device for either MII bus testing or to interface to an external PHY. If EXTPHY is set, the EPHY must not be enabled. EXTPHY must be cleared (0) if using the MC9S12NE64 internal EPHY. If EXTPHY is cleared, pins function as general-purpose input/output pins. Table 1 describes the MII interface mode for each MII pin.

Table 1. MII Signal Descriptions

Pin MII Function EXTPHY Bit Set (EXTPHY = 1)	MII Signal Description
MII_RXER	Receive error
MII_RXDV	Receive data valid
MII_RXCLK	Receive clock

Table 1. MII Signal Descriptions (Continued)

Pin MII Function EXTPHY Bit Set (EXTPHY = 1)	MII Signal Description
MII_RXD3	Receive data 3
MII_RXD2	Receive data 2
MII_RXD1	Receive data 1
MII_RXD0	Receive data 0
MII_TXER	Transmit error
MII_TXEN	Transmit data enable
MII_TXCLK	Transmit clock
MII_TXD3	Transmit data 3
MII_TXD2	Transmit data 2
MII_TXD1	Transmit data 1
MII_TXD0	Transmit data 0
MII_COL	Collision detected
MII_CRS	Carrier sense
MII_MDIO	Management data I/O
MII_MDC	Management data clock

Hardware MAC Address

Each network device should have a unique 6-byte media access control (MAC) hardware address that identifies that device on a network. A valid MAC hardware address should be assigned by the developer. This address is used by the datalink layer, which is implemented by the MC9S12NE64 integrated Ethernet controller and low-level drivers. If the device is not connected to a real network, a random MAC hardware address can be used as long as no other device with the same 6-byte MAC hardware address is on the same network.

MAC hardware address groups are assigned by the IEEE EtherType Field Registration Authority. The IEEE-assigned MAC hardware address group is defined by the upper 24-bits of the MAC address. This MAC hardware address group is called the organizational unique identifier (OUI). The lower 24-bits of the MAC address are specified by the vendor as a serial number in most cases. The user must obtain a MAC hardware address group for products that use the MC9S12NE64.

The assigned MAC hardware address must be programmed into the MAC unicast address (MACAD) register in FLASH memory. These registers are one-time writable after reset.

The MACAD registers are shown in Figure 11.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read: Write:	MACAD[47:32]															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read: Write:	MACAD[31:16]															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read: Write:	MACAD[15:0]															
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 11. MACAD Registers

MAC Address Recognition and Filtering

MAC address recognition and filtering can be controlled using PROM, CONMC, and BCREF bits in the receive control and status (RXCTS) register. Figure 12 shows the RXCTS register.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	RXACT	0	0	RFCE	0	PROM	CONMC	BCREJ
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 12. Receive Control and Status (RXCTS) Register

Address filtering is very powerful because it is managed in the EMAC state machine, thus it does not require CPU bandwidth. In fact, while using this feature, unwanted Ethernet packets are blocked and do not cause a flag or interrupt. With the MC9S12NE64, four filtering modes are available: unicast, broadcast, multicast, and promiscuous filtering modes. Unicast, broadcast, and multicast modes can work together, but the promiscuous filtering mode overrides all MAC hardware address recognition and filtering.

Unicast Filter Mode

In unicast filter mode, the EMAC uses the MAC unicast address (MACAD) registers, which contain the unique 6-byte address. The destination address of an incoming packet is compared to the MACAD value. If the incoming frame's destination address field does not match the MACAD value, the frame is rejected. To set the device to unicast filter mode, the PROM bit must be cleared.

Broadcast Filter Mode

A broadcast Ethernet frame is identified by an Ethernet frame with a destination MAC address of FF-FF-FF-FF-FF-FF (6-byte address of all 1s). A network uses the broadcast message format to address a particular message to every device on the network. The address resolution protocol (ARP) uses this mechanism when executing an address request process.

Setting the BCREJ bit of the RXCTS register will block broadcast messages from being accepted by the MC9S12NE64. If the PROM bit is set, all address filtering is overridden and all Ethernet data frames are accepted by the MC9S12NE64, regardless of the destination MAC address.

Multicast Filter Mode and Hash Table

In general, multicast Ethernet packets are used by network devices to address a particular message to devices on the network that belong to a particular group. The developer or network administrator should assign mutlicast groups.

If the value of the left-most bit of the MAC hardware address is 1, the Ethernet packet is considered to be multicast. If the value of the left-most bit is 0, the Ethernet packet is considered to be unicast.

Using the CONMC bit, the MC9S12NE64 can be configured to either accept all incoming Ethernet packets that are defined as multicast, or to accept only those multicast Ethernet packets that match the EMAC multicast hash table.

- CONMC = 0 — All multicast Ethernet messages will be accepted, regardless of the contents of multicast hash table
- CONMC = 1 and PROM = 0 — Only multicast Ethernet packets that pass the multicast hash table test are accepted by the MC9S12NE64

The multicast hash table is implemented by the multicast hash table (MCHASH) registers, MCHASH[63:0]. To initialize MCHASH, the 6-byte destination address of a multicast group must be mapped into 1 of 64 bits of the MCHASH register. The mapping into MCHASH is accomplished by calculating a 32-bit CRC value of the 6-byte destination address and then selecting the six most significant bits of the CRC-encoded result. Next, the MCHASH register bit position that corresponds to the 6-bit value must be set. Then, in multicast filter mode, the EMAC calculates the hash value of an incoming Ethernet packet; if that value matches one of the initialized values, the packet would pass the hash table test and be accepted.

NOTE

The broadcast message is a special case of a multicast message, but the CONMC bit has no effect on broadcast messages.

Promiscuous Filter Mode

Promiscuous filtering mode is set using the PROM of the RXCTS register.

- PROM = 0 — Unicast, broadcast, and multicast filtering modes behave as described in the previous section
- PROM = 1 — Promiscuous mode is enabled and all frames are accepted, regardless of address

Setting the MC9S12NE64 to enable promiscuous mode is useful for diagnosing network issues, but it can overburden the MCU.

