DM9000A

16 / 8 Bit Ethernet Controller with General Processor Interface

Application Notes V1.20

Technical Reference Manual

Davicom Semiconductor, Inc

1 Introduction

1.1 General Description

The DM9000A is a fully integrated, powerful and cost-effective Fast Ethernet MAC controller with a general processor interface, an EEPROM interface, a 10/100 PHY and 16K Byte SRAM (13K Byte for RX FIFO and 3K Byte for TX FIFO). It is designed with low power, single-voltage and high performance process that supports 3.3V with 5V tolerant I/O. Besides, the DM9000A supports 8/ 16-bit processor interface to internal memory accesses for many different processors. It is good integrated 10/100 Mbps transceiver with AUTO-MDIX and IP/TCP/UDP-Checksum Offload.

The goal of this document is for the embedded design engineers, to implement the DM9000A LAN chip on any processor's architecture quickly and successfully, with providing the exact reference information and pertaining to many embedded systems. The software programming is very simple, so users can port the software drivers to any system easily.

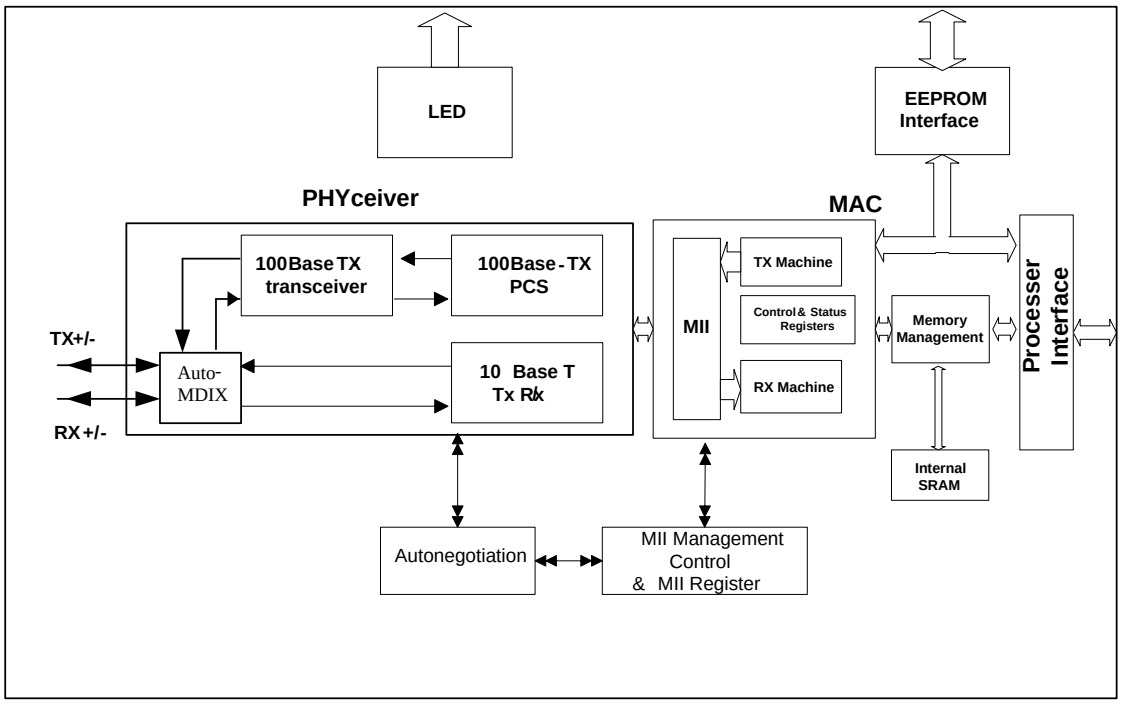


Figure 1.1 DM9000A Internal Block Diagram.

2 General Processor Bus Description

This chapter is intended to aid design engineers connecting the DM9000A device to a micro-processor or micro-controller. The discussion will include the pin functional table, and the individual control signals of the DM9000A involved in the connection between the device and an associated micro-processor/ micro-controller in detail.

