# DM9000

*32/ 16/ 8-Bit Three-In-One Fast Ethernet Controller*

# Application Notes V1.22

**Technical Reference Manual**

**Davicom Semiconductor, Inc**

# 1 Introduction

## 1.1 General Description

The DM9000 is a fully integrated and cost-effective Fast Ethernet MAC controller with a general processor interface, an EEPROM interface, a 10/100M PHY and 4K-dword SRAM (3K-byte for TX FIFO and 13K-byte for RX FIFO). It is designed with low power and high performance process that supports 3.3V with 5V tolerant I/O. Besides, the DM9000 supports 8/ 16/ 32-bit processor interface to internal memory accesses for many different processors.

The goal of this document is for the embedded design engineers, to implement the DM9000 LAN chip on any processor's architecture quickly and successfully, with providing the exact reference information and pertaining to many embedded systems.

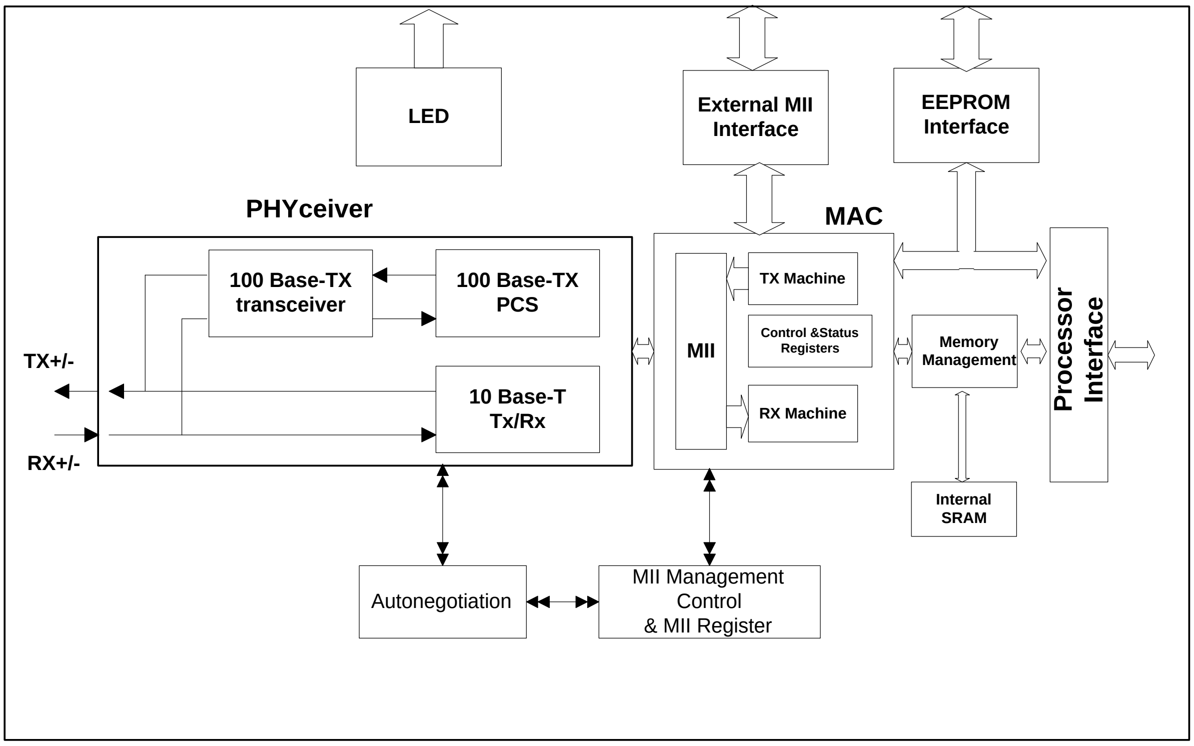


Figure 1.1 DM9000 Internal Block Diagram

## 2 General Processor Bus Description

This section is intended to aid design engineers connecting the DM9000 device to a micro-processor or micro-controller. The discussion will include the pin functional table, and the individual control signals of the DM9000 involved in the connection between the device and an associated micro-processor/ micro-controller in detail.

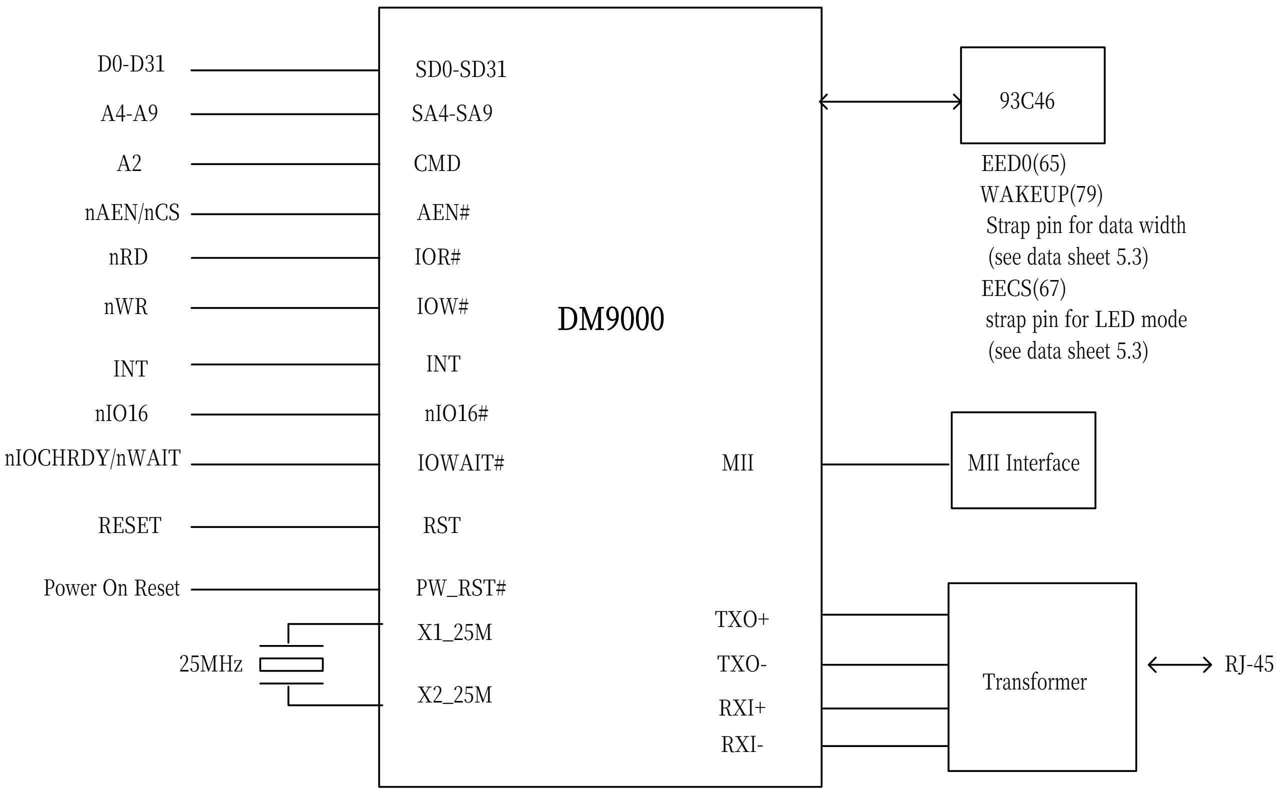


**Figure 2.1 Signal Connection with a Processor Interfacing**

### 2.1 ISA Bus

The DM9000 supports an asynchronous bus interface. The industry standard ISA bus is one of the typical asynchronous buses.

#### 2.1.1 Pin Function Table

**Note: The pins of ISA interface except SD8, SD9 and IO16 all have a pulled down**

**resistor about 60k Ohm internally.**

| ISA Bus Signal | DM9000 Signal | Pin No. | I/O | Description |
|---|---|---|---|---|
| IORC# | IOR# | 1 | I | ISA READ Command<br>The default is low active. The polarity can be changed via EEPROM setting. |
| IOWC# | IOW# | 2 | I | ISA WRITE Command<br>The default is low active. The polarity can be modified via EEPROM setting. |
| AEN | AEN# | 3 | I | Address Enable<br>A low active signal is used to select the DM9000. |
| CHRDY | IOWAIT | 4 | O | ISA Command Ready<br>This ISA signal is driven low to insert the wait cycles to current host read/ write command.<br>The polarity can be changed via EEPROM setting. |
| RESET | RST | 14 | I | Hardware RESET Command<br>When this pin is asserted high, The DM9000 performs an internal system reset. |
| SD0 ~ 15 | SD0 ~ 15 | 6,7,8,9, 10,11,12, 13,89, 88,87,86, 85,84, 83,82 | I/O | ISA Data Bus 0 ~ 15 |
| SA4 ~ 9 | SA4 ~ 9 | 93,94,95, 96,97,98 | I | ISA Address Bus 4 ~ 9<br>These pins are used to select the DM9000 I/O base address 300H ~ 370H. |
| SA2 | CMD | 92 | I | COMMAND Type<br>When an input signal is low, the access of the command cycle is ADDRESS port:<br>The DM9000 device address port = 300H ~ 370H + 0<br>When an input signal is high, the access of the command cycle is DATA port:<br>The DM9000 device data port = 300H ~ 370H + 4 |
| IO16# | IO16 | 91 | O | ISA WORD Command Indication<br>This pin is used to indicate the access of internal memory is 16-bit word mode. The default is low active and open-collected which can be programmed by EEPROM setting. |
| IRQn | INT | 100 | O | ISA Interrupt Request<br>The default is high active and force output. Its polarity can be modified by setting the EEPROM or the strap pin MDC. |

**Table 2.1 Pin Function Table for ISA Bus**

### 2.1.2 I/O Base Address Decoding Example

The I/O base address for ISA bus can be programmed from 300H to 370H. The default value of I/O base address is 300H.

| A9 | A8 | A7 | A6 | A5 | A4 | I/O based address |
|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 | 300H |
| 1 | 1 | 0 | 0 | 0 | 1 | 310H |

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 1 | 0 | 320H |
| 1 | 1 | 0 | 0 | 1 | 1 | 330H |
| 1 | 1 | 0 | 1 | 0 | 0 | 340H |
| 1 | 1 | 0 | 1 | 0 | 1 | 350H |
| 1 | 1 | 0 | 1 | 1 | 0 | 360H |
| 1 | 1 | 0 | 1 | 1 | 1 | 370H |

**Table 2.2 I/O Based Address for ISA Bus**

The chart below shows the decoding of I/O base address 300H:

| A15 | A14 | A13 | A12 | A11 | A10 | A9 | A8 | A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 2.1.3 ISA Bus 8/ 16-Bit Mode and MII Interface Setting

The DM9000 provides two I/O modes: 8/ 16-bit data width of memory commands for ISA bus to read/ write the data from/ to hardware DATA port. The decoder table is shown as follows,

| WAKEUP | EEDO | Data width |
|---|---|---|
| 0 | 0 | 16-bit |
| 0 | 1 | 32-bit (Not supported for ISA interface) |
| 1 | 0 | 8-bit |
| 1 | 1 | Reserved |

The EEDO pin is also used as a strap pin. It combines with another strap pin WAKEUP, and these two pins can be used to set the 8/ 16/ 32-bit data width of the internal memory access for RX/ TX packets. The logic "1" means the strap pin is pulled high.

In ISA bus interface, the 32-bit DATA I/O mode is not supported, because ISA bus supports only 8-bit or 16-bit DATA I/O mode. And the system designer can use the SD [16:31] pins as the MII interface function.

The interrupt status register ISR (REG_FE) can help us to check the I/O mode setting:

| ISR Bit [7: | 6] | IOMODE (data width) |
|---|---|---|
| 0 | 0 | 16-bit mode |
| 0 | 1 | 32-bit mode (Not supported for ISA interface) |
| 1 | 0 | 8-bit mode |
| 1 | 1 | Reserved |

### 2.1.4 Command Type

The CMD pin is an input pin and used to set the COMMAND type. When the CMD pin detected the input signal is low, the access of this command cycle is for ADDRESS port. When input is high, the access of this command cycle is for DATA port. The following diagram (Figure 2.2) is an example for the DM9000 CMD pin connecting to ISA bus.

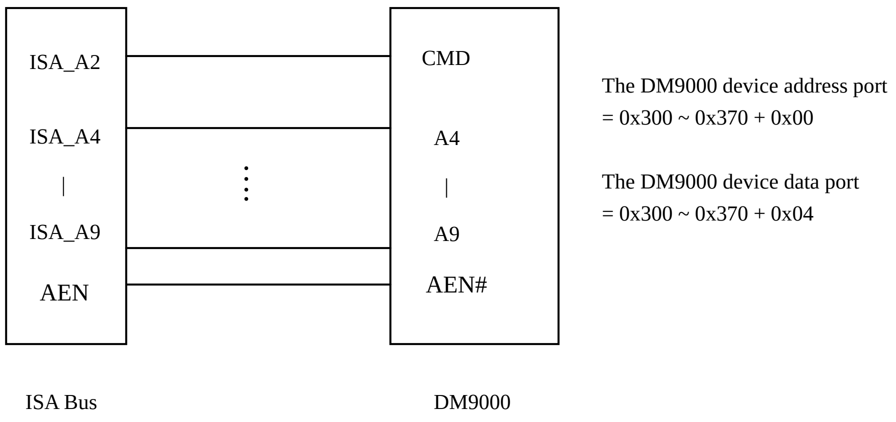

ISA_A2
ISA_A4
|
ISA_A9
AEN
CMD
A4
|
A9
AEN#
The DM9000 device address port = 0x300 ~ 0x370 + 0x00
The DM9000 device data port = 0x300 ~ 0x370 + 0x04
ISA Bus
DM9000

**Figure 2.2 DM9000 CMD Pin and ISA Bus.**

## 2.2 Typical Signal Connection with Processor Bus

The DM9000 can support many general processors parallel bus interfaces such as 8051, x86, ARM, MIPS, Hitachi H8, OAK, Intel XScale, and Motorola 68k/ iMx… It is designed to facilitate the implementation of Fast Ethernet connectivity solutions for many embedded systems.