Ethertype Filtering

Ethertype filtering is provided as an additional means to reduce MC9S12NE64 CPU use. This feature uses the internal EMAC state machine to automatically filter-out and ignore incoming Ethernet packets before the packet is accepted by the EMAC buffer. It also indicates to the CPU that a packet is received without using any CPU cycles. Figure 13 is the Ethertype control (ETCTL) register, and Figure 14 is the Ethertype (ETYPE) register. If any ETCTL register bit is set, the Ethertype filter is enabled.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	FPET	0	0	FEMW	FIPV6	FARP	FIPV4	FIEEE
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 13. Ethertype Control (ETCTL) Register

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read:	ETYPE															
Write:																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 14. Ethertype (ETYPE) Register

Ethertype filtering is based on filtering an incoming Ethernet packet's length/type data field in the MAC header. Figure 15 shows the position of the 2-byte length/type data field in the standard Ethernet packet structure. Figure 15 shows that the length/type data is a part of the MAC header after the MAC hardware source address.

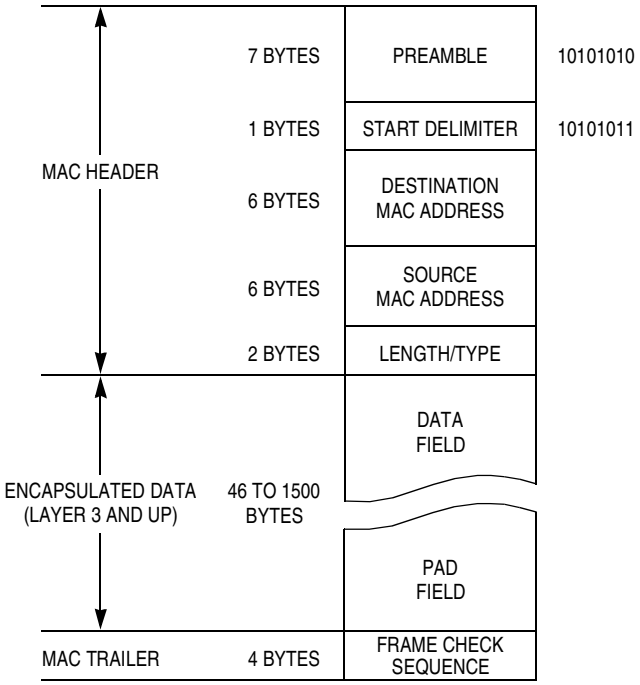


Figure 15. Format and Content of Ethernet Packets (Repeated)

Ethertype filtering allows only packets with an approved type/length field value to be accepted by the MC9S12NE64 filter. Multiple Ethertype filtering targets can be designated in ETCTL register. The ETCTL register provides five fixed Ethertype targets and one programmable Ethertype target. The available filtering targets include:

- FPET (programmable Ethertype) — If this filter target is used (FPET = 1), a 2-byte value for an Ethertype filter target must be provided using the ETYPE register
- FEMW (emWare® data type) — Ethertype value of 0x8876
- FIPV6 (Internet protocol version 6) — Ethertype value of 0x86DD
- FARP (address resolution protocol) — Ethertype value of 0x0806
- FIPV4 (Internet protocol version 4) — Ethertype value of 0x0800
- FIEEE (IEEE 802.3 length field) — Length data field that ranges from 0x0000 to 0x05DC

Configuring Ethernet Buffers and Maximum Frame Length Settings

The MC9S12NE64 has 8K of RAM that is available and shared between user RAM and the EMAC Ethernet buffer space. The EMAC Ethernet buffer space consists of three buffers, which contain:

- One transmit buffer — The user must store the following data in the transmission buffer to prepare a data packet for transmission:
  - Destination address
  - Source address
  - Type/length
  - Data

- Two receive buffers — Incoming Ethernet packets store the following data in a receive buffer if a buffer is available:
  - Destination address
  - Source address
  - Type/length
  - Data
  - Frame check sequence

Each EMAC Ethernet buffer is designed to hold only one Ethernet packet at a time. In addition, the EMAC Ethernet buffer space is user programmable. The following section provides a detailed review of the EMAC Ethernet buffers.

Buffer Size and Starting Address Mapping (BUFMAP)

The buffer size and starting address mapping data field (BUFMAP) in the Ethernet buffer configuration register (BUFCFG) allows the user to program the EMAC Ethernet buffer space and specify the ratio between user RAM and RAM used for the EMAC buffers.

BUFMAP specifies the EMAC Ethernet buffer size configuration options that determine the size of the MC9S12NE64 receive and transmit Ethernet buffers within system RAM.

The BUFCFG register is shown in Figure 16.

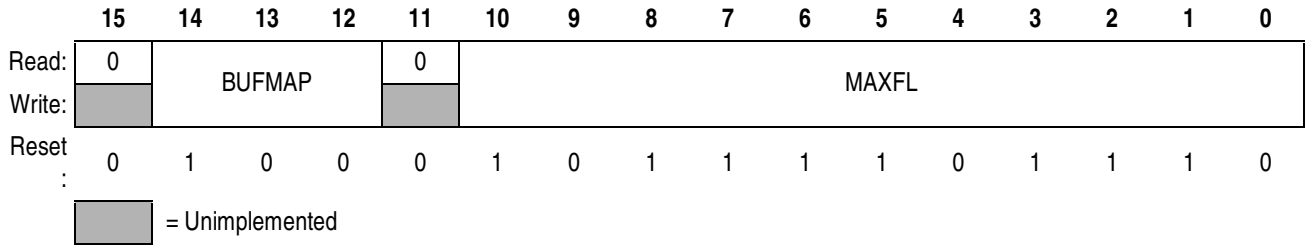


Figure 16. Ethernet Buffer Configuration (BUFCFG)

Table 2 provides the configurations for the MC9S12NE64 system RAM usage with the available BUFMAP settings. Table 2 shows that when EMAC Ethernet buffer space is maximized, user RAM is reduced.