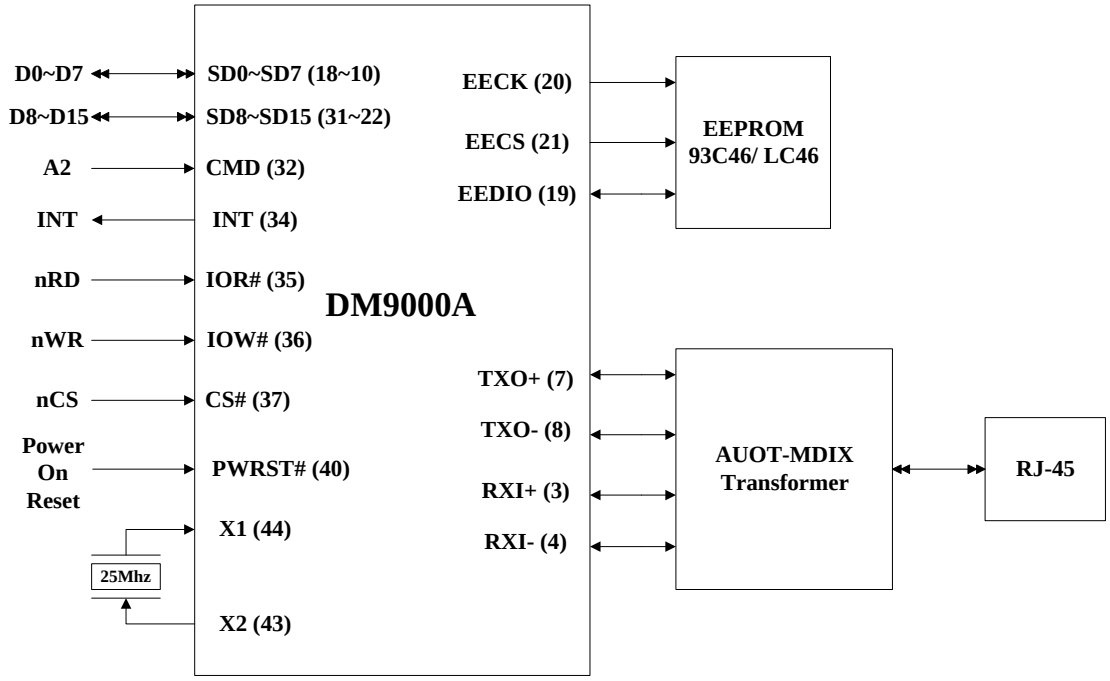


Figure 2.1 Signal Connection with a Processor Interfacing.

2.1 Signals Connection with Processor Bus

The DM9000A can interface directly with most general processors local bus such as ARM, MIPS, Intel, TI, Motorola, NEC, and Hitachi... It is designed to suit for the implementation of Fast Ethernet connectivity solutions to many embedded systems.

2.1.1 Pin Function Table

Note: The pins of processor parallel interface all have a pulled down resistor about 60K Ohm internally.

Processor Bus Signal	DM9000A Signal	Pin No.	I/O	Description
nRD	IOR#	35	I	Processor READ Command This pin is low active at default; its polarity can be modified by the EEPROM setting.
nWR	IOW#	36	I	Processor WRITE Command This pin is low active at default; its polarity can be changed by the EEPROM setting.
nCS / nAEN	CS#	37	I	Chip Select A low active signal is used to select the DM9000A. Its polarity can be changed by the EEPROM setting.
SD0 ~ 7	SD0 ~ 7	18,17,16 14,13,12 ,11,10	I/O	DATA Bus 0 ~ 7
SD8 ~ 15	SD8 ~ 15	31,29,28 27,26,25 ,24,22	I/O	DATA Bus 8 ~ 15 (in 16-bit mode)
CMD	CMD	32	I	Command Type When low, the access of this command cycle is INDEX port When high, the access of this command cycle is DATA port
INT	INT	34	O	Interrupt Request This pin is high active and open-collected at default; its polarity and its output type can be modified by the EEPROM setting.