### 2.2.1 Pin Function Table

**Note: The pins of processor parallel interface except SD8, SD9 and IO16 all have a pulled down resistor about 60k Ohm internally.**

| Processor Bus Signal | DM9000 Signal | Pin No. | I/O | Description |
|---|---|---|---|---|
| nRD | IOR# | 1 | I | Processor READ Command<br>This pin is low active at default; its polarity can be modified by EEPROM setting. |
| nWR | IOW# | 2 | I | Processor WRITE Command<br>This pin is low active at default; its polarity can be changed by EEPROM setting. |
| nAEN / nCS | AEN# | 3 | I | Address Enable<br>A low active signal is used to select the DM9000. |
| nIOCHRDY / | IOWAIT | 4 | O | Command Ready |

| | | | | |
|---|---|---|---|---|
| nWAIT | | | | When a command is issued before last command is completed, the IOWAIT will be pulled low to indicate the current command is waited This pin is low active at default; its polarity can be modified by EEPROM setting.<br>This pin is open-collected as default, it can be programming by EEPROM setting. |
| RESET | RST | 14 | I | When this pin is asserted high, The DM9000 performs an internal system RESET. |
| SD0 ~ 15 | SD0 ~ 15 | 6,7,8,9,10,11,12,13,89,88,87,86,85,84,83,82 | I/O | Data Bus 0 ~ 15 |
| SA4 ~ 9 | SA4 ~ 9 | 93,94,95,96,97,98 | I | Address Bus 4 ~ 9<br>These pins are used to select the DM9000 IO base address 300H ~ 370H.<br>The DM9000 device address = nCS + 300H ~ 370H<br>These pins can be also set by hardware directly. If the DM9000 IO base address is 300H, then SA9,SA8 are connect to VCC and SA7, SA6, SA5, SA4 are connect to GND.<br>The DM9000 device address = nCS + 300H |
| SA2 | CMD | 92 | I | Command Type<br>When low, the access of this command cycle is ADDRESS port<br>The DM9000 device address port = nCS + 300H ~ 370H + 0<br>When high, the access of this command cycle is DATA port<br>The DM9000 device data port = nCS + 300H ~ 370H + 4 |
| nIO16 | IO16 | 91 | O | WORD Command Indication<br>When the access of internal memory is word or dword width, this pin will be asserted.<br>This pin is low active at default; its polarity can be modified by EEPROM setting.<br>This pin is open-collected as default, it can be programming by EEPROM setting. |
| INT | INT | 100 | O | Interrupt Request<br>This pin is high active at default, its polarity can be modified by EEPROM setting or strap pin MDC.<br>This pin is open-collected as default, it can be programming by EEPROM setting. |
| SD16 ~ SD31 | SD16 ~ 31 (in double word mode) | 56,53,52 51,50,49 47,46,45 44,43,41 40,39,38 37 | I/O | Processor Data Bus 16 ~ 31<br>These pins are used as the data bus bits 16 ~ 31.<br>When the DM9000 is set to double word mode, the strap pin EEDO is pulled high and WAKEUP is not pull-high. |
| N/A | IO32# (in double word mode) | 57 | O | Double WORD Command Indication<br>This pin is used as the double word command indication when the DM9000 is set to double word data mode and this pin will be asserted when the access of internal memory is double word width.<br>This pin is low active at default |

**Table 2.3 Pin Function Table for Parallel Interface**

### 2.2.2 I/O Base Address Decoding

The I/O base address for processor bus interface can be programmed from 300H to 370H. The default value of I/O base address is 300H.

| A9 | A8 | A7 | A6 | A5 | A4 | I/O base address |
|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 | 300H |
| 1 | 1 | 0 | 0 | 0 | 1 | 310H |
| 1 | 1 | 0 | 0 | 1 | 0 | 320H |
| 1 | 1 | 0 | 0 | 1 | 1 | 330H |
| 1 | 1 | 0 | 1 | 0 | 0 | 340H |
| 1 | 1 | 0 | 1 | 0 | 1 | 350H |
| 1 | 1 | 0 | 1 | 1 | 0 | 360H |
| 1 | 1 | 0 | 1 | 1 | 1 | 370H |

**Table 2.4 I/O Based Address for Parallel Interface**

The chart below shows the decoding of I/O based address 300H:

| A15 | A14 | A13 | A12 | A11 | A10 | A9 | A8 | A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### 2.2.3 Processor Parallel Bus 8/ 16/ 32-Bit Mode and MII Interface Setting

The DM9000 provides three I/O modes: byte/ word/ dword of memory commands for the processor parallel bus to read/ write the data from hardware DATA port. The decoder table is shown as follows.

| WAKEUP | EEDO | Data width |
|---|---|---|
| 0 | 0 | 16-bit |
| 0 | 1 | 32-bit |
| 1 | 0 | 8-bit |
| 1 | 1 | Reserved |

The EEDO pin is also used as a strap pin. It combines with another strap pin WAKEUP, and it can set the data width of the internal memory access for RX/ TX packets. The logic "1" means the strap pin is pulled high.

In the DM9000, the 32-bit I/O mode SD [16:31] and MII interface share the same pins. If the designer sets the WAKEUP and EEDO pins to the 32-bit mode, the external MII interface function will be disabled automatically.

The interrupt status register ISR (REG_FE) can help us to check the I/O mode setting.

| ISR (REG_FE) Bit 7 | 6 | IOMODE (data width) |
|---|---|---|
| 0 | 0 | 16-bit mode |
| 0 | 1 | 32-bit mode |
| 1 | 0 | 8-bit mode |
| 1 | 1 | Reserved |

### 2.2.4 Command Type

The CMD pin is used to set the COMMAND type. When the CMD pin is pulled high, the access of this commands cycle is for DATA port. If pulled low, the access of this command cycle is for ADDRESS port. The following diagram (Figure 2.3) is an example for the DM9000 CMD pin connected to an embedded system.

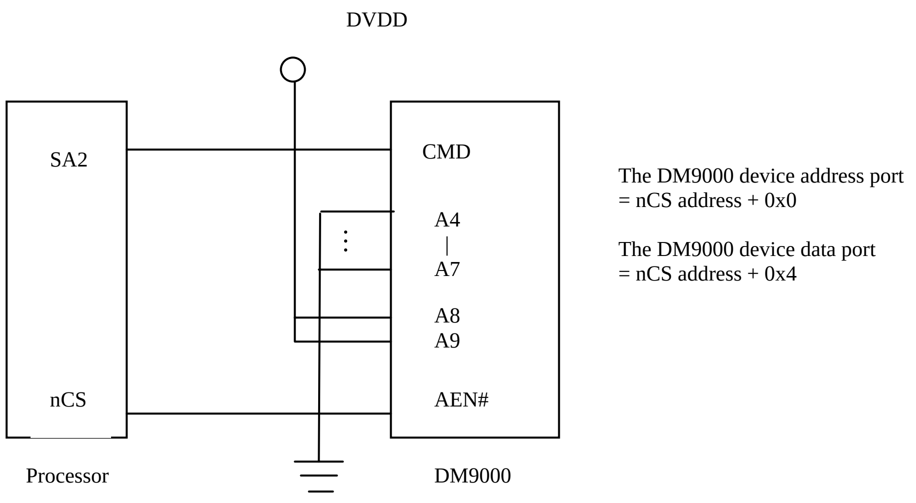


**Figure 2.3 DM9000 CMD Pin and Processor Parallel Bus.**

# 3 System Hardware Design

## 3.1 How to Select the DM9000

The AEN is an input pin. It is low active used to select the DM9000 LAN chip. The SA4 ~ SA9 are the address bus 4 ~ 9. When SA9 and SA8 are high, SA7 and AEN are low, and SA6, SA5, SA4 are matched with strap pins TXD [2:0], and then the DM9000 is selected.

## 3.2 How to Change I/O Base Address

The default I/O base address of the DM9000 is 300H. Once everything setup properly, the system designers can change the I/O base address to other value by writing to the EEPROM word 6 bit[11:9] with an available I/O base address, then write "01" to the EEPROM word 3 bit[5:4] to enable the setting of word 6 bit[11:9]. After all these steps completed successfully, the system designer may power off and restart the device that was originally assigned to I/O base address 300H. The DM9000 will load the proper I/O base address from the serial EEPROM at power-on.

If system designers design the system without EEPROM, the I/O base address still can be changed. The pins TXD [2:0] are used to be the strap pins of modifying the I/O base address. The I/O base address = (strap pin value of TXD [2:0]) * 10H + 300H.

Besides, it is worthy of note that if the system designer designs the system with 32-bit mode without EEPROM, the TXD[2:0] will be used as the data port connected to the host; but according to description above, the TXD[2:0] is also the strap pins for modifying the I/O base address. Therefore, the DM9000 cannot be selected with correct I/O base address. We recommend that if the system designer uses the strap pins TXD [2:0] as the data port, it is better to load the EEPROM on the design device. It is because the I/O base address of the EEPROM will not be changed by strap pins TXD [2:0] accordingly. The priority of the EEPROM is higher than that of the strap pins TXD [2:0] for setting the I/O base address. The priority is EEPROM > IO strap pins > default setting.

### 3.2.1 Strap Pins Setting

The DM9000 provides 7 strap pins for the system designs follows. Note: the I/O base address would not be changed by the strap pins TXD [2:0] in the 32-bit mode. And, the INT polarity also would not be modified by the strap pin 57 MDC in the 32-bit mode. Because of the pin configuration, from the pin 37 LINK_I to the pin 57 MDC originally, for the 32-bit data bus now.

**Strap pins control list:**

| Pin | Name | Strap | Description |
|---|---|---|---|
| 50~52 | TXD[2:0] | I/O base address | External MII Transmit Data, TXD [2:0] can modify the I/O base address. The I/O base address = (strap pin value of TXD [2:0]) * 10H + 300H. Note: When SA9 and SA8 are high, SA7 and AEN are low, and SA6, SA5, SA4 are matched to the TXD [2:0] 3 pins, and then the DM9000 is selected in the processor mode. |
| 57 | MDC | INT active high/low | MII Serial Management Data Clock, MDC can modify INT pin 100 polarity at default active high. When the MDC pin is pulled high, the INT pin is low active; otherwise the INT pin is high active. |
| 65 | EEDO | 32-bit DATA I/O | Data to EEPROM, EEDO combines with strap pin WAKEUP, and it can set the data width of the internal memory access for RX/ TX packets.<br>The decoder table is the following, where the logic "1" means the strap pin is pulled high.<br>WAKEUP EEDO Data width<br>0 0 16-bit<br>0 1 32-bit<br>1 0 8-bit<br>1 1 Reserved. |
| 67 | EECS | LED mode 0/ 1 | This pin is Chip Select to EEPROM and is also used as a strap pin to define the LED modes. When it is pulled high, the LED mode is mode 1; otherwise it is mode 0. |
| 79 | WAKEUP | 8-bit DATA I/O | It combines with EEDO, the strap pin 65 above. |

**Table 3.1 Strap Pin Control Table**

### 3.3 Serial EEPROM Operation

The DM9000 supports a serial EEPROM interface. The EEPROM of the DM9000 holds the following parameters:

1. Ethernet individual address.

2. I/O base address.
3. uP/ ISA control bus pins polarity setting.
4. Wake-up mode control.

All of the above mentioned values are read from the EEPROM upon hardware power-on reset. For example, the specific target device is IC 93C46, the 1024-bit serial EEPROM. And the EEPROM of the DM9000 is 93LC46 such as support ATmel, Micro-chip, ATC, and CSI. All EEPROM accesses are done in words. All EEPROM addresses in the specification are specified as word addresses.