Table 2. EMAC Ethernet Buffer Size and User RAM

BUFMAP	Individual Buffer Size (Bytes)	Total Size of EMAC Ethernet Buffer Space (Bytes)	Remainder RAM for User Application (Bytes)
0	128	384 = 0.375K	7.625K
1	256	768 = 0.75K	7.25K
2	512	1536 = 1.5K	6.5K
3	1K	3072 = 3K	5K
4	1.5K	4608 = 4.5K	3.5K

Because the maximum size of an Ethernet frame is approximately 1.5K, a setting of BUFMAP = 4 would allow each of the three MC9S12NE64 buffers to hold one Ethernet packet of the maximum allowable size as dictated by the IEEE 802.3 specification. Setting BUFMAP < 4 can:

- Maximize user RAM
- Filter Ethernet packets based on size

The setting of BUFMAP is a system/network design decision. If the devices on a network should accept only Ethernet packets of a certain size limit, BUFMAP can be configured to ignore packets that are too large. Setting BUFMAP to create a buffer based on size can reduce the burden on the CPU by ignoring packets that are too large. It is recommended that the system/network is designed to avoid use of large packets to maximize user RAM.

If a packet exceeds the receive buffer size, when BUFMAP < 4, the corresponding receive overrun error flag is set and the packet is filtered-out and ignored. No receive error flag or any other flag is set. No CPU bandwidth is used because the EMAC state machine does all packet filtering.

Receive Maximum Frame Length (MAXFL)

The receive maximum frame length (MAXFL) bits in the BUFCFG register allow the user to define a packet size limit. This setting compliments the BUFMAP bits configuration and can also be used to filter-out unwanted Ethernet packets based on their size. The MAXFL setting specifies the maximum receive frame length (in bytes). Received Ethernet packets that exceed MAXFL cause the babbling receive error interrupt flag (BREIF) to set. A CPU interrupt can be configured to occur if the babbling receive error interrupt enable bit (BREIE) is set.

Setting MAXFL prevents the buffer overrun flag from being set and does not automatically filter-out and ignore the packets. The packet must be manually removed by clearing the corresponding received valid flag and clearing the BREIF bit.

Flow Control

Flow control is an optional part of the IEEE 802.3 specification and is applicable only in full-duplex mode. Flow control allows a network device to pause network traffic by sending or receiving a pause Ethernet packet to relieve network traffic congestion.

Receiving Pause Frames

If the reception flow control enable (RFCE) bit is set in the RXCTS register, the receiver detects incoming pause frames. The detection of a pause packet is accomplished by one or both of the following:

- Multicast destination address of 01-80-C2-00-00-01 is detected
- Type/length field Ethertype value is 0x8808

A 16-bit value in the incoming pause packet specifies the duration of the pause event in units of 512 bit times (valid values are from 0x0000 to 0xFFFF). When RFCE is set and a pause frame is detected, the receive flow control interrupt flag (RFCIF) in the IEVENT register is asserted and the EMAC transmitter stops transmitting data frames for the received pause duration.

Hardware Generated Pause Control Frame Transmission

If no transmission is in progress and the EMAC is in full-duplex mode, a PAUSE command can be launched by writing a value of 0x02 to the 2-bit transmit command (TCMD) field in the transmit control and status (TXCTS) register. Figure 17 shows the TXCTS register.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	TXACT	0	CSFL	PTRC	SSB	0	0	0
Write:							TCMD	
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 17. Transmit Control and Status (TXCTS) Register

Before transmitting a pause packet, the user must configure the pause timer value and counter (PTIME) register. The PTIME register specifies the duration of the pause event in units of 512 bit times. To write to the PTIME register, the pause timer register control (PTRC) bit in the TXCTS register must be set, because when set, writes to the PTIME register update the duration of a pause control frame.

NOTE

The reception of a pause frame stops transmission using the START command, but it does not prevent transmission of pause control frames. In addition, pause frames may be accepted even if both receive buffers are full.

MII Management Interface

The MII management interface consists of a pair of signals that are used to send and receive information across the MII between the MAC and PHY to display PHY status information or configure different PHY options. These signals are also available in external PHY mode (see Table 1). To initialize the MII management interface, the management clock rate select (MDCSEL) bit field in the MII management command and status (MCMST) register must be configured. See the MCMST register in Figure 18.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	BUSY	NOPRE	MDCSEL			
Write:	OP							
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 18. MII Management Command and Status (MCMST) Register

The MDCSEL bit field sets the MDC frequency and must be equal-to or less-than 2.5 MHz, according to the IEEE 802.3 specification. Using the following equation, with a 25-MHz clock driving the MC9S12NE64, the MDCSEL bit field should be set to 0x05:

$$\text{MDC frequency} = \text{bus clock frequency} \div (2 \times \text{MDCSEL})$$

Other important data registers referenced for MII management read and write operations are included in the list below with a brief description:

- MII management PHY address (MPADR) — 5-bit PADDR field in MPADR specifies the PHY address. When using the MII management interface with the internal PHY, the PHY address setting in the EMAC and EPHY must match. With a 5-bit field, an MII management interface can target up to 32 attached PHY devices.
- MII management register address (MRADR) — 5-bit RADDR field in MRADR specifies 1 of the 32 internal PHY registers of a device. These internal PHY registers are 16 bits wide. A subset of 32 registers is defined by the IEEE 802.3 standard. These 32 PHY registers can be accessed only through the MII management interface and are not visible through the MC9S12NE64 register map.
- MII management write data (MWDATA) — The 16-bit WDATA data field in MWDATA is used during an MII management write operation. Before initiating an MII write operation, the value to be written must be stored in the WDATA data field.
- MII management read data (MRDATA) — 16-bit RDATA data field in MRDATA contains the MII management read operation result. RDATA is valid only when the MII management transfer complete interrupt flag (MMCIF) in the interrupt event (IEVENT) register is set after a valid read frame operation.

Read Operation

Before performing a read operation by MII management, values for the PADDR and RADDR fields must be configured by the user indicating which PHY device is to be addressed and which 16-bit register is to be read, respectively. Setting the OP field in the MCMST register to 0x02 while the BUSY bit is clear initiates the MII management read and sets the BUSY bit. As soon as the read MII management frame operation is complete, the BUSY bit clears, the MRDATA register is updated with the MII read result, and the MMCIF bit is set.