Table 2.1 Pin Function Table for Processor Interface

2.1.2 8/ 16-Bit Mode Setting

There are two operation modes of DATA bus width, 8-bit or 16-bit, when access to the internal memory in the DM9000A. These two modes are selected by the strap pin 21 EECS shown the following table:

EECS (pin 21)	DATA width
0	16-bit
1	8-bit

Where, "1" means pull-high with the 10K Ohm resistor, and "0" means floating (default).

The status of DATA width operation mode can be examined from Bit [7] of ISR REG. FEH,

ISR (REG. FEH)	IOMODE (DATA width)
Bit [7]	Operation
0	16-bit mode
1	8-bit mode

2.1.3 Command Type

In the DM9000A, there are only two registers, named INDEX port and DATA port, which can be accessed directly. These two ports are distinguished by the CMD pin in the access command cycle. When CMD is low in command cycle, INDEX port is accessed; otherwise, when CMD is high, DATA port is accessed.

All of these control and status registers in the DM9000A are accessed indirectly by the INDEX/ DATA ports. The command sequence to access the specified control/ status register is: firstly, to write the register's address into INDEX port, then read/ write their data through DATA port. The following diagram (Figure 2.2) is an example for the DM9000A CMD pin connected to an embedded system.

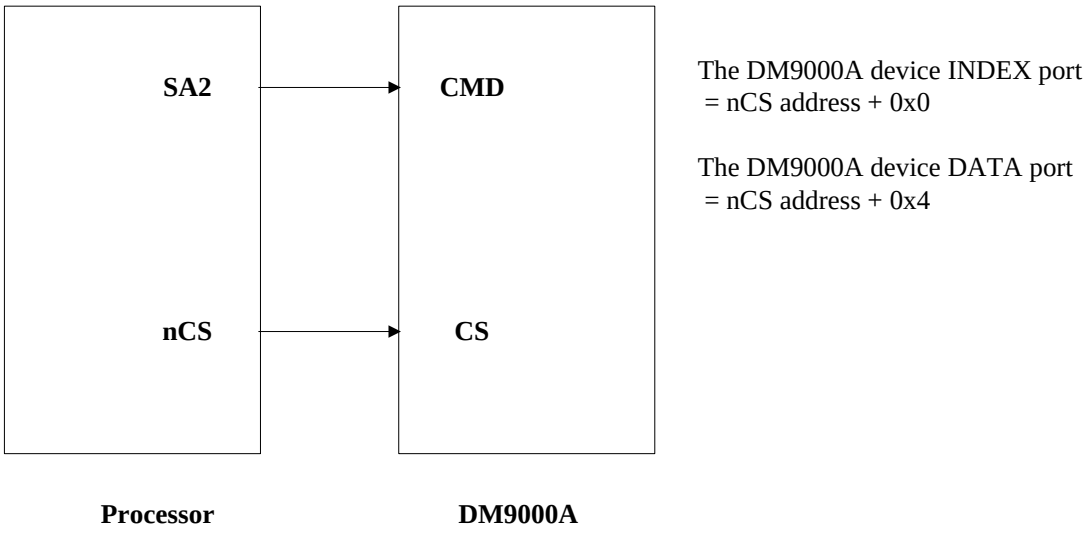


Figure 2.2 CMD Pin and Processor Interface.

3 System Hardware Design

3.1 How to Select Chip

In the command cycle, the DM9000A is accessed by CS pin cooperated with IOR or IOW pins. These pins are low active at default, while their polarity can be modified by EEPROM setting to suit for the application of various processor types.
Both of CS pin and IOW/ IOR pin should be active to write/ read the value into INDEX port or DATA port. So, if the IOW and IOR signals in the system are only used by the DM9000A, the CS pin can be forced to the active logic level to simplify the system design.