### 3.3.1 EEPROM Format

| Name | Word | Offset | Programming Value (Hex) | Description |
|---|---|---|---|---|
| MAC Address | 0 | 0 ~ 5 | 00<br>XX<br>XX<br>XX<br>XX<br>XX | 6-byte Ethernet address. |
| Auto Load Control | 3 | 6 ~ 7 | 1455 | Bit [1:0] = 00: Disable vendor ID and product ID programming.<br>* 01: Enable vendor ID and product ID programming.<br>Bit [3:2] = 00: Disable setting of word 6 bit [8:0].<br>* 01: Enable setting of word 6 bit [8:0].<br>Bit [5:4] = 00: Disable setting of word 6 bit [11:9].<br>* 01: Enable setting of word 6 bit [11:9].<br>Bit [7:6] = 00: Disable setting of word 7 bit [3:0].<br>* 01: Enable setting of word 7 bit [3:0].<br>Bit [9:8] = 00: Reserved.<br>Bit [10] = 0: Disable setting of word 7 bit [7].<br>* 1: Enable setting of word 7 bit [7].<br>Bit [11] = 0: Reserved.<br>Bit [12] = 0: Disable setting of word 7 bit [8].<br>* 1: Enable setting of word 7 bit [8].<br>Bit [13] = 0: Reserved.<br>Bit [15:14] = 00: Reserved.<br>Note: The remark * is now programming value. |
| Vendor ID | 4 | 8 ~ 9 | 0A46 | 2-byte vendor ID. |
| Product ID | 5 | 10~11 | 9000 | 2-byte product ID. |
| Pin Control | 6 | 12~13 | 01E6 | When word 3 bit[3:2]= 01, these bits can control the IOR, IOW , INT, IOWAIT and IO16 pins polarity:<br>Bit [0] = 0: Reserved.<br>Bit [1] = 0: uP/ ISA IOR pin is active high.<br>* 1: uP/ ISA IOR pin is active low.<br>Bit [2] = 0: uP/ ISA IOW pin is active high.<br>* 1: uP/ ISA IOW pin is active low.<br>Bit [3] = * 0: uP/ ISA INT pin is active high. |

| | | | | |
|---|---|---|---|---|
| | | | | 1: uP/ ISA INT pin is active low.<br>Bit [4] = * 0: uP/ ISA INT pin is force output.<br>1: uP/ ISA INT pin is force open-collected.<br>Bit [5] = 0: uP/ ISA IOWAIT pin is active high.<br>* 1: uP/ ISA IOWAIT pin is active low.<br>Bit [6] = 0: uP/ ISA IOWAIT pin is force output.<br>* 1: uP/ ISA IOWAIT pin is force open-collected.<br>Bit [7] = 0: uP/ ISA IO16 pin is active high.<br>* 1: uP/ ISA IO16 pin is active low.<br>Bit [8] = 0: uP/ ISA IO16 pin is force output.<br>* 1: uP/ ISA IO16 pin is force open-collected.<br>When word 3 bit[5:4]= 01, the I/O base can be re-configured:<br>Bit[11:9]= 000: Default I/O base address = 300H<br>*000 : 300H<br>001 : 310H<br>010 : 320H<br>011 : 330H<br>100 : 340H<br>101 : 350H<br>110 : 360H<br>111 : 370H<br>Bit [15:12] = 0000: Reserved. |
| Wake-up Mode Control | 7 | 14~15 | 0180 | Depend on the setting of word 3 bit[15:6] to accept auto load control:<br>Bit [0] = * 0: The WAKEUP pin is active high.<br>1: The WAKEUP pin is active low.<br>Bit [1] = * 0: The WAKEUP pin is in level mode.<br>1: The WAKEUP pin is in pulse mode.<br>Bit [2] = * 0: Magic packet wakeup event is disable.<br>1: Magic packet wakeup event is enabled.<br>Bit [3] = * 0: Link_change wakeup event is disable.<br>1: Link_change wakeup event is enabled.<br>Bit [6:4] = 000: Reserved.<br>Bit [7] = 0: LED mode 0.<br>* 1: LED mode 1.<br>Bit [8] = 0: The internal PHY is disabled after power-on.<br>* 1: The internal PHY is enable after power-on.<br>The GPR bit [0] and the GPIO0 pin 68 are modified from this bit.<br>Bit [15:9] = 0000000: Reserved. |
| Reserved | 8~63 | 16-127 | - | Programmable by user. |

**Note: The programming value of the EEPROM is for reference only. For example: "00:ox:ox:ox:ox:ox, 55, 14, 46, 0A, 00, 90, E6, 01, 80, 01" is an EEPROM value list.**

**Table 3.2 EEPROM Format**

## 3.4 GPIO Pins Setting

The DM9000 provides 4 GPIO pins for the system design. The default value of GPCR bit [0] is "1" GEP_CNTL0=1 for GPIO0 as the output mode, and outputting a high signal "1" GEPIO0=1 will power down the internal PHY or other external MII/ RMII device. Let GEPIO0=0 turn it on. The default values of GPCR bit [3:1] are all "0" for GPIO1 ~ 3 as the input mode respectively. The two registers, GPCR (REG_1E) and GPR (REG_1F), are used to program high/ low value of the output/ input port for the GPIO0 ~ GPIO3 pins at pin 68~71.

**GPCR (REG_1E) GPIO interface control:**

| Bit | Name | Default | Description |
|---|---|---|---|
| 7:4 | RESERVED | 0,RO | Reserved. |
| 3 | GEP_CNTL3 | 0,RW | Setting the input/output port of GPIO3 pin 71. "1" is represented the output port, "0" is represented the input port. |
| 2 | GEP_CNTL2 | 0,RW | Setting the input/output port of GPIO2 pin 70. "1" is represented the output port, "0" is represented the input port. |
| 1 | GEP_CNTL1 | 0,RW | Setting the input/output port of GPIO1 pin 69. "1" is represented the output port, "0" is represented the input port. |
| 0 | GEP_CNTL0 | 1,RW | Setting the input/output port of GPIO0 pin 68. "1" is represented the output port, "0" is represented the input port. If the GPIO0 is set to output port, the power management of the internal PHY can be controlled by GPR bit [0]. |

**Table 3.3 General Purpose Control Register (GPCR) Table**

**GPR (REG_1F) GPIO interface control:**

| Bit | Name | Default | Description |
|---|---|---|---|
| 7:4 | RESERVED | 0,RO | Reserved. |
| 3 | GEPIO3 | 0,RW | If GPIO3 is output port, GPR bit [3] sets to "1" to enable the pin 71 output high or sets to "0" to enable the pin 71 output low.<br>If GPIO3 is input port, the value of GPR bit [3] is "1" to represent a high signal is received. In contract, the value of GPR bit [3] is "0" to represent a low signal is received. |
| 2 | GEPIO2 | 0,RW | If GPIO2 is output port, GPR bit [2] sets to "1" to enable the pin 70 output high or sets to "0" to enable the pin 70 output low.<br>If GPIO2 is input port, the value of GPR bit [2] is "1" to represent a high signal is received. In contract, the value of GPR bit [2] is "0" to represent a low signal is received. |
| 1 | GEPIO1 | 0,RW | If GPIO1 is output port, GPR bit [1] sets to "1" to enable the pin 69 output high or sets to "0" to enable the pin 69 output low.<br>If GPIO1 is input port, the value of GPR bit [1] is "1" to represent a high signal is received. In contract, the value of GPR bit [1] is "0" to represent a low signal is received. |

| | | | |
|---|---|---|---|
| 0 | GEPIO0 | 1,RW | If GPIO0 is output port, GPR bit [0] sets to "1" to enable the pin 68 output high and power down the internal PHY, or sets to "0" to enable the pin 68 output low and power on the internal PHY.<br>If GPIO0 is input port, the value of GPR bit [0] is "1" to represent a high signal is received. In contract, the value of GPR bit [0] is "0" to represent a low signal is received. |

**Table 3.4 General Purpose Register (GPR) Table**

## 3.5 Timing Analysis

The timing diagrams and parameter tables are shown below for the READ and WRITE cycle presented to the DM9000 device.

### 3.5.1 READ Cycle

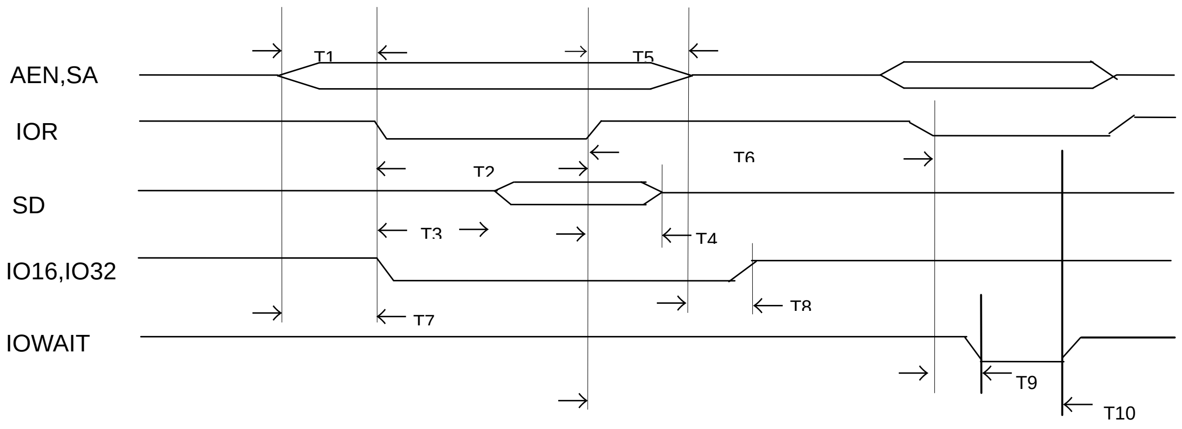


**Figure 3.1 Processor Register READ Timing**

| Symbol | Parameter | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|
| T1 | System address valid to IOR valid | 5 | | | ns |
| T2 | IOR width | 22 | | | ns |
| T3 | SD valid after IOR ready | | | 10 | ns |
| T4 | IOR invalid to SD invalid | | | 4 | ns |
| T5 | IOR invalid before system address invalid | 5 | | | ns |
| T6 | IOR invalid to next IOR valid （access DM9000） | 80 | | | ns |
| T7 | System address valid to IO16,IO32 valid | | | 5 | ns |
| T8 | System address invalid to IO16, IO32 invalid | | | 5 | ns |
| T9 | If T6 < 80ns, IOWAIT asserted | | 5 | | ns |
| T10 | Last IOR de-asserted to IOWAIT de-asserted | | | 84 | ns |