Write Operation

To perform a write operation by MII management interface, the user must provide a value for the PADDR and RADDR fields, indicating which PHY device is to be addressed and which 16-bit register is to be read, respectively. The user must also provide the value to be written and store that value in the WDATA data field. Setting the OP field in the MCMST register to 0x01 while the BUSY bit is clear initiates the MII management write and sets the BUSY bit. As soon as the write MII management frame operation is complete, the BUSY bit clears and the MMCIF bit is set.

MC9S12NE64 Ethernet Physical Transceiver (EPHY)

The EPHY is an implementation of an Ethernet physical transceiver (PHY) as defined by IEEE 802.3 standard. For basic operation, the EPHY must be supplied a 25-MHz clock input specified at 25 ppm.

The EPHY is compliant with IEEE 802.3 specifications for 10BASE-T (clause 14) and 100BASE-TX (clauses 24 and 25).

- 10BASE-T specification — PHY operation at 10 Mbps, called Ethernet, over un-shielded twisted pair (UTP) copper cable.
- 100BASE-TX specification — PHY operation at 100 Mbps, called Fast Ethernet, also over UTP copper cable.

The EPHY is also compliant with Ethernet operation using category 5 UTP copper cable at cable lengths of 100 meters.

Table 3. Summary of EPHY Compatibility

Technology	Maximum Segment Length (Meters)	Encoding Method	Topology Media		Bit Rate (Mbps)
10BASE-T	100	Manchester	Star	2 Pair UTP Cat. 3, 4, 5	10
100BASE-TX	100	4B/5B with MLT-3	Star	2 Pair UTP Cat. 5	100

The EPHY (like the EMAC) supports the MII and the MII management interface (see IEEE 802.3 clause 22). The EMAC and EPHY use the MII to exchange data, set up the EPHY, and communicate status. The EPHY also includes the following features:

- Supports auto-negotiation
- Auto-negotiation next page ability
- Full-duplex and half-duplex support
- Digital adaptive equalization
- Baseline wander (BLW) correction
- Loopback modes

The EPHY provides digital/analog encoding and decoding, which is required for the MC9S12NE64 to communicate on the UTP cable. A block diagram of this peripheral is provided in the Figure 19. Figure 19 shows that the clock input of the EPHY is the 25-MHz reference clock (REF CLOCK).

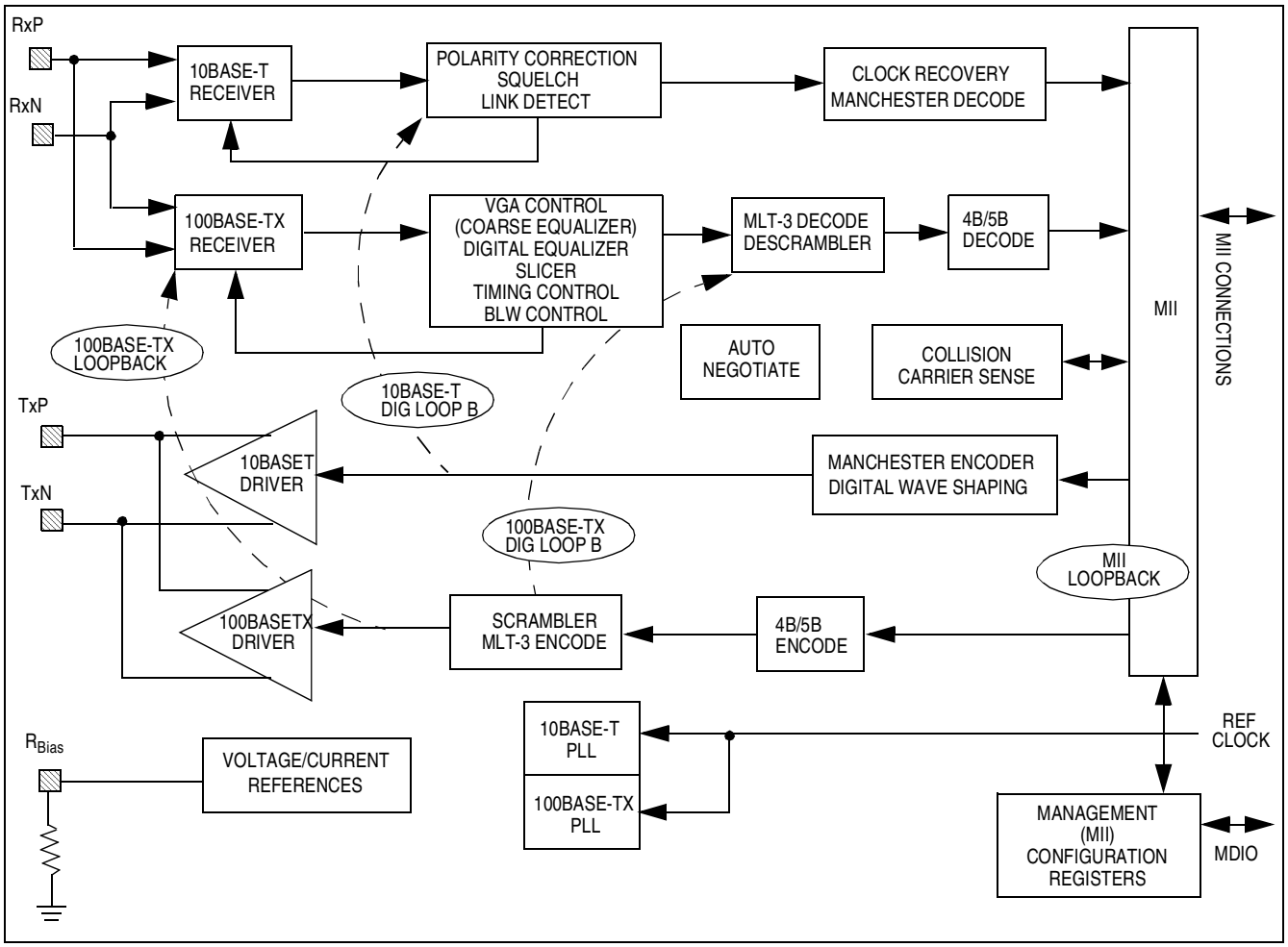


Figure 19. EPHY Block Diagram

EPHY Registers and Configuration Options

The integrated EPHY is designed to provide control and status access by the EMAC (through the MII management interface). Therefore, the EPHY has only a few directly accessible registers. The EPHY registers available from the EPHY register map include:

- Ethernet physical transceiver control register 0 (EPHYCTL0)
- Ethernet physical transceiver control register 1 (EPHYCTL1)
- Ethernet physical transceiver status register (EPHYSR)

EPHYCTL0 and EPHYCTL1 are used mainly to enable the EPHY (by setting the EPHYCTL0 EPHYEN bit). Figure 20 shows the EPHYCTL0 and EPHYCTL1 registers.