3.2 Strap Pins Setting

The DM9000A provides the following strap pins:

Strap pins control list:

Pin	Name	Strap	Description
20	EECK	INT Polarity Type	0: INT active High 1: INT active Low
21	EECS	DATA Bus Width	0: DATA 16-bit mode 1: DATA 8-bit mode
25	GP6	INT Output Type	**available in 8-bit mode only** 0: INT Force-Output mode 1: INT Open-Drain mode

Table 3.1 Strap Pin Control Table

Note: "1" means pull-high with the 10K Ohm resistor, and "0" means floating (default).

3.3 Serial EEPROM Operation

The DM9000A supports a serial EEPROM interface. The EEPROM of the DM9000A holds the following parameters:

1. Ethernet node address.
2. Vendor ID and Product ID auto load.
3. Processor control bus pins polarity setting.
4. Wake-up mode control.
5. PHY ON or PHY OFF while powered up.
6. AUTO-MDIX enable/ disable setting.
7. LED1 & LED2 pins act as IO16 pin and IOWAIT/ WAKE pin in 16-bit mode.
8. LED mode 0 or mode 1 setting.

All of the above mentioned settings are read from the EEPROM upon hardware power-on reset. The specific target device is IC 93C46, the 1024-bit serial EEPROM. (EEPROM is 93C46/ LC46 such as ATmel, Micro-chip, ATC, and CSI.)
All of accesses to the EEPROM are done in words. All of the EEPROM addresses in the specification are specified as word addresses.

The default settings of the DM9000A can be changed by the I/O strap pins, or the EEPROM bits settings with higher priority. The priority for setting the pins polarity is EEPROM > strap pins > default setting.

3.3.1 EEPROM Format

Name	Address (Word)	Programming Value (Hex)	Description
Ethernet Node Address	0 ~ 2	XX XX XX XX XX XX	6-byte Ethernet node address.
Auto Load Control	3	5445	Bit [1:0] = 00: Disable vendor ID and product ID programming. *01: Accept vendor ID and product ID programming. Bit [3:2] = 00: Disable setting of Word 6 Bit [8:0]. *01: Accept setting of Word 6 Bit [8:0]. Bit [5:4]: Reserved = 0.

			Bit [7:6] = 00: Disable setting of Word 7 Bit [3:0]. *01: Accept setting of Word 7 Bit [3:0]. Bit [9:8]: Reserved = 0. Bit [11:10] = 00: Disable setting of Word 7 Bit [7]. *01: Accept setting of Word 7 Bit [7]. Bit [13:12] = 00: Disable setting of Word 7 Bit [8]. *01: Accept setting of Word 7 Bit [8]. Bit [15:14] = 00: Disable setting of Word 7 Bit [15:12]. *01: Accept setting of Word 7 Bit [15:12]. Note: The remark * is now programming value.
Vendor ID	4	0A46	2-byte vendor ID.
Product ID	5	9000	2-byte product ID.
Pin Control	6	01E7	When Word 3 Bit [3:2] = 01, these bits can control the CS, IOR, IOW , INT, IOWAIT and IO16 pins polarity: Bit [0] = 0: Processor CS pin is active high. *1: Processor CS pin is active low. Bit [1] = 0: Processor IOR pin is active high. *1: Processor IOR pin is active low. Bit [2] = 0: Processor IOW pin is active high. *1: Processor IOW pin is active low. Bit [3] = *0: Processor INT pin is active high. 1: Processor INT pin is active low. Bit [4] = *0: Processor INT pin is force output. 1: Processor INT pin is force open-collected. Bit [5] = 0: Processor IOWAIT is active high. *1: Processor IOWAIT is active low. Bit [6] = 0: Processor IOWAIT is force output. *1: Processor IOWAIT is force open-collected. Bit [7] = 0: Processor IO16 is active high. *1: Processor IO16 is active low. Bit [8] = 0: Processor IO16 is force output. *1: Processor IO16 is force open-collected. Bit [15:9]: Reserved = 0.
Wake-up Mode Control	7	4180	Depend on the setting of Word 3 Bit [15:6] to accept auto load control: Bit [0] = *0: WAKE pin is active high. 1: WAKE pin is active low. Bit [1] = *0: WAKE pin is in level mode. 1: WAKE pin is in pulse mode. Bit [2] = *0: Magic packet wake-up event is disabled. 1: Magic packet wake-up event is enabled. Bit [3] = *0: Link_change wake-up event is disabled. 1: Link_change wake-up event is enabled. Bit [6:4]: Reserved = 0. Bit [7] = 0: LED mode 0. *1: LED mode 1. Bit [8] = 0: The internal PHY is disabled after power-on. *1: The internal PHY is enabled after power-on. The GPR REG. 1FH Bit [0] is modified from this Bit [8]. Bit [13:9]: Reserved = 0. Bit [14] = 0: AUTO-MDIX OFF, *1: AUTO-MDIX ON. Bit [15]: Reserved = 0.