**Table 3.5 Parameters for READ Cycle**

### 3.5.2 WRITE Cycle

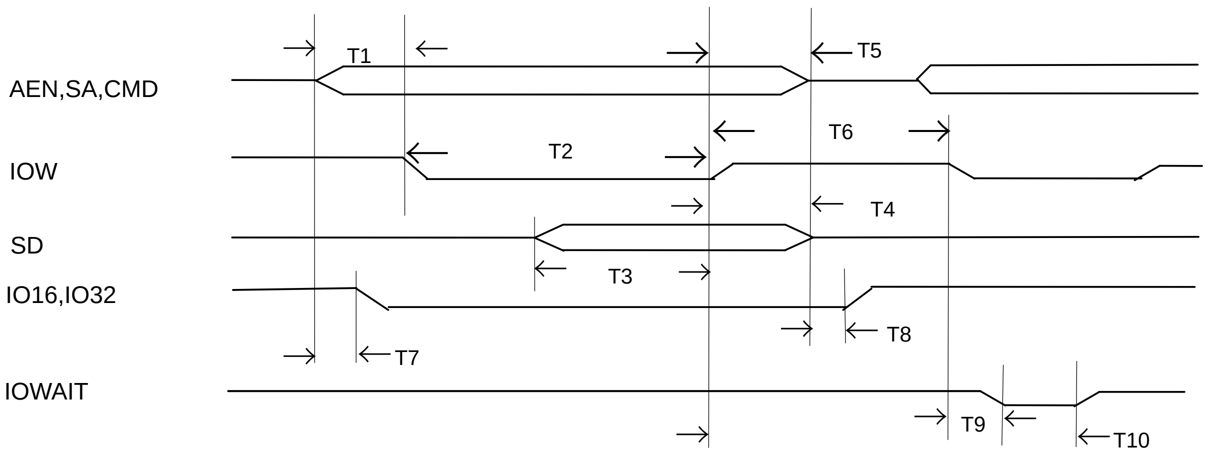


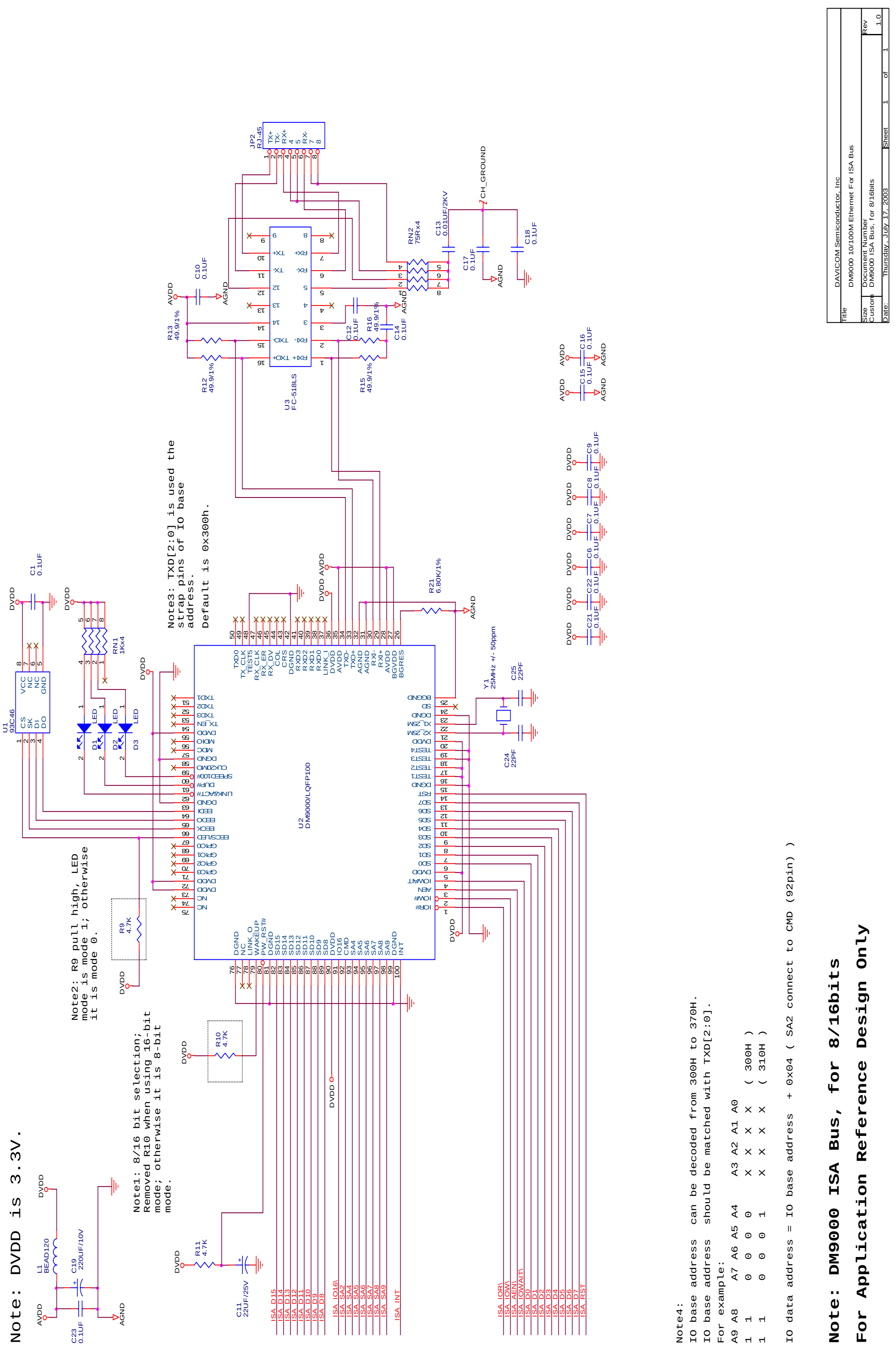
**Figure 3.2 Processor Register WRITE Timing**

| Symbol | Parameter | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|
| T1 | System address valid to IOW valid | 5 | | | ns |
| T2 | IOW width | 22 | | | ns |
| T3 | SD setup time | 22 | | | ns |
| T4 | SD hold time | 2 | | | ns |
| T5 | IOW invalid before system address invalid | 5 | | | ns |
| T6 | IOW invalid to next IOW valid （access DM9000） | 80 | | | ns |
| T7 | System address valid to IO16,IO32 valid | | | 5 | ns |
| T8 | System address invalid to IO16, IO32 invalid | | | 5 | ns |
| T9 | If T6 < 80ns, IOWAIT asserted | | 5 | | ns |
| T10 | Last IOW de-asserted to IOWAIT de-asserted | | | 84 | ns |

**Table 3.6 Parameters for WRITE Cycle**

## 3.6 Schematic Reference Design

### 3.6.1 DM9000 ISA Bus for 8/ 16-Bit


DAVICOM Semiconductor, Inc.
Title: DM9000 10/100M Ethernet For ISA Bus
Size: Custom
Document Number: DM9000 ISA Bus, for 8/16bits
Rev: 1.0
Date: Thursday, July 17, 2003
Sheet 1 of 1
U2 DM9000/LQFP100
U3 FC-518LS
JP2 RJ-45
U1 93C46
RN1 1Kx4
RN2 75Rx4
Y1 25MHz +/- 50ppm
L1 BEAD120
CH_GROUND
DVDD
AVDD
AGND

```
Note: DVDD is 3.3V.

Note1: 8/16 bit selection;
Removed R10 when using 16-bit
mode; otherwise it is 8-bit
mode.

Note2: R9 pull high, LED
mode is mode 1; otherwise
it is mode 0.

Note3: TXD[2:0] is used the
strap pins of IO base
address.
Default is 0x300h.

Note4:
IO base address  can be decoded from 300H to 370H.
IO base address  should be matched with TXD[2:0].
For example:
A9 A8    A7 A6 A5 A4    A3 A2 A1 A0
1  1     0  0  0  0     X  X  X  X    ( 300H )
1  1     0  0  0  1     X  X  X  X    ( 310H )

IO data address = IO base address  + 0x04 ( SA2 connect to CMD (92pin) )
```

**Note: DM9000 ISA Bus, for 8/16bits**
**For Application Reference Design Only**

**Figure 3.3 Schematic of ISA Bus for 8/ 16-Bit**

## 3.6.2 DM9000 Host Parallel Bus for 8-Bit

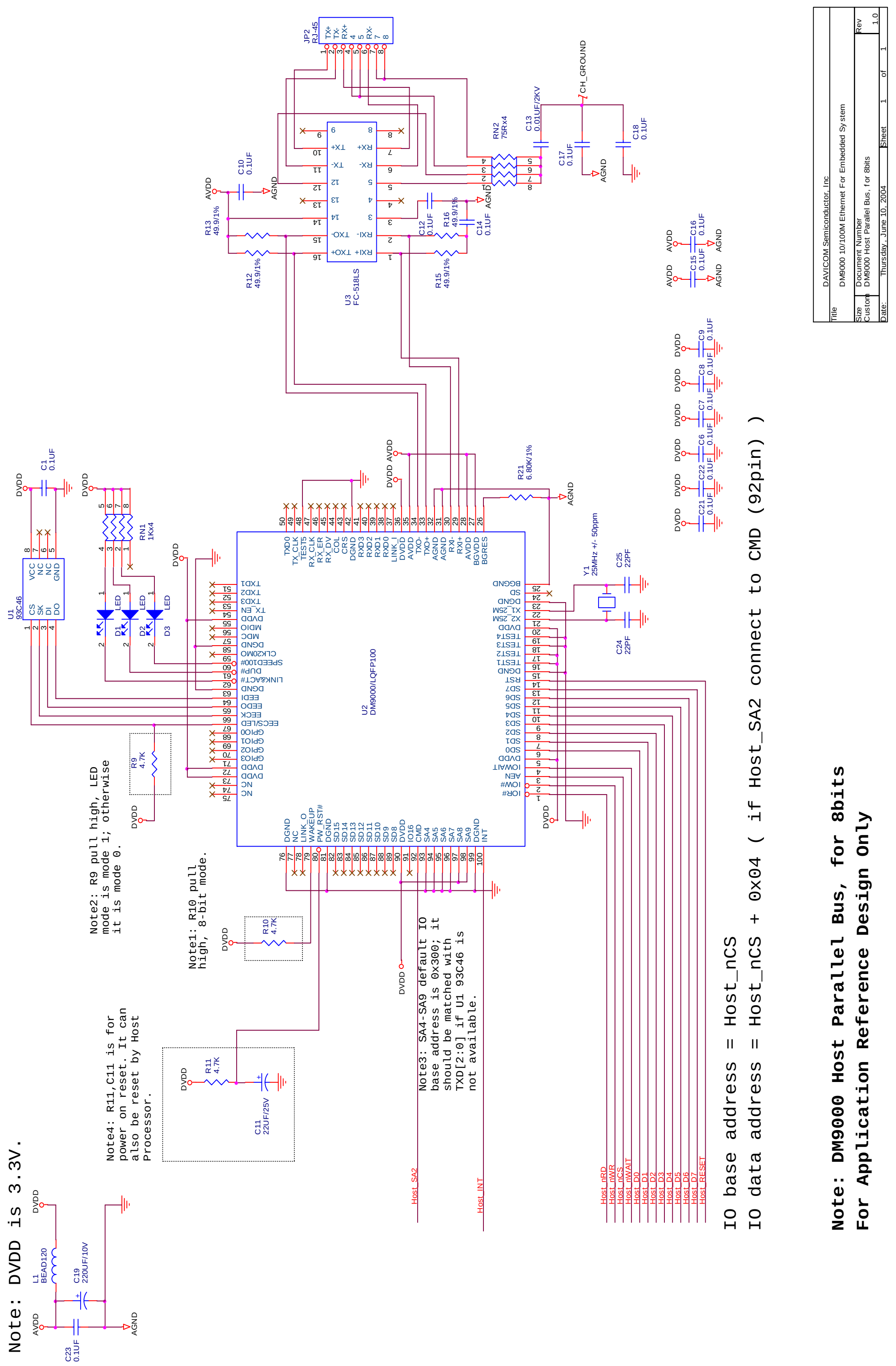

Note: DVDD is 3.3V.
Note4: R11,C11 is for power on reset. It can also be reset by Host Processor.
Note2: R9 pull high, LED mode is mode 1; otherwise it is mode 0.
Note1: R10 pull high, 8-bit mode.
Note3: SA4-SA9 default IO base address is 0x300; it should be matched with TXD[2:0] if U1 93C46 is not available.
IO base address = Host_nCS
IO data address = Host_nCS + 0x04 ( if Host_SA2 connect to CMD (92pin) )
Note: DM9000 Host Parallel Bus, for 8bits For Application Reference Design Only
U2 DM9000/LQFP100
U1 93C46
U3 FC-518LS
JP2 RJ-45
Y1 25MHz +/- 50ppm
DAVICOM Semiconductor, Inc.
DM9000 10/100M Ethernet For Embedded System
DM9000 Host Parallel Bus, for 8bits
Thursday, June 10, 2004
Rev 1.0