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
Ethernet Physical Transceiver Control Register 0 (EPHYCTL0)	Read:	EPHYEN	ANDIS	DIS100	DIS10	LEDEN	EPHYWAI	0	EPHYIEN
	Write:								
	Reset:	0	1	1	1	0	0	0	0
Ethernet Physical Transceiver Control Register 1 (EPHYCTL1)	Read:	0	0	0	PHYADD4	PHYADD3	PHYADD2	PHYADD1	PHYADD0
	Write:								
	Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 20. Ethernet Physical Transceiver Control Register 0 and Register 1

Before enabling the EPHY, it is important to configure several EPHY options. Some basic settings will be made using the EPHYCTL0 and EPHYCTL1 registers, but the majority of the EPHY settings will be configured through the MII management interface through the EMAC (see the Internal EPHY Control and Status Registers section). The EPHY options that can be set through the EPHY register map are:

- EPHY address configuration (see EPHYADD[4:0] in EPHYCTL1)
- EPHY clock generation disable (see DIS100 and DIS10 in EPHYCTL0)
- EPHY LED enable (see LEDEN in EPHYCTL0)
- EPHY enable (see EPHYEN in EPHYCTL0)
- Auto-negotiation disable setting (see ANDIS in EPHYCTL0)
- EPHY interrupt enable (see EPHYIEN in EPHYCTL0)

Configuring the EPHY address, EPHY clock generation disable, and LED enable is straightforward. The EPHY address must be recorded by the user so that when an MII management operation is executed, the correct PHY address (see the MPADDR register description in the MII Management Interface section) is provided. The EPHY enable, EPHY clock generation disable, and LED enable bits are typically set or cleared as part of the EPHY initialization sequence.

Auto-Negotiation Disable

Auto-negotiation is a mechanism that allows two network devices to select the best common speed and duplex mode automatically during link initiation. The EPHY can be configured to advertise specific speed, duplex mode, and pause operation abilities by configuring the EPHY auto-negotiate advertisement register, which is accessible only through the MII management interface (see the Auto-Negotiate Advertisement Register section). Auto-negotiation is defined in the IEEE 802.3 standard in clause 28.

The MC9S12NE64 can be operated with or without enabling auto-negotiation. In some instances, the user may need to specify speed, duplex mode, and flow control settings for a particular application. Disabling auto-negotiation may be required to establish a link due to interoperability issues.

If the ANDIS bit in the EPHYCTL0 register has a reset value of 1 (and EPHYEN is set, which indicates that the EPHY is enabled), auto-negotiation is disabled. The ANDIS bit is internally latched to the ANE bit of the EPHY PHY control register. Typically, the PHY control register is accessible only by the EMAC through the MII management interface. The PHY control register is part of the internal EPHY register that

is not accessible from the MC9S12NE64 register map. With auto-negotiation disabled, the speed and duplex mode for the EPHY must be manually set using the MII management interface and writing to the DPLX and DATARATE bits of the PHY control register (see Figure 22).

If auto-negotiation is used, the ANDIS bit in the EPHYCTL0 register should be cleared (alternatively, the ANE bit of the EPHY PHY control register can be set through the MII). When the EPHY and EPHY clock are enabled, the MC9S12NE64 will use auto-negotiation to determine speed, duplex mode, and flow control settings. When auto-negotiation is complete, the device must ensure that the EMAC is also configured to operate with the settings that the auto-negotiation process resolved. To determine the resolved capabilities, the EPHY proprietary status register must be read using an EMAC MII read operation and then decoded (see Figure 26).

EPHY Interrupt Enable

Before enabling the EPHY interrupts (by setting the EPHYIEN bit in the EPHYCTL0 register), the EPHY PHY interrupt control register (see Figure 27) must be configured. The PHY interrupt control register is not visible from the EMAC MII management interface (see the Interrupt Control Register section).

Clearing the EPHY Interrupt

Figure 21 shows the Ethernet physical transceiver status register (EPHYSR), which is used to clear an EPHY interrupt. Clearing an EPHY interrupt is a two-step process:

- Read the EPHY internal interrupt register (register 0x10) through the MII management interface (MDIO)
- Write the EPHYIP bit to 1

Reading an internal EPHY register through the MII management interface is performed by an MII read operation by the EMAC. For an EMAC MII read operation, the user must provide a PHY address (PADDR) and PHY register address (RADDR). After the MII read operation, the result of the MII read is stored in the MRDATA register (see the MII Management Interface section).

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	100DIS	10DIS	0	0	0	EPHYIF
Write:								
Reset:	0	0	1	1	0	0	0	0

= Unimplemented

Figure 21. Ethernet Physical Transceiver Status Register (EPHYSR)

Internal EPHY Control and Status Registers

These registers are not directly accessible from the MC9S12NE64 register map. Using the EMAC MII read and write operation will provide access to these internal EPHY registers. IEEE 802.3 specifies the register set, which consists of 32 individual 16-bit PHY registers. The IEEE 802.3 specification refers to these registers as the MII management register set (see IEEE 802.3, clause 22). IEEE 802.3 specifies the contents of registers 0 through 15; registers 16 through 31 can be defined by the developer.