Note: The programming value of the EEPROM is only for reference to list 16-bit values, "XX:XX:XX:XX:XX:XX, 5445, 0A46, 9000, 01E7, 4180".

Table 3.2 EEPROM Format in 8-bit mode

Name	Address (Word)	Programming Value (Hex)	Description
Ethernet Node Address	0 ~ 2	XX XX XX XX XX XX	6-byte Ethernet node address.
Auto Load Control	3	5405	Bit [1:0] = 00: Disable vendor ID and product ID programming. *01: Accept vendor ID and product ID programming. Bit [3:2] = 00: Disable setting of Word 6 Bit [8:0]. *01: Accept setting of Word 6 Bit [8:0]. Bit [5:4]: Reserved = 0. Bit [7:6] = 00: Disable setting of Word 7 Bit [3:0]. *01: Accept setting of Word 7 Bit [3:0] and Word 7 Bit [13:12] = 10 to let LED2 act as WAKE in 16-bit mode only. Bit [9:8]: Reserved = 0. Bit [11:10] = 00: Disable setting of Word 7 Bit [7]. *01: Accept setting of Word 7 Bit [7]. Bit [13:12] = 00: Disable setting of Word 7 Bit [8]. *01: Accept setting of Word 7 Bit [8]. Bit [15:14] = 00: Disable setting of Word 7 Bit [15:12]. *01: Accept setting of Word 7 Bit [15:12]. Note: The remark * is now programming value.
Vendor ID	4	0A46	2-byte vendor ID.
Product ID	5	9000	2-byte product ID.
Pin Control	6	01E7	When Word 3 Bit [3:2] = 01, these bits can control the CS, IOR, IOW , INT, IOWAIT and IO16 pins polarity: Bit [0] = 0: Processor CS pin is active high. *1: Processor CS pin is active low. Bit [1] = 0: Processor IOR pin is active high. *1: Processor IOR pin is active low. Bit [2] = 0: Processor IOW pin is active high. *1: Processor IOW pin is active low. Bit [3] = *0: Processor INT pin is active high. 1: Processor INT pin is active low. Bit [4] = *0: Processor INT pin is force output. 1: Processor INT pin is force open-collected. Bit [5] = 0: Processor IOWAIT is active high. *1: Processor IOWAIT is active low. Bit [6] = 0: Processor IOWAIT is force output.