Figure 3.4 Schematic of Host Parallel Bus for 8-Bit

### 3.6.3 DM9000 Host Parallel Bus for 8/ 16-Bit

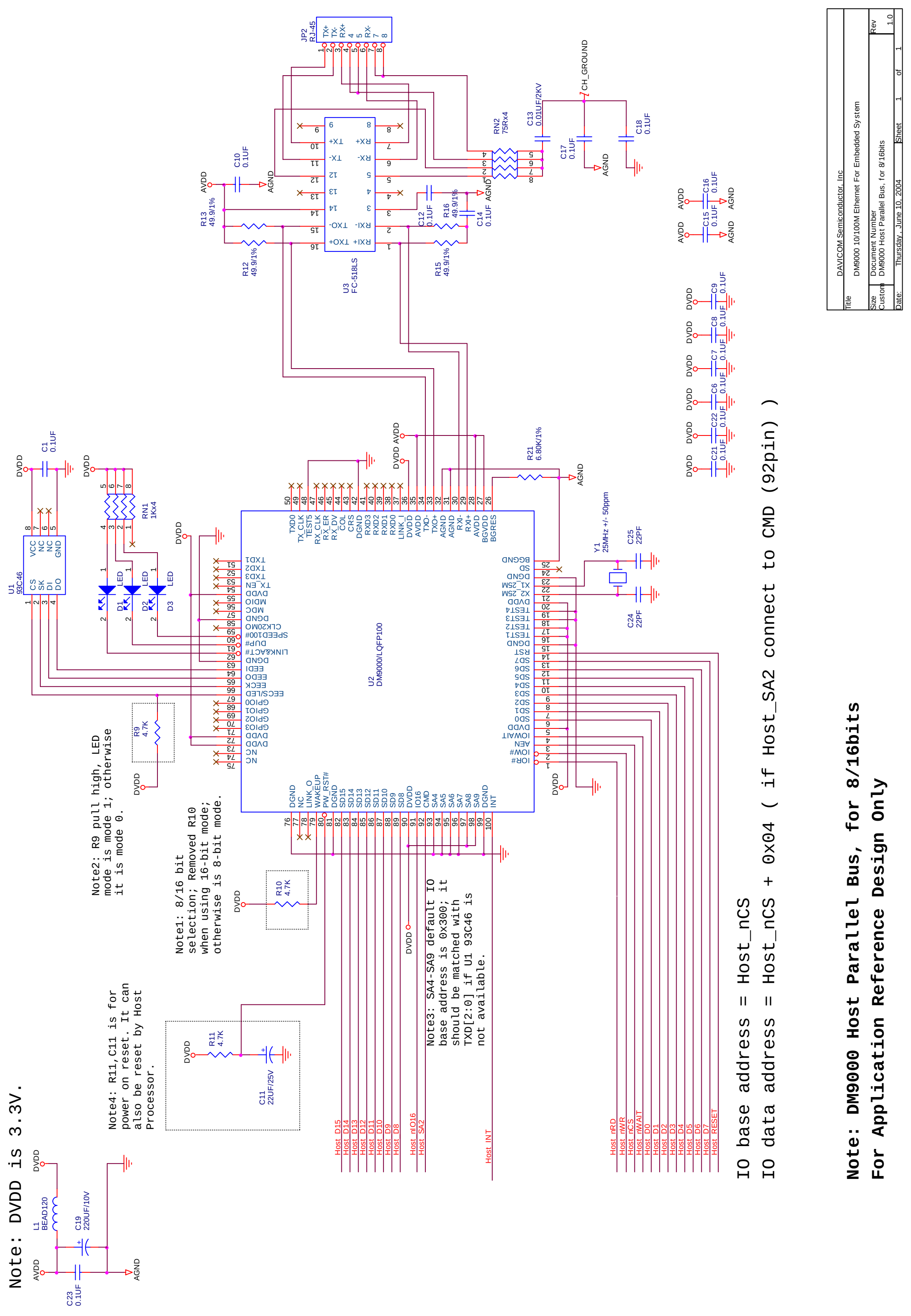
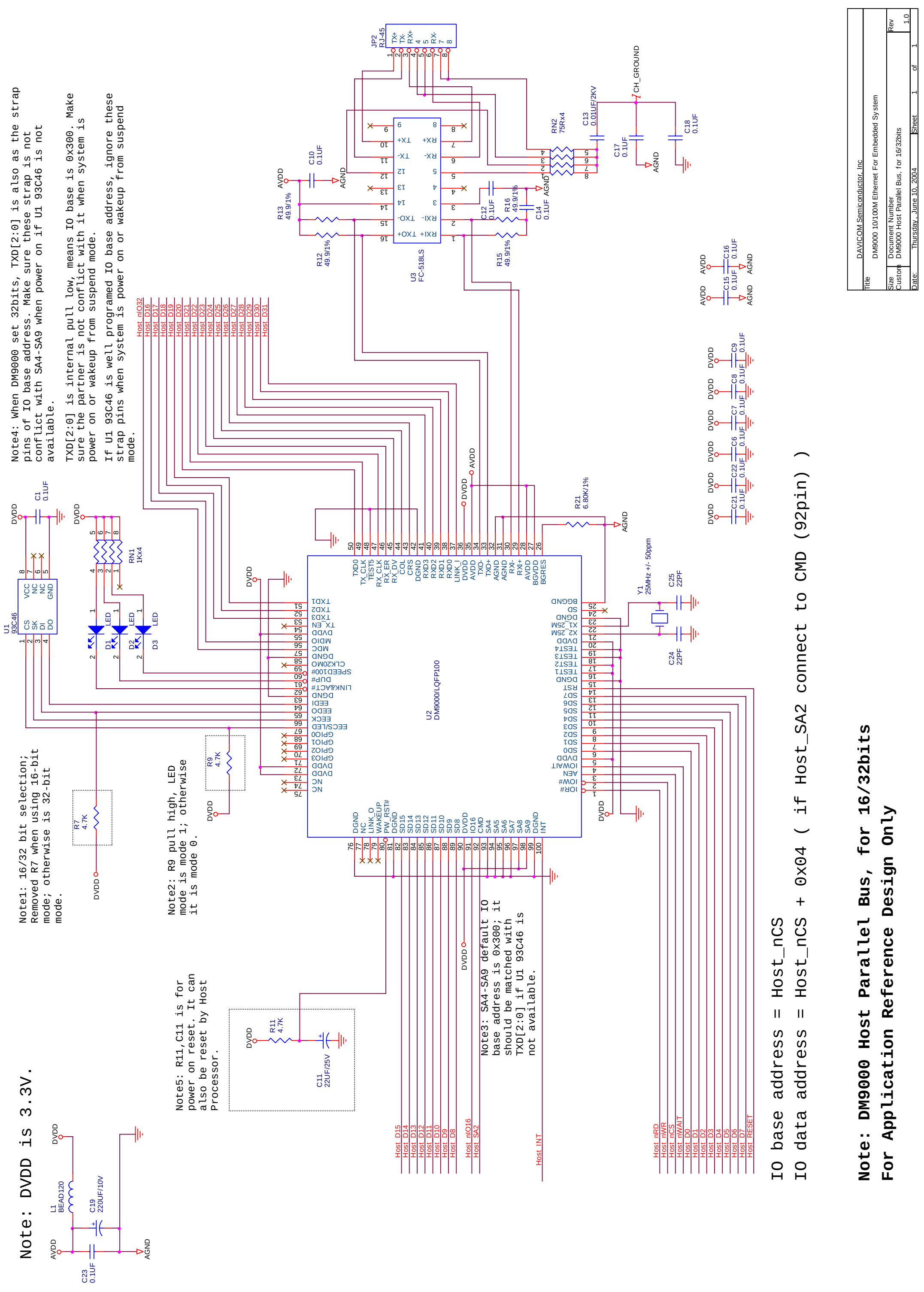

Note: DVDD is 3.3V.
Note4: R11,C11 is for power on reset. It can also be reset by Host Processor.
Note1: 8/16 bit selection; Removed R10 when using 16-bit mode; otherwise is 8-bit mode.
Note2: R9 pull high, LED mode is mode 1; otherwise it is mode 0.
Note3: SA4-SA9 default IO base address is 0x300; it should be matched with TXD[2:0] if U1 93C46 is not available.
IO base address = Host_nCS
IO data address = Host_nCS + 0x04 ( if Host_SA2 connect to CMD (92pin) )
Note: DM9000 Host Parallel Bus, for 8/16bits
For Application Reference Design Only
U2 DM9000/LQFP100
U3 FC-518LS
JP2 RJ-45
U1 93C46
Y1 25MHz +/- 50ppm
DAVICOM Semiconductor, Inc
DM9000 10/100M Ethernet For Embedded System
DM9000 Host Parallel Bus, for 8/16bits
Thursday, June 10, 2004
Sheet 1 of 1
Rev 1.0

Figure 3.5 Schematic of Host Parallel Bus for 8/ 16-Bit

### 3.6.4 DM9000 Host Parallel Bus for 16/ 32-Bit


DAVICOM Semiconductor, Inc.
DM9000 10/100M Ethernet For Embedded System
DM9000 Host Parallel Bus, for 16/32bits
Thursday, June 10, 2004
U2 DM9000/LQFP100
U1 93C46
U3 FC-518LS
JP2 RJ-45

Note4: When DM9000 set 32bits, TXD[2:0] is also as the strap pins of IO base address. Make sure these strap is not conflict with SA4-SA9 when power on if U1 93C46 is not available.

TXD[2:0] is internal pull low, means IO base is 0x300. Make sure the partner is not conflict with it when system is power on or wakeup from suspend mode.

If U1 93C46 is well programed IO base address, ignore these strap pins when system is power on or wakeup from suspend mode.

Note1: 16/32 bit selection; Removed R7 when using 16-bit mode; otherwise is 32-bit mode.

Note2: R9 pull high, LED mode is mode 1; otherwise it is mode 0.

Note5: R11,C11 is for power on reset. It can also be reset by Host Processor.

Note3: SA4-SA9 default IO base address is 0x300; it should be matched with TXD[2:0] if U1 93C46 is not available.

Note: DVDD is 3.3V.

IO base address = Host_nCS

IO data address = Host_nCS + 0x04 ( if Host_SA2 connect to CMD (92pin) )

**Note: DM9000 Host Parallel Bus, for 16/32bits**
**For Application Reference Design Only**

**Figure 3.6 Schematic of Host Parallel Bus for 16/ 32-Bit**

## 3.6.5 DM9000 Host Parallel Bus to MII Adapter

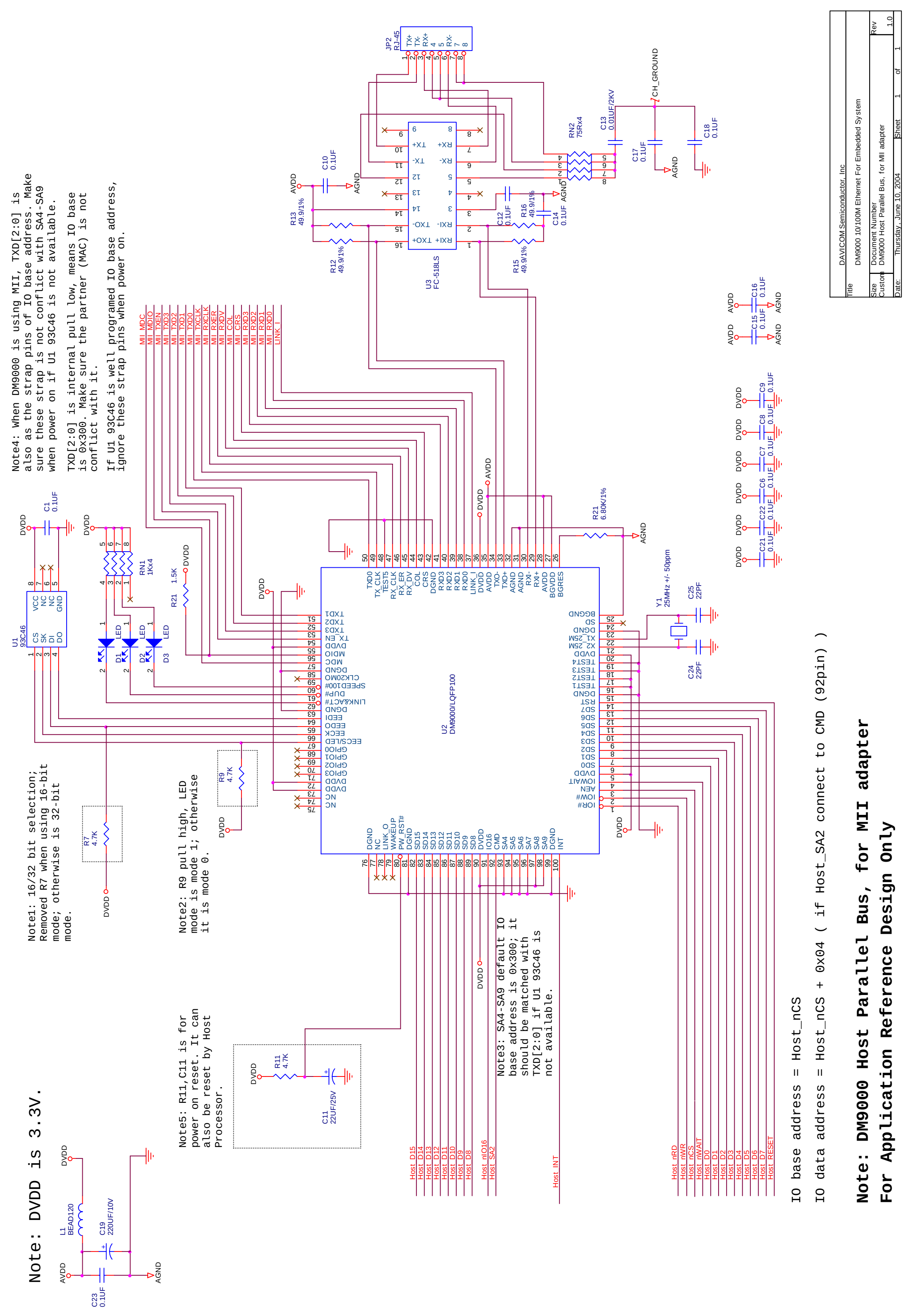

Note4: When DM9000 is using MII, TXD[2:0] is also as the strap pins of IO base address. Make sure these strap is not conflict with SA4-SA9 when power on if U1 93C46 is not available.
TXD[2:0] is internal pull low, means IO base is 0x300. Make sure the partner (MAC) is not conflict with it.
If U1 93C46 is well programmed IO base address, ignore these strap pins when power on.
Note1: 16/32 bit selection; Removed R7 when using 16-bit mode; otherwise is 32-bit mode.
Note2: R9 pull high, LED mode is mode 1; otherwise it is mode 0.
Note3: SA4-SA9 default IO base address is 0x300; it should be matched with TXD[2:0] if U1 93C46 is not available.
Note5: R11,C11 is for power on reset. It can also be reset by Host Processor.
Note: DVDD is 3.3V.
IO base address = Host_nCS
IO data address = Host_nCS + 0x04 ( if Host_SA2 connect to CMD (92pin) )
Note: DM9000 Host Parallel Bus, for MII adapter
For Application Reference Design Only
U2 DM9000/LQFP100
U3 FC-518LS
JP2 RJ-45
U1 93C46
DAVICOM Semiconductor, Inc.
DM9000 10/100M Ethernet For Embedded System
Document Number DM9000 Host Parallel Bus, for MII adapter
Rev 1.0
Date: Thursday, June 10, 2004
Sheet 1 of 1

Figure 3.7 Schematic of Host Parallel Bus to MII Adapter

# 4 Reset Operation and PHY Power-down Mode

The DM9000 can be reset by either hardware reset or software reset. A hardware reset can be accomplished by power-on active low or asserting high the RST pin 14. A software reset can be accomplished by setting the network control register (NCR REG_00) RST bit [0] =1.

The internal PHY of the DM9000 can be powered down by writing "1" to GEPIO0 bit[0] of the GPR register, or by setting the power down bit[11]=1 in the basic mode control register (BMCR) of the MII registers. In the power-down state, the power consumption is reduced to a minimum under 10 mA.

## 4.1 Hardware Reset

Both the MAC and the PHY are reset when the hardware reset operation is accomplished.

### 4.1.1 Power On Reset

An active low signal is used to reset the DM9000 LAN chip. The PW_RST# pin 80 is asserted low for at least 20 ms. all of the MAC and PHY registers will be reset to the default values and the hardware strap pins will also be latched. The DM9000 is ready after 5 us when this pin is de-asserted and then the data are downloaded from EEPROM.

### 4.1.2 Processor Reset

The DM9000 can be reset by the GPIO pin of the host processor. The RST pin 14 is an input pin and asserted high for at least 2 us to reset the DM9000 LAN chip. All of the MAC registers will be reset to the default values.