PHY Control Register

The PHY control register has several important control registers for EPHY basic operation. Some basic control settings are discussed in this section. Figure 22 shows the PHY control register.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read:	RESET	LOOP BACK	DATA RATE	ANE	PDWN	ISOL	RAN	DPLX	COL TEST	0	0	0	0	0	0	0
Write:																
Reset:	0	0	1	X	0	0	0	1	0	0	0	0	0	0	0	0

Figure 22. PHY Control Register (MII Management Register 0)

- DATA_RATE — Allows the user to select 10 Mbps or 100 Mbps speed if auto-negotiation is disabled. If DATA_RATE = 0, then 10 Mbps is selected. If DATA_RATE = 1, then 100 Mbps is selected.
- ANE — Setting this bit enables auto-negotiation. This bit can be latched on start-up based on the setting of the ANDIS bit in the EPHYCTL0 register (see the EPHY Registers and Configuration Options section).
- RAN — Setting RAN can restart auto-negotiation. Restarting auto-negotiation is required when the link to the network is lost. (For example, unplugging the PHY from the network will cause link loss.) The status of the link can be determined by EPHY interrupts (see the Interrupt Control Register section).
- DPLX — The duplex mode is manually configured with the DPLX bit when auto-negotiation is disabled. Duplex mode options are full-duplex and half-duplex. Setting the DPLX bit configures the link for full-duplex mode.

PHY Status Register and Proprietary Status Register

There are two status registers available to the user, the PHY status register and the proprietary status register. Most status indication, for basic operation, that will be discussed is contained in the proprietary status register. Typically, the MC9S12NE64 reads this status register when an EPHY interrupt occurs. The two most important interrupts are those which occur when:

- Auto-negotiation is complete
- Status of the link has changed

This section describes several important bits. See the MC9S12NE64 data sheet for a complete description of all bits.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read:	100-T4	100X-FD	100X-HD	10T FD	1T HDO	0	0	0	0	SUP PRE	AN COMP	REM FLT	AN ABL	LNK STST	JAB DT	EX CAP
Write:																
Reset:	0	0	1	X	0	0	0	1	0	0	0	0	0	0	0	0

Figure 23. PHY Status Register (MII Management Register 1)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read:	0	LNK	PDMD	SPD	0	ANNC	PRCVD	ANC MODE	0	0	PLR	0	0	0	0	0
Write:																
Reset:	0	0	1	X	0	0	0	1	0	0	0	0	0	0	0	0

Figure 24. Proprietary Status Register (MII Management Register 17)

- ANCOMP and ANNC — Both indicate whether auto-negotiation is complete. ANNC is a duplicate of ANCOMP. When auto-negotiation is enabled, the contents of MII management registers 4 through 7 are valid. When auto-negotiation is complete, the EMAC must be adjusted to ensure that duplex and flow control settings are correct (by referencing the DPMD and SPD bits). The user is notified of a change in auto-negotiation by an EPHY interrupt.
- LNKSTST and LNK — If LNKSTST = 1 (a link is established), LNK must be clear (LNK = 0). An EPHY interrupt indicates a change in the network link.
- PDMD — Indicates the current duplex mode of the PHY. When set, the PHY is operating in full-duplex mode. When clear, operation is half-duplex.
- SPD — Indicates the current speed mode of the PHY. When set, the PHY is operating in 100 Mbps mode. When clear, operation is 10 Mbps mode.
- PRCVD and ANC_MODE: Indicate status of auto-negotiation process. When PRCVD = 1, the auto-negotiation base page has been received. If ANC_MODE = 1, auto-negotiation not resolved to a common operating mode. See the Auto-Negotiate Advertisement Register section.

Auto-Negotiate Advertisement Register

The configuration of this register determines which functional capabilities the EPHY advertises during the auto-negotiation process. The advertisement options that can be set by the auto-negotiate advertisement register are:

- NXTP — Next page capability
- FLC_TL — Flow control capability
- TAF_100FD — 100BASE-TX full-duplex capability
- TAF_100HD — 100BASE-TX half-duplex capability
- TAF_10FD — 10BASE-T full-duplex capability
- TAF_10HD — 10BASE-T half-duplex capability

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read:	NXTP	0	RFLT	0	0	FLCTL	0	TAF 100FD	TAF 100HD	TAF 10FD	TAF 10HD	SELECTOR FIELD [4:0]				
Write:																
Reset:	1	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0

Figure 25. Auto-Negotiate Advertisement Register (MII Management Register 4)

Figure 25 shows the auto-negotiate advertisement register. Setting the bits for a particular Ethernet option ensures that a capability is advertised during auto-negotiation. For example, if the user does not want to allow a network device to ever negotiate to full-duplex operation at 100 Mbps using flow control, the user could ensure that both the FLCTL and TAF_100FD bits are clear before enabling the EPHY clock generation (if auto-negotiation is set).

Interrupt Control Register

The interrupt control register indicates which events would trigger an EPHY interrupt. An EPHY interrupt is indicated by the EPHYIF bit in the EPHYSR register; a CPU interrupt is triggered only if the EPHYIEN bit in the EPHYCTL0 bit is set. Also, when an EPHY interrupt occurs, a two-step interrupt-clearing mechanism is required (as discussed in the Clearing the EPHY Interrupt section). A brief discussion of select PHY interrupt events is provided in this section. The interrupt control register is shown in Figure 26.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read:	0	ACKIE	PRIE	LCIE	ANIE	PDFIE	RFIE	JABIE	0	ACKR	PGR	LNK CNG	AN CNG	PDF	RMTF	JABI
Write:																
Reset:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 26. Interrupt Control Register (MII Management Register 16)

See the MC9S12NE64 data sheet for a complete description of all bits. Some interrupt control register bits are described below:

- ACKIE (acknowledge bit received interrupt enable) and PRIE (page received interrupt enable) — Indicate that the auto-negotiation process is exchanging base pages.
- LCIE (link changed enable) — Occurs when the link changes meaning (either a new link has been established, or an established link has been lost). To determine what has occurred after this interrupt, the PHY status register and proprietary status register must be read and decoded according to the PHY Status Register and Proprietary Status Register section.
- ANIE (auto-negotiation changed enable) — Occurs when the state of the auto-negotiation state machine has changed since the last access of this register. A change to the auto-negotiation state machine may require re-verification of the resolved auto-negotiation settings (see the PHY Status Register and Proprietary Status Register section).
- PDFIE — Occurs when both auto-negotiation and parallel detection mechanisms for determining a link partner's ability fail.