			*1: Processor IOWAIT is force open-collected. Bit [7] = 0: Processor IO16 is active high. *1: Processor IO16 is active low. Bit [8] = 0: Processor IO16 is force output. *1: Processor IO16 is force open-collected. Bit [15:9]: Reserved = 0.
Wake-up Mode Control	7	4180	Depend on the setting of Word 3 Bit[15:6] to accept auto load control: Bit [0] = *0: WAKE pin is active high. 1: WAKE pin is active low. Bit [1] = *0: WAKE pin is in level mode. 1: WAKE pin is in pulse mode. Bit [2] = *0: Magic packet wake-up event is disabled. 1: Magic packet wake-up event is enabled. Bit [3] = *0: Link_change wake-up event is disabled. 1: Link_change wake-up event is enabled. Bit [6:4]: Reserved = 0. Bit [7] = 0: LED mode 0. *1: LED mode 1. Bit [8] = 0: The internal PHY is disabled after power-on. *1: The internal PHY is enabled after power-on. The GPR REG. 1FH Bit [0] is modified from this Bit [8]. Bit [11:9]: Reserved = 0. Bit [13:12] = *00: LED2 act normal. 01: LED2 act as IOWAIT in 16-bit mode only. 10: LED2 act as WAKE in 16-bit mode only. Bit [14] = 0: AUTO-MDIX OFF, *1: AUTO -MDIX ON. Bit [15] = *0: LED1 act normal, 1: LED1 act as IO16 in 16-bit mode only.

Note: List the programming values of the EEPROM in 16-bit mode,

"XX:XX:XX:XX:XX:XX, 5405, 0A46, 9000, 01E7, C180" if LED1 pin 39 act as "IO16".

"XX:XX:XX:XX:XX:XX, 5405, 0A46, 9000, 01E7, 5180" if LED2 pin 38 act as "IOWAIT".

"XX:XX:XX:XX:XX:XX, 5445, 0A46, 9000, 01E7, 6180" if LED2 pin 38 act as "WAKE".

"XX:XX:XX:XX:XX:XX, 5405, 0A46, 9000, 01E7, D180" if LED1 pin 39 act as "IO16" and LED2 pin 38 act as "IOWAIT" pin used.

"XX:XX:XX:XX:XX:XX, 5445, 0A46, 9000, 01E7, E180" if LED1 pin 39 act as "IO16" and LED2 pin 38 act as "WAKE" pin used.

Table 3.3 EEPROM Format in 16-bit mode

3.4 GPIO Pins Setting

If the DM9000A operated in 8-bit mode, there are 6 general purpose pins, GP1~GP6, can be used. Their I/O types are controlled by GPCR REG. 1EH. And their I/O data are presented by GPR REG. 1FH.

The default values of GPCR Bit [3:1] are all "0"s for the GP3 ~ GP1 pins as the input mode respectively. But, the GPCR Bit [6:4] GPC64 are all forced to "1"s only for the GP6 ~ GP4 pins as the output mode.

GPCR (REG. 1EH) GPIO interface control:

Bit	Name	Default	Description
7	RESERVED	0, RO	Reserved.
6:4	GPC64	111, RO	Forced to "1"s only as the output ports of GP6 ~ 4 pin 25, 26, 27.
3:1	GPC31	000, RW	GP3~1 pin 28, 29, 31 set to be the input/ output port: "1": the output port represented "0": the input port represented.
0	RESERVED	1, RO	Reserved.

Table 3.4 General Purpose Control Register (GPCR) Table

GPR (REG. 1FH) GPIO interface control:

Bit	Name	Default	Description
7	RESERVED	0, RO	Reserved.
6:4	GPO64	000, RW	GP6~4 pin 25, 26, 27 are forced to the output ports only. Set to "1": the pin output high, set to "0": the pin output low.

3	GPIO3	0, RW	If GP3 is output port, GPR Bit [3] sets to "1" to enable the pin 28 output high or sets to "0" to enable the pin 28 output low. If GP3 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	GPIO2	0, RW	If GP2 is output port, GPR Bit [2] sets to "1" to enable the pin 29 output high or sets to "0" to enable the pin 29 output low. If GP2 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	GPIO1	0, RW	If GP1 is output port, GPR Bit [1] sets to "1" to enable the pin 31 output high or sets to "0" to enable the pin 31 output low. If GP1 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	PHYPD	1, RW	"1": power down the internal PHY "0": power up the internal PHY.

Table 3.5 General Purpose Register (GPR) Table

3.5 Schematic Reference Design

3.5.1 Application Schematics for 8-Bit and 16-Bit