## 4.2 Software Reset

A software reset can be accomplished by setting RST bit [0] =1 in the network control register (NCR REG_00). After the reset, some registers will be reset to their default value.

## 4.3 PHY Power Down Mode

In the power-down mode, the DM9000 will disable all transmit, receive and MII interface function except the MDC/ MDIO management interface. There are two ways to set the power-down mode.

### 4.3.1 MII Register Setting

In the MII registers, the basic mode control register (BMCR) power down bit[11] can be set high "1" to enable the PHY power-down mode.

### 4.3.2 GPIO0 Setting

GPIO0 is used for powering down the internal PHY, and it's the pin 68 default output high. Once, if the PHY is desired to be activated, the system driver needs to clear this power down bit, GEPIO0 bit [0] of the GPR register, by writing "0".

# 5 How to Program DM9000

## 5.1 How to Read/ Write DM9000 Register

The DM9000 supports 8 different sets of I/O base address: 0x300/ 0x304, 0x310/ 0x314, 0x320/ 0x324, 0x330/ 0x334, 0x340/ 0x344, 0x350/ 0x354, 0x360/ 0x364 and 0x370/ 0x374. The default value of I/O base address is 0x300/ 0x304. It also can be re-configured to one of the sets for another I/O base address, by power-on downloaded the word 6 bit[11:9] pin control from the serial EEPROM 93C46, or by latched from the strap pins TXD[2:0].

There are only two addressing ports through the access of the host interface. One port is ADDRESS port and the other is DATA port. The ADDRESS port is decoded by the pin CMD=0 and the DATA port is decoded by the pin CMD=1. The content of the ADDRESS port is the address of the register to access the DATA port later. Before accessing the 8-bit data in any register or the 8/ 16/ 32-bit data in the RX/ TX FIFO SRAM (total 16K bytes), the address of the register must save into the ADDRESS port. And, the address of the register is also named "port number". Please refer to the DM9000 datasheet ch.9.1 about host interface.

Here are the examples to read and write the DM9000 register:

```
u8 ior(int IOaddr, int reg) {
   delay (STD_DELAY);
   outb (reg, IOaddr);     /* I/O output a byte to ADDRESS port, select the register*/
   delay (STD_DELAY);
   return inb(IOaddr + 4);/*I/O input a byte from DATA port, READ the register data*/
}

void iow(int IOaddr, int reg, u8 dataB) {
   outb(reg, IOaddr);      /* I/O output a byte to ADDRESS port, select the register*/
   delay(STD_DELAY);
   outb(dataB, IOaddr+4);  /* I/O output a byte to DATA port, WRITE the register data*/
   delay(STD_DELAY);       /* default STD_DELAY=0 for the standard I/O delay */
}
```

## 5.2 Driver Initializing Steps

1. if the internal PHY is required open, the following steps are to activate it:

    i. set GPCR (REG_1E) GEP_CNTL0 bit[0]=1

    ii. set GPR (REG_1F) GEPIO0 bit[0]=0

The default status of the DM9000 is to power down the internal PHY by the value GEPIO0=1. Since the internal PHY have been powered down, the wakeup procedure will be needed to enable it. Please refer to the section 3.4 about the GPIO settings.

2. do the software reset twice to initial DM9000:

    i. set NCR (REG_00) bit[2:0]=011 for a period time, at least 20 us.

    ii. clear NCR (REG_00) bit[2:0]=000

    issue 2nd reset

    iii. set NCR (REG_00) bit[2:0]=011 for a period time, at least 20 us.

    iv. clear NCR (REG_00) bit[2:0]=000

3. program the NCR register. Choose normal mode by setting NCR (REG_00) LBK bit[2:1]=00. The system designer can choose the network operations such as setting the internal/ external PHY, the full/ half duplex mode, the loop-back mode, and enable wakeup event. Please refer to the datasheet ch.6.1 about setting the NCR register.

4. clear TX status by reading the NSR register (REG_01). The bit [2] TX1END, bit [3] TX2END, and bit [5] WAKEST will be automatically cleared by reading it or writing "1". Please refer to the datasheet ch.6.2 about the NSR register setting.

5. Read the EEPROM saved data.

6. Write Node address 6 bytes into the physical address registers (REG_10 ~ REG_15).

7. Write Hash table 8 bytes into the multicast address registers (REG_16 ~ REG_1D).

8. set the IMR register (REG_FF) PAR bit[7]=1 to enable the Pointer Automatic Return function, which is the memory read/ write address pointer of the RX/ TX FIFO SRAM.

9. depend on OS and DDK of the system to handle the NIC interrupt or polling service.

10. Program the IMR register (REG_FF) PRM bit [0]/ PTM bit [1] to enable the RX/ TX interrupt. Before doing this, the system designer needs to register the interrupt handler routine. For example, if the driver needs to generate the interrupt after one package is transmitted, the interrupt mask register IMR PTM bit [1] =1 will be set. And, if the interrupt is generated after the DM9000 received one new packet incoming, IMR PRM bit [0] should be set to "1".

11. Program the RCR register to enable RX. The RX function is enabled by setting the RX control register (REG_05) RXEN bit [0] =1. The choice of the other bits bit [6:0] is depended on the system design. Please refer to the datasheet ch.6.6 about setting the RCR register.

12. NIC is being activated and ready RX/ TX now.

## 5.3 How to Read/ Write EEPROM Data

The DM9000 supports the serial EEPROM interface and provides a very easy method to access it. It is only to read/ write the EEPROM&PHY control/ address/ data register (REG_0B ~ REG_0E), which are used to control and read/ write the EEPROM address or data.

For example, IC 93C46 is a 64-word (1024-bit) EEPROM and the word addresses are mapped to the EROA bit[5:0] of the EEPROM&PHY address register (EPAR REG_0C). Besides, EPAR PHY_ADR bit [7:6] must be set "00" to enable the word address for the EEPROM. And, the setting PHY_ADR bit [7:6] =01 of EPAR is for the PHY address selected. Please refer to the datasheet ch.6.12 ~ ch.6.14 about the EEPROM&PHY registers setting.

### 5.3.1 Read EEPROM Data

The following steps are shown to read data from the serial EEPROM (SROM):

1. Write the word address into EPAR (REG_0C).
2. Write command=0x04 into the EEPROM&PHY Control Register (EPCR REG_0B) to start the SROM + READ operation:
   i. EPCR (REG_0B) EPOS bit [3] =0: select SROM mode (this is default: 0).
   ii. EPCR (REG_0B) ERPRR bit [2] =1: issue READ command.
3. Read EPCR (REG_0B) and wait until ERRE bit [0] =0 ok, or just following the step 4.

4. Wait at least 20~500 us, then write "0" into EPCR (REG_0B) to clear READ command.
5. Read the SROM data high byte from EE_PHY_H (REG_0E) and the low byte from EE_PHY_L (REG_0D) in the EEPROM&PHY Data registers.

There is an example, read the SROM word address 0x02, shown as follows:

1. write word address 0x02 into EPAR (REG_0C)

```
iow(IOaddr, 0x0C, 0x02);
```

2. write the SROM + READ command=0x04 into EPCR (REG_0B)

```
iow(IOaddr, 0x0B, 0x04);
```

3. wait until EPCR (REG_0B) bit[0]=0 ok, or just following the step 4

```
while ( (ready_eeprom = ior(IOaddr, 0x0B)) & 1 ) {; };/* detect SROM READ ready*/
```

4. delay at least 20~500 us, then write "0" into EPCR (REG_0B)

```
udelay(150);
iow(IOaddr, 0x0B, 0);
```

5. EE_PHY_H (REG_0E) keeps the SROM data high byte for the designer to read it

```
(u8) e2prom[i*2+1] = ior(IOaddr, 0x0E);
```

6. EE_PHY_L (REG_0D) keeps the SROM data low byte for the designer to read it

```
(u8) e2prom[i*2] = ior(IOaddr, 0x0D);
```

Another example, read MAC address from the SROM word address 0&1&2 :

```
For (i = 0; i < 3; i++) {          /* read the DM9000 MAC address from SROM */
     iow(IOaddr, 0x0C, i);
     iow(db, 0x0B, 0x04);
     udelay(150);
     iow(IOaddr, 0x0B, 0);
     /* read SROM word data from EPDR to device address */
     (u8) dev_addr[i*2] = ior(IOaddr, 0x0D);      /* low byte of 16-bit word */
     (u8) dev_addr[i*2+1] = ior(IOaddr, 0x0E);    /* high byte of 16-bit word */
}
```

### 5.3.2 Write EEPROM Data

The following steps are to write data into SROM:

1. write the word address into EPAR (REG_0C).
2. write the SROM data high byte into EE_PHY_H (REG_0E) and the low byte into EE_PHY_L (REG_0D).

3. write command=0x12 into EPCR (REG_0B) to start the SROM + WRITE operation:
   i. EPCR (REG_0B) WEP bit[4]=1: enable SROM WRITE operation.
   ii. EPCR (REG_0B) EPOS bit[3]=0: select SROM mode (this is default: 0).
   iii. EPCR (REG_0B) ERPRW bit[1]=1: issue WRITE command.
4. read EPCR (REG_0B) and wait until ERRE bit[0]=0 ok, or just following step 5.
5. wait at least 20~500us, then write "0" into EPCR (REG_0B) to clear WRITE command.

For example, write 0xAA55 into the word address 0x08 of the SROM reserved words:

1. write address=08 to EPAR (REG_0C)
```
iow(IOaddr, 0x0C, 0x08);
```
2. write the high byte 0xAA into EE_PHY_H and the low byte 0x55 into EE_PHY_L
```
iow(IOaddr, 0x0E, 0xAA);

iow(IOaddr, 0x0D, 0x55);
```
3. write the SROM WRITE command=0x12 into EPCR (REG_0B)
```
iow(IOaddr, 0x0B, (0x02 | 0x10) );
```
4. wait until EPCR (REG_0B) ERRE bit[0]=0 ok, or just following step 5
```
while((ready_eeprom = ior(IOaddr, 0x0B) ) & 1) {; };/* detect SROM WRITE ok */
```
5. delay at least 20~500 us, then write "0" into EPCR (REG_0B)
```
udelay(150);

iow(IOaddr, 0x0B, 0);
```

Another example, write MAC address 00:E0:63:0E:56:78 into SROM word address 0&1&2:

```
srom_write( IOaddr, 0x00, 0xE000 );          /* low_byte=00, high_byte=E0 */
srom_write( IOaddr, 0x01,0x0E63 );           /* low_byte=63, high_byte=0E */
srom_write( IOaddr, 0x02,0x7856 );           /* low_byte=56, high_byte=78 */

void srom_write( int IOaddr, int offset, u16 dataW )

{

 /* write the SROM word address into EPAR (REG_0C) */

 iow(IOaddr, 0x0C, offset);


 /* write the high byte into EE_PHY_H (REG_0E) and low byte into EE_PHY_L (REG_0D)*/

 iow(IOaddr, 0x0D, (dataW & 0xff) );         /* low byte of 16-bit word */

 iow(IOaddr, 0x0E, ( (dataW >> 8) & 0xff) ); /* high byte of 16-bit word */


 /* issue the SROM WRITE command=0x12 into EPCR (REG_0B) */

 iow(IOaddr, 0x0B, (0x02 | 0x10) );
 /* wait SROM WRITE completed then write "0" to clear the SROM WRITE command */
 udelay(150);
 iow(IOaddr, 0x0B, 0);
}
```

### 5.4 How to Read/ Write PHY Register

The DM9000 PHY supports only 32 registers, which are mapped to EPAR (REG_0C) bit[4:0]. The default value of EPAR (REG_0C) PHY_ADR bit[7:6] is "01" to select the PHY mode. There are 4 selections for the DM9000 PHY address. Force PHY_ADR bit[7:6]=01 if the PHY address is selected. And, the setting PHY_ADR bit[7:6]=00 is for the EEPROM address selected. The start value of the PHY register address is 0x40 at offset=0x00. The most three significant bit[4:2] of the PHY register address is forced to default "000b". And the default address of the PHY is "00001b". Please refer to the datasheet ch.6.12 ~ ch.6.14 about the EEPROM&PHY registers setting.

#### 5.4.1 Read PHY Register

The following steps to read data from the PHY register:

1. write the PHY register offset into EPAR (REG_0C) bit[4:0] and the PHY address "01" into EPAR (REG_0C) bit[7:6].
2. write command=0x0C into EPCR (REG_0B) to start the PHY + READ operation:
   i. EPCR (REG_0B) EPOS bit[3]=1: select PHY mode (0: select SROM, see ch.5.3)
   ii. EPCR (REG_0B) ERPRR bit[2]=1: issue READ command.
3. read EPCR (REG_0B) and wait until ERRE bit[0]=0 ok, or just following step 4.
4. wait at least 1~200 us, then write "8" into EPCR (REG_0B) to clear READ command.
5. read the PHY data high byte from EE_PHY_H (REG_0E) and the low byte from EE_PHY_L (REG_0D) in the EEPROM&PHY Data registers.