Reading the interrupt control register (through the EMAC MII management interface) is an important part of the clearing mechanism for the EPHYIF bit, but it is more important to decode which PHY interrupt occurred and correctly handle these interrupt events.

Initializing the MC9S12NE64 Ethernet Controller

The EMAC and EPHY are designed as two separate modules on the MC9S12NE64, but if using the internal EPHY, they should be initialized together. The following procedure will prepare the MC9S12NE64 integrated Ethernet controller for general Ethernet operation.

1. Initialize the CRG to generate a 25-MHz internal bus clock. This is required for 100-Mbps operation. A 2.5-MHz bus clock would be acceptable for 10-Mbps operation, but remember that the MC9S12NE64 requires a 25-MHz clock input.
2. Disable the EPHY clock by setting the DIS10 and DIS100 bits in the EPHYCTL0 register to 1. The EPHY clocks will not be enabled until both EMAC and EPHY are completely configured.
3. Configure the PHY address using the EPHYCTL1 register bits EPHYADD[4:0]. EPHYADD[4:0] will latch EPHY register 14 when the EPHYEN bit is set.
4. Configure auto-negotiation:
   - If auto-negotiation is used, clear the ANDIS bit in the EPHYCTL0 register.
   - If auto-negotiation is not used, set the ANDIS bit in the EPHYCTL0 register.
5. Enable EPHY LEDs by setting the LEDEN bit in the EPHYCTL0 register.
6. Enable EPHY interrupts by setting the LEDEN bit in the EPHYCTL0 register.
7. Enable EPHY by setting the EPHYEN bit in the EPHYCTL0 register. If the EPHY is enabled, MII operation between the EMAC and EPHY is possible.
8. Configure the EMAC MDC clock using the MDCSEL bits in the MCMST register.
9. Configure the EMAC Ethernet buffer space in memory using the BUFMAP bit field in BUFCFG register.
10. Configure the EMAC maximum reception frame length using the MAXFL bit field in the BUFCFG register.
11. Configure the MAC hardware 6-byte address using the MACAD registers.
12. Configure the EMAC Ethertype filter mode using the ETYPE and ETCTL registers.
13. Configure the EMAC MAC hardware address filter mode. Options include configuring the BCREJ, CONMC, and PROM bits of the RXCTS register. The receive flow control configuration (RFCE bit) must be configured only if auto-negotiation is disabled. If auto-negotiation is used, this setting must be configured after auto-negotiation is complete and the pause setting is resolved.
14. Configure EMAC loopback (MLB bit) and external PHY mode (EXTPHY bit) if required. If auto-negotiation is disabled, the EMAC duplex mode (FDX bit) can also be configured. If auto-negotiation is enabled, the EMAC duplex mode should be configured after auto-negotiation is complete.
15. Enable the EMAC by setting the EMACE bit in the NETCTL register.
16. Configure and enable EMAC interrupts as needed using the EMAC interrupt mask (IMASK) register.
17. Initialize and transmit pause time duration:
    a. Set the PTRC bit in the TXSCTS register.
    b. Configure the PTIME register.
18. Enable system interrupts.

19. Configure EPHY through the EMAC MII management interface — Configure speed, duplex mode, and flow control EPHY auto-negotiation advertisement by writing to the EPHY auto-negotiate advertisement register.
20. Configure EPHY through the EMAC MII management interface:
    a. Configure the EPHY interrupts by writing to the EPHY interrupt control register.
    b. Read the EPHY interrupt control register to verify content of the EPHY interrupt control register.
21. Start the EPHY clock generators.
    – If auto-negotiation is used, start the EPHY clock generators by clearing the DIS100 and DIS10 bit of the EPHYCTL0 register.
    – If auto-negotiation is not used:
    a. Configure the EPHY through the EMAC MII management interface and configure speed and duplex mode by writing to the DPLX and DATARATE bits in the EPHY control register.
    b. Start the EPHY clock generators by clearing the DIS100 and DIS10 bit of the EPHYCTL0 register.
22. If auto-negotiation is used, as soon as both auto-negotiation is complete and a link is established, the negotiated pause and duplex settings can be determined from the EPHY MII registers. The EMAC then must be updated by configuring the RFCE and FDX bits to match the negotiated pause and duplex settings.

After initialization, the MC9S12NE64 can transmit and receive Ethernet packets on a network. A brief overview of transmit and receive operations using the MC9S12NE64 is provided in the following sections.

Using the MC9S12NE64 Ethernet Interface

After configured and initialized, the MC9S12NE64 Ethernet interface is easy to use. An overview of sending and receiving Ethernet packets is described in this section.

Buffer Transmit Operation

To start an Ethernet transmission, the user data, destination address, source address, and length/type field data must be written to the transmit buffer.

The packet can be transmitted using a START command. The START command is launched by writing a value of 0x01 to the 2-bit transmit command (TCMD) field in the transmit control and status (TXCTS) register when the transmitter active status (TXACT) bit is clear. The transmitter automatically appends the frame check sequence.

Buffer Receive Operation

Valid data is received when the receive buffer (A or B) complete flag is set. The received data is stored in the receive buffer and is available for user access.

To clear a buffer that has been processed, the receive buffer complete interrupt flag for the corresponding buffer must be cleared.

If the two receive buffers are full, other incoming Ethernet packets are dropped. The two receive buffers are full when RXACIF and RXBCIF in the IEVENT register are set to 1.

Network Data Transaction at Upper Layer Protocols

Reception and transmission of Ethernet packets as described in the Using the MC9S12NE64 Ethernet Interface section is not the level that typical user Ethernet applications use. Typically, a software stack such as TCP/IP is used to initiate and maintain communications within the network and assist the user network application. Figure 3 shows ISO's OSI communication model. Figure 27 provides an overview of how the user data from the user application is encapsulated and eventually is transmitted.

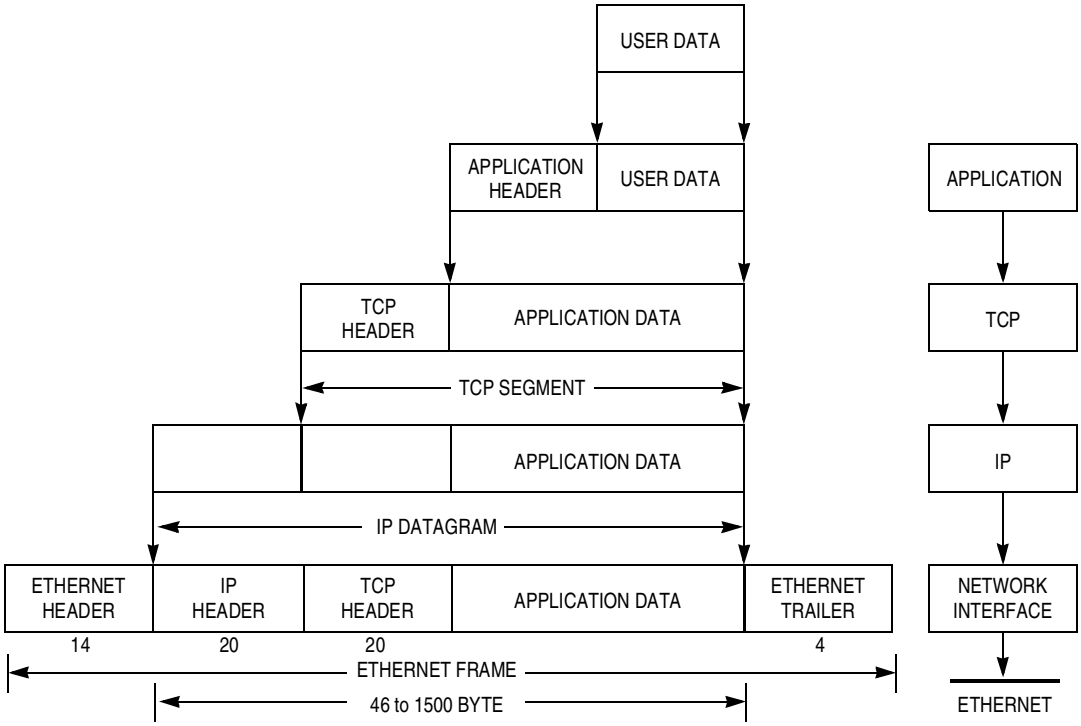


Figure 27. User Data Encapsulated to and from the Upper Layers TCP/IP Model

When a packet is received, the protocol layers are decoded and removed. This process, called de-encapsulation, correctly processes the packet with the appropriate upper layer protocols to access the user data.

Table 4 describes some of the upper layer protocols shown in Figure 3 and Figure 27. These protocols are components of a TCP/IP stack. Transmission and reception of user data at the upper protocol layers is managed by TCP/IP software. A TCP/IP stack defines a set of protocols that allows network devices to connect to a specific device and exchange data on a network. These protocols, defined by RFC (request for comments), enable an embedded device to send email, serve web pages, transfer files, and provide other basic connectivity functions.

Table 4. Some Upper Layer Protocols in the TCP/IP Stack Model

Acronym/ Term	Description	Definition
DHCP	Dynamic host configuration protocol	Allocates IP addresses dynamically
DNS	Domain name server	Program/computer that converts a domain name into its IP address
FTP	File transfer protocol	Used to transfer files across a network
HTTP	Hyper text transfer protocol	Used to transmit web pages
ICMP	Internet control message protocol	Used to report errors from IP level and above
IP	Internet protocol	Mechanism for delivering packets across a network
SMTP	Simple mail transfer protocol	Used for sending and receiving email
SNMP	Simple network management protocol	Used by computers that monitor and manage network activity to communicate with one another and the computers they are monitoring
TCP	Transmission control protocol	Guarantees delivery of data
TFTP	Trivial file transfer protocol	Subset of FTP that does not require valid username and password
UDP	User datagram protocol	Found at the network layer along with the TCP protocol. UDP does not guarantee reliable, sequenced packet delivery. If data does not reach its destination, UDP does not retransmit, but TCP does.

Conclusion

The MC9S12NE64 is a tightly-integrated, easy-to-use, single-chip solution for embedded Ethernet applications. Because the MC9S12NE64 integrates Ethernet functionally in a single package, it provides a low-cost solution. The Ethernet functionality of the MC9S12NE64 complies to the IEEE 802.3 specification to provide improved network functionality and interoperability. Configuration and initialization of this tightly-integrated solution is also enhanced by C-based development tools. Combining the MC9S12NE64 with a TCP/IP stack provides an interface for developing Ethernet-enabled embedded devices with a rich set of network functionality.

NOTE

With the exception of mask set errata documents, if any other Freescale Semiconductor document contains information that conflicts with the information in the device user guide, the user guide should be considered to have the most current and correct data.

Although specific methods and tools were used to develop and debug this demo, Freescale Semiconductor does not recommend or endorse any particular methodology, tool, or vendor. These methods and tools are provided only to describe the generic principles and features that may be required for development of a networked device.

This page is intentionally blank

This page is intentionally blank

How to Reach Us:

USA/Europe/Locations not listed:
Freescale Semiconductor Literature Distribution
P.O. Box 5405, Denver, Colorado 80217
1-800-521-6274 or 480-768-2130

Japan:
Freescale Semiconductor Japan Ltd.
SPS, Technical Information Center
3-20-1, Minami-Azabu
Minato-ku
Tokyo 106-8573, Japan
81-3-3440-3569

Asia/Pacific:
Freescale Semiconductor H.K. Ltd.
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T. Hong Kong
852-26668334

Learn More:
For more information about Freescale Semiconductor products, please visit
http://www.freescale.com

AN2692
Rev. 0.2, 9/2004

freescale™
semiconductor