For example, read the PHY register of ANAR at offset=0x04:

1. write offset=0x04 into EPAR (REG_0C) bit[4:0] and "01" into EPAR (REG_0C) bit[7:6]
   ```
   iow(IOaddr, 0x0C, (offset= 0x04 | 0x40) );
   ```
2. write the PHY READ command=0x0C into EPCR (REG_0B)
   ```
   iow(IOaddr, 0x0B, (0x04 | 0x08) );
   ```

3. wait until EPCR REG_0B ERRE bit[0]=0 ok, or just following step 4

```
while((ready_phy = ior(IOaddr, 0x0B)) & 1) {; };    /* detect PHY READ ready */
```

4. delay at least 1~200 us, then write "8" into EPCR (REG_0B) to clear it and keep PHY

```
udelay(20);
iow(IOaddr, 0x0B, 0x08);
```

5. read the high byte from EE_PHY_H (REG_0E) and the low byte from EE_PHY_L

```
(u8) phy[offset*2+1] = ior(IOaddr, 0x0E); /* high byte of 16-bit word */
(u8) phy[offset*2] = ior(IOaddr, 0x0D);   /* low byte of 16-bit word */
```

### 5.4.2 Write PHY Register

The following steps to write PHY registers:

1. write the PHY register offset into EPAR (REG_0C) bit[4:0] and the PHY address bit[1:0]=01 into EPAR (REG_0C) bit[7:6].
2. write the PHY data high byte into EE_PHY_H (REG_0E) and the low byte into EE_PHY_L (REG_0D).
3. write command=0x0A into EPCR (REG_0B) to start the PHY + WRITE operation:
   i. EPCR (REG_0B) EPOS bit[3]=1: select PHY mode (0: select SROM, see ch.5.3)
   ii. EPCR (REG_0B) ERPRW bit[1]=1: issue WRITE command.
4. read EPCR (REG_0B) and wait until ERRE bit[0]=0 ok, or just following step 5.
5. wait at least 1~200us, then write "8" into EPCR (REG_0B) to clear WRITE command.

For example, 100M Full-duplex mode to write data=0x2100 into BMCR at offset=0x00:

1. write offset=0x0 into EPAR (REG_0C) bit[4:0] and "01" into EPAR (REG_0C) bit[7:6]

```
iow(IOaddr, 0x0C, (offset=00 | 0x40) );
```

2. write the PHY high byte 0x21 into EE_PHY_H and the low byte 0x0 into EE_PHY_L

```
iow(IOaddr, 0x0E, 0x21);
iow(IOaddr, 0x0D, 0x00);
```

3. write the PHY WRITE command=0x0A into EPCR (REG_0B)

```
iow(IOaddr, 0x0B, (0x02 | 0x08) );
```

4. wait until EPCR (REG_0B) ERRE bit[0]=0 ok, or just following step 5

```
while((ready_phy = ior(IOaddr, 0x0B)) & 1) {; };    /* detect PHY WRITE ready */
```

5. delay at least 1~200us, then write "8" into EPCR (REG_0B) to clear it and keep PHY

```
udelay(20);  iow(IOaddr, 0x0B, 0x08);
```

## 5.5 How to Transmit Packets

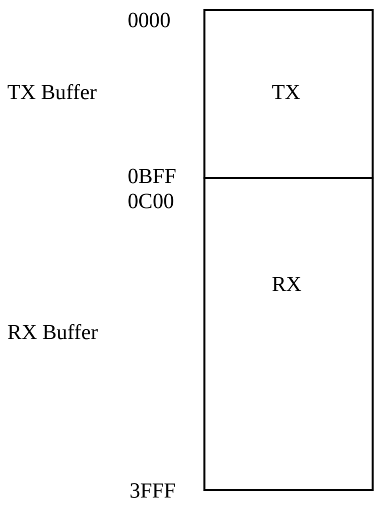


**Figure 5.1 Packet Transmitting Buffer**

Before transmitting a packet, the data of the packet must save into the TX FIFO SRAM, which is the internal SRAM address 0 ~ 0xBFF in the DM9000 MAC and use the output port byte command to select MWCMD (REG_F8), the memory data write command with address increment register. The length of the packet is defined as the high byte put in MDRAH (REG_FD) and the low byte put in MDRAL (REG_FC). The final step is to set the bit[0] TXREQ (Transmit Request) in TCR (REG_02) for transmitting the packet automatically.

The DM9000 will generate an interrupt latched at PTS bit[1]=1 of ISR (REG_FE), if setting PTM bit[1]=1 of IMR (REG_FF), and also to set a completion flag latched at NSR (REG_01) either TX1END bit[2]=1 or TX2END bit[3]=1 in toggle to indicate that the packet is transmitted. The DM9000 supports byte/ word/ dword memory data command to quickly write the packet's data into the TX FIFO SRAM, dependent on the system hardware application.

### 5.5.1 Transmit a Packet

Step 1: check the memory data 8/ 16/ 32-bit width.

```
(u8) io_mode = ior( IOaddr, 0xFE ) >> 6; /* ISR bit[7:6] IOMODE keep DATA I/O mode*/
```

Step 2: write the packet's data into TX FIFO SRAM, depended on byte/ word/ dword mode.

```
outb( IOaddr, 0xF8 );            /* trigger MWCMD (REG_F8) with write_ptr++ */
/* u8 TX_data[]: the transmitting data, int TX_length: the length of TX_data[] */
```

```
if ( io_mode == DM9000_BYTE_MODE ) { /* I/O 8-bit byte mode if ( io_mode == 2 ) */
for ( i = 0; i < TX_length; i++ )  /* loop to write a byte data into TX FIFO SRAM */
outb( TX_data[i], IOaddr + 4 );                  }

else if ( io_mode == DM9000_WORD_MODE ) { /* I/O 16-bit word mode if(io_mode== 0) */
(int) length_tmp = ( TX_length + 1 ) / 2;
for ( i = 0; i < length_tmp; i++ ) /* loop to write a word data into TX FIFO SRAM */
outw( ( (u16 *) TX_data)[i], IOaddr + 4 );        }

else if ( io_mode == DM9000_DWORD_MODE ) { /* I/O 32-bit dword mode if(io_mode==1)*/
(int) length_tmp = ( TX_length + 3 ) / 4;
for ( i = 0; i < length_tmp; i++ ) /* loop to write a dword data into TX FIFO SRAM*/
/* outl() is output long word to 32-bit I/O port */
outl( ( (u32 *) TX_data)[i], IOaddr + 4 );        }
```

Step 3: write the transmitting length into MDRAL (REG_FC) & MDRAH (REG_FD)

```
/* write high byte of the TX data length into MDRAH REG_FD */
iow( IOaddr, 0xFD, (TX_length >> 8) & 0xff );
/* write low byte of the TX data length into MDRAL REG_FC */
iow( IOaddr, 0xFC, TX_length & 0xff );
```

Step 4: start to transmit a packet

```
iow( IOaddr, TCR, 1 );   /* set a TX request command, TXREQ, bit[0] of TCR REG_02 */
```

### 5.5.2 Check a Completion Flag

If the driver is used the polling/ interrupt method, the program segment can be inserted into the TX routine for detecting a packet transmission completed or even double-check interrupt:

```
(u8) TX_status = ior( IOaddr, 0x01 );   /* read NSR (REG_01) for TX completed */
if ( TX_status & (4 | 8) ) { /* success */ }; /* check if TX1END or TX2END=1, TX ok*/
```

The program segment shown as follows can be inserted into the interrupt handler, if the driver is used the interrupt driven and setting the IMR PTM bit[1]=1 enable:

```
(u8) INT_status = ior( IOaddr, 0xFE );  /* got DM9000 interrupt status in ISR */
if ( INT_status & 0x02 ) { /* success */ };  /* check if PTS bit[1]=1, TX ok */
iow( IOaddr, 0xFE, 0x02 );              /* clear PTS bit[1] latched in ISR */
```

## 5.6 How to Receive Packets

The received and filtered packet's data save in the RX FIFO, which is the internal SRAM address 0x0C00 ~ 0x3FFF (13K-byte) in the DM9000 MAC. There are four bytes for the MAC header of each packet saving in the RX FIFO SRAM, and using the MRCMDX (REG_F0) & MRCMD (REG_F2) registers to get the information of the packet incoming.

The first byte is used to check whether a packet is received and filtered in the RX SRAM. If this byte is "01", it means there is a packet received. If this byte is "00", it means there is no packet received in the RX SRAM. Before reading the other bytes, make sure the first byte of the MAC header is "01". But, if neither "01" nor "00", the DM9000 MAC/ PHY must do a software-reset in order to recover from the un-stable states between the system bus and the DM9000 LAN chip. The second word saves the "status" information of the received packet. The format of the status high byte is the same as RSR (REG_06). Please refer to the datasheet ch.6.7. According to this format, the received packet can be verified as a correct packet or an error packet. The third and fourth bytes are the "length" of the received packet. The others bytes are the received packet's data, or named the RX payload. The following diagram is shown the format of the received package frame:

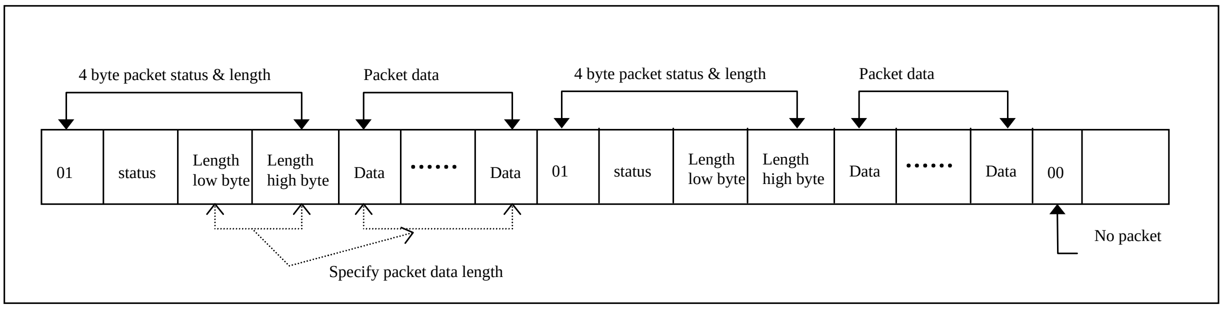


**Figure 5.2 Block Diagram of the Received Packets**

### 5.6.1 Receive Interrupt Service Routine

The following program segment can be inserted to the interrupt handler if the driver is designed as the interrupt driven or the polling method for waiting packets received ready:

```
u8 INT_status = ior( IOaddr, 0xFE );    /* got DM9000 interrupt status in ISR */
if ( INT_status & 0x01 ) { /* doing ch.5.6.2 */ }; /* check if PRS bit[0]=1, RX ok*/
iow( IOaddr, 0xFE, 0x01 );         /* clear PRS latched in ISR bit[0] (see ch.5.5.2)*/
```

### 5.6.2 Receive a Packet

The definition of MRCMDX (REG_F0) is the memory data read command without address increment register. MRCMDX is only used to read the received packet ready flag "01" from the RX FIFO SRAM.

Here is an example to get RX ready:

```
(u8) RX_ready = ior( IOaddr, 0xF0 );    /* dummy READ the packet RX ready flag */
RX_ready = (u8) inb( IOaddr + 4 ); /* got the most updated data */
if ( RX_ready == 0x01 ) {     /* RX ready check: this byte must be "01" or "00" */
/* check the RX status & length (see ch.5.6.3) and income packets (see ch.5.6.4) */
} else if ( RX_ready != 0x00 ) {   /* stop interface and wait to reset device*/
iow( IOaddr, 0xFF, 0x80 );         /* stop INT request */
iow( IOaddr, 0xFE, 0x0F );         /* clear ISR status */
iow( IOaddr, 0x05, 0x00 );         /* stop RX function */
(u8) device_wait_reset = TRUE;     /* raise the MAC/ PHY software-reset flag */
/* iow( IOaddr, 0x00, 1 );         // it’s quick software-reset to replace above
udelay( IOaddr, 20 );
iow( IOaddr, 0x00, NCR_set );
iow( IOaddr, 0xFF, 0x80 );
iow( IOaddr, 5, RCR_set | 1 );     */
/* then, re-new system variables and counters dropped… */
}
```

### 5.6.3 Check the Packet Status and Length

MRCMD (REG_F2): memory data read command with the increment of the RX read pointer. The read pointer will be increased after reading the memory read command MRCMD (REG_F2). The size, the increment of the RX read pointer, is depended on the system application, which the I/O bus width is byte/ word/ dword to increase 1/ 2/ 4 bytes respectively. MRCMD (REG_F2) is only used to read the RX status, length and the packet's data from the RX SRAM.

Here is an example to get the RX status and length:

```
(u8) io_mode = ior( IOaddr, 0xFE ) >> 6; /* ISR bit[7:6] IOMODE keep DATA I/O mode*/
outb( 0xF2, IOaddr );              /* trigger MRCMD (REG_F2) with read_ptr++ */
```

```
/* int RX_status: the RX packet status, int RX_length: the length of the RX packet*/

if ( io_mode == DM9000_BYTE_MODE ) { /* I/O 8-bit byte mode if ( io_mode == 2 ) */
RX_status = inb( IOaddr + 4 ) + ( inb( IOaddr + 4 ) << 8 );
RX_length = inb( IOaddr + 4 ) + ( inb( IOaddr + 4 ) << 8 ); }

else if ( io_mode == DM9000_WORD_MODE ) { /* I/O 16-bit word mode if(io_mode== 0) */
RX_status = inw( IOaddr + 4 ) & 0xff00;      /* got the high byte of the status */
RX_length = inw( IOaddr + 4 );          }

else if ( io_mode == DM9000_DWORD_MODE ) { /* I/O 32-bit dword mode if(io_mode==1)*/
(u32) status_tmp = inl( IOaddr + 4 );        /* got the RX 32-bit dword data */
RX_status = (u16)( status_tmp & 0xff00 );    /* got the high byte of the status */
RX_length = (u16)( ( status_tmp >> 16 ) & 0xffff );    }
```

### 5.6.4 Receive the Packet's Data

The read pointer will be increased after reading the memory read command MRCMD REGF2. According to the length of the received packet, dump out the RX payload and the 4-byte CRC.

Here is an example to get the RX packet's data:

```
/* u8 RX_data[] : the data of the received packet with the 4-byte CRC checksum */
if ( io_mode == DM9000_BYTE_MODE ) { /* I/O 8-bit byte mode if ( io_mode == 2 ) */
for ( i = 0 ; i < RX_length ; i++ ) /* loop to READ a byte data from RX FIFO SRAM */
 RX_data[ i ] = (u8) inb( IOaddr + 4 );      }

else if ( io_mode == DM9000_WORD_MODE ) { /* I/O 16-bit word mode if(io_mode== 0) */
int length_tmp = ( RX_length + 1 ) / 2;
for ( i = 0 ; i < length_tmp ; i++ ) /* loop to READ a word data from RX FIFO SRAM*/
 ( (u16 *)RX_data)[ i ] = inw( IOaddr + 4 );      }

else if ( io_mode == DM9000_DWORD_MODE ) { /* I/O 32-bit dword mode if(io_mode==1)*/
int length_tmp = ( RX_length + 3 ) / 4;
for ( i = 0 ; i < length_tmp ; i++ )/* loop to READ a dword data from RX FIFO SRAM*/
 ( (u32 *)RX_data)[ i ] = inl( IOaddr + 4 ); } /* inl() is input long word from 32-bit
I/O port */
```

# 6 Diagnostic

This chapter is the simply debug procedure for hardware engineers to check the embedded system with the DM9000 LAN chip is workable.

## 6.1 MAC Address

Make sure MAC address, the 6-byte Ethernet individual address, is stored into the EEPROM 93C46 before using the Ethernet card.

For example:

MAC address 00:01:02:03:04:05

Bellow MAC addresses are for special purpose, don't use it.

FF FF FF FF FF FF : When all bits are set to "1", it is for broadcast address.

01 XX XX XX XX XX : When the bit[0] of the first byte is set to 1, it is for multicast address.

## 6.2 DM9000 Register Read / Write Test

The DM9000 registers read/ write by the processor is a good way to check if your system bus to the DM9000 MAC is workable. Make sure your driver have the right I/O base address mapped to the DM9000 LAN chip. Try to write a value to some write-able registers of the DM9000 and read it back to verify it.
EX: To write MAC address into the DM9000 registers from PAB0 (REG_10) to PAB5 (REG_15), then read it back in order to verify it.

## 6.3 EEPROM 93C46 Word Data Test

The DM9000 would read/ write the word data from the EEPROM 93C46. And, double check the EEPROM word 0 ~ 2 MAC address is the same as that in the PAR REG_10 ~ REG_15.

## 6.4 Ethernet Link

To check the DM9000 was linked correctly with another Ethernet, there are two ways for you to verify it. First, the link LED is light. Second, you can read the network status register (NSR REG_01) in the DM9000 to check if the status LINKST bit[6] is equal to "1".

For example, linking with 100Mbps Ethernet without RX/ TX data, the value of NSR (REG_01) will be 0x40. Please make sure the liking cable is correct.

EX. Between two Ethernet cards, you need a crossover cable, and if the DM9000 links to a PC via a switch or a hub, you will need a straight cable.

### 6.5 External Loopback Test

Using a crossover cable plugged to RJ-45, you will receive the packets what you send.

### 6.6 Transmitting Packets

If your pass above tests, you can try to send packets to another Ethernet device.

### 6.7 Receiving Packets

Try to receive packets from another Ethernet device.

### 6.8 System Test

Try to ping another IP address from the DM9000 device. If the reply message was correct, the Ethernet device with the DM9000 LAN chip is workable now.

# 7 The Others.

## 7.1 The priority of the EEPROM, IO strap pins, and the default setting.

The default setting of DM9000 can be changed by the IO strap pins or EEPROM with higher priority at EEPROM setting. For example, the default IO based address of DM9000 is 0x300 (i.e. SA4~9 are set to 0x300). If the strap pin TXD0 is pulled high and the strap pins TXD1 and 2 are default setting, the IO based address set by the IO strap pins is 0x0310. And if the bit[5:4] of word 3 is "01", and bit[11:9] of word 6 are "111" for EERPOM setting, the IO based address of EEPROM setting is 0x0370. These settings are different but based on the priority rule; the DM9000 chip will be operated in the IO based address 0x0370 by the EEPROM setting.

## 7.2 Identify how many DM9000 on the system.

In an open system, the following program segment can identify how many DM9000 are used on the system.

```
for (IOaddr=0x0300 , IOdata = 0x0304; IOaddr<0x380; IOaddr+=0x10 , IOdata+= 0x10)
  {
   if((ior(0x28) == 0x46) && (ior(0x29) == 0x0A)
     if((ior(0x2A) == 0x00) && (ior(0x2B) == 0x90)
       {
           printf(" \n DM9000 is found and the location is %04x", IOaddr);
           result ++;     /* success */
       }
 }
return result;
```

Because the VID and PID might be changed by EEPROM setting, there is another way for your reference in the following.

```
for (IOaddr=0x0300 , IOdata = 0x0304; IOaddr<0x380; IOaddr+=0x10 , IOdata+= 0x10)
  {
   if((ior(0x08) == 0x37) && (ior(0x09) == 0x38)
     {
           printf(" \n DM9000 is found and the location is %04x", IOaddr);
           result ++;     /* success */
     }

 }
return result;
```

### 7.3 How to transmit and receive more than 2048-byte packets

If a system transmits or receives more than 2048-byte packets which size is over spec., the TCR bit[6] and RCR bit[6] should be set to "1".

### 7.4 The performance of DM9000

The performance of the DM9000 is depend-on the capability of MPU/ MCU. If the MPU/MCU is so fastest to match the DM9000 IOR#/IOW# maximum speed timing, the performance will be: 22ns + 84ns = 106ns ➔ 9433962bps ➔ 9.21Mbps.

| | | | | | |
|---|---|---|---|---|---|
| 8bit mode | Receive or transmit data | ➔ | 8*9.21Mbps | ➔ | 73.68Mbps |
| | Rec. and trans. data at the same time | ➔ | 8*9.21Mbps/2 | ➔ | 36.84Mbps |
| 16bit mode | Receive or transmit data | ➔ | 16*9.21Mbps | ➔ | 147.36 Mbps |
| | Rec. and trans. data at the same time | ➔ | 16*9.21Mbps/2 | ➔ | 73.68 Mbps |
| 32bit mode | Receive or transmit data | ➔ | 32*9.21Mbps | ➔ | 294.72 Mbps |
| | Rec. and trans. data at the same time | ➔ | 32*9.21Mbps/2 | ➔ | 147.36 Mbps |

Please note the DM9000 is 10/100Mbps Ethernet NIC, so the maximum speed is 100Mbps.

### 7.5 WOL (Wakeup on LAN)

DM9000 LAN chip also supports the WOL (Wakeup on LAN) function. There are three methods to generate WOL signal, which are the Magic Packet, Link Change, and Sample Frame. This section will discuss how to implement the WOL function in details.

(1) Magic Packet

If DM9000 receives a broadcast packet which content is "FF FF FF FF FF FF"+"Note Address 16 times", then the wakeup pin 79 WAKEUP will be activated. Please note that the Node address must be the same with the Physical address in the PAR REG_10 ~ REG_15.

For example:

```
iow(WCR , ior(WCR) | 0x08) ;
iow(NCR , ior(NCR) | 0x40);
/* DM9000 Node address in PAR REG_10 ~ REG_15 is 00 60 6e 90 00 01 */
/* If DM9000 receive the packets as follows, the WOL mode will be active. */
/* FF FF FF FF FF FF 00 60 6E 90 00 01 00 60 6E 90 00 01 00 60 6E 90 00 01 */
/* 00 60 6E 90 00 01 00 60 6E 90 00 01 00 60 6E 90 00 01 00 60 6E 90 00 01 */
/* 00 60 6E 90 00 01 00 60 6E 90 00 01 00 60 6E 90 00 01 00 60 6E 90 00 01 */
/* 00 60 6E 90 00 01 00 60 6E 90 00 01 00 60 6E 90 00 01 00 60 6E 90 00 01 */
/* 00 60 6E 90 00 01                                                        */
```

(2) Link Change

No matter the internal PHY or external MII of DM9000 is used, if the link status has been changed, the wakeup pin will be active.

For example,

```
iow(WCR , ior(WCR) | 0x20) ;

iow(NCR , ior(NCR) | 0x40);
```

(3) Sample Frame

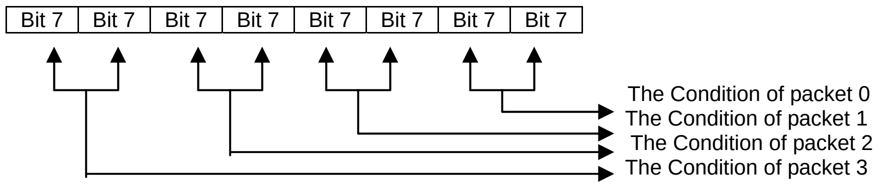
The Sample Frame would be made up of 6 packets and the max size of each packet can be 2048-byte. The wakeup pin will be active if DM9000 receives only one set of the sample frame packet. The following description will show how to set the Sample Frame in details.

- First close the DM9000 receives function and set the bit[0] of RCR (REG_05) to "0".
- Stop the self recover function. Set the bit[7] PAR of IMR (REG_FF) to "0".
- Save the 6 packets max into the DM9000 FIFO SRAM. The format is shown below.

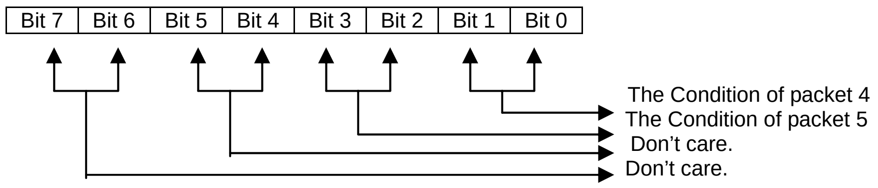
| m-byte3 / packet 3 | m-byte2 / packet 2 | m-byte1 / packet 1 | m-byte0 / packet 0 | DM900 Memory location |
|---|---|---|---|---|
| packet 3-byte 0 | packet 2-byte 0 | packet 1-byte 0 | packet 0-byte 0 | 0000h |
| packet 3-byte 1 | packet 2-byte 1 | packet 1-byte 1 | packet 0-byte 1 | 0004h |
| : | : | : | : | 0008h |
| : | : | : | : | 000Ch |
| : | : | : | : | 0010h |
| : | : | : | : | : |
| packet 3-byte 2047 | packet 2-byte 2047 | packet 1-byte 2047 | packet 0-byte 2047 | 1FFCh |

| m-byte3 / packet 5 | m-byte2 / packet 4 | m-byte1 / Condition 1 | m-byte0 / Condition 0 | |
|---|---|---|---|---|
| packet 5-byte 0 | packet 4-byte 0 | Cond. 1-byte 0 | Cond. 0-byte 0 | 2000h |
| packet 5-byte 1 | packet 4-byte 1 | Cond. 1-byte 1 | Cond. 0-byte 1 | 2004h |
| : | : | : | : | 2008h |
| : | : | : | : | 200Ch |
| : | : | : | : | 2010h |
| : | : | : | : | : |
| packet 4-byte 2047 | packet 4-byte 2047 | Cond. 1-byte 2047 | Cond. 0-byte 2047 | 3FFCh |

The format of Condition 0


Bit 7
Bit 7
Bit 7
Bit 7
Bit 7
Bit 7
Bit 7
Bit 7
The Condition of packet 0
The Condition of packet 1
The Condition of packet 2
The Condition of packet 3

The format of Condition 1



If the condition is "00" or "01", don't care the byte; if "01", check the byte is matched or not; if "11", stop checking.

- Restart the Sample Frame function by setting the bit[4] SAMPLEEN of WCR (REG_0F) to "1" and enable the WOL function by setting the WAKEEN bit[6]=1 in NCR (REG_00).
- Restart the DM9000 receiver function by setting the bit[0] of RCR (REG_05) to "1".

For example,

```
    iow(RCR , 0x00);

    iow(IMR , ior(IMR) & 0x0f);

    /* Here the Sample Frame put in the DM9000 FIFO SRAM 0x0000~0x3FFF */

    iow(WCR , ior(WCR) | 0x10);

    iow(NCR , ior(NCR) | 0x40);

    iow(RCR , REG_05 | 1);
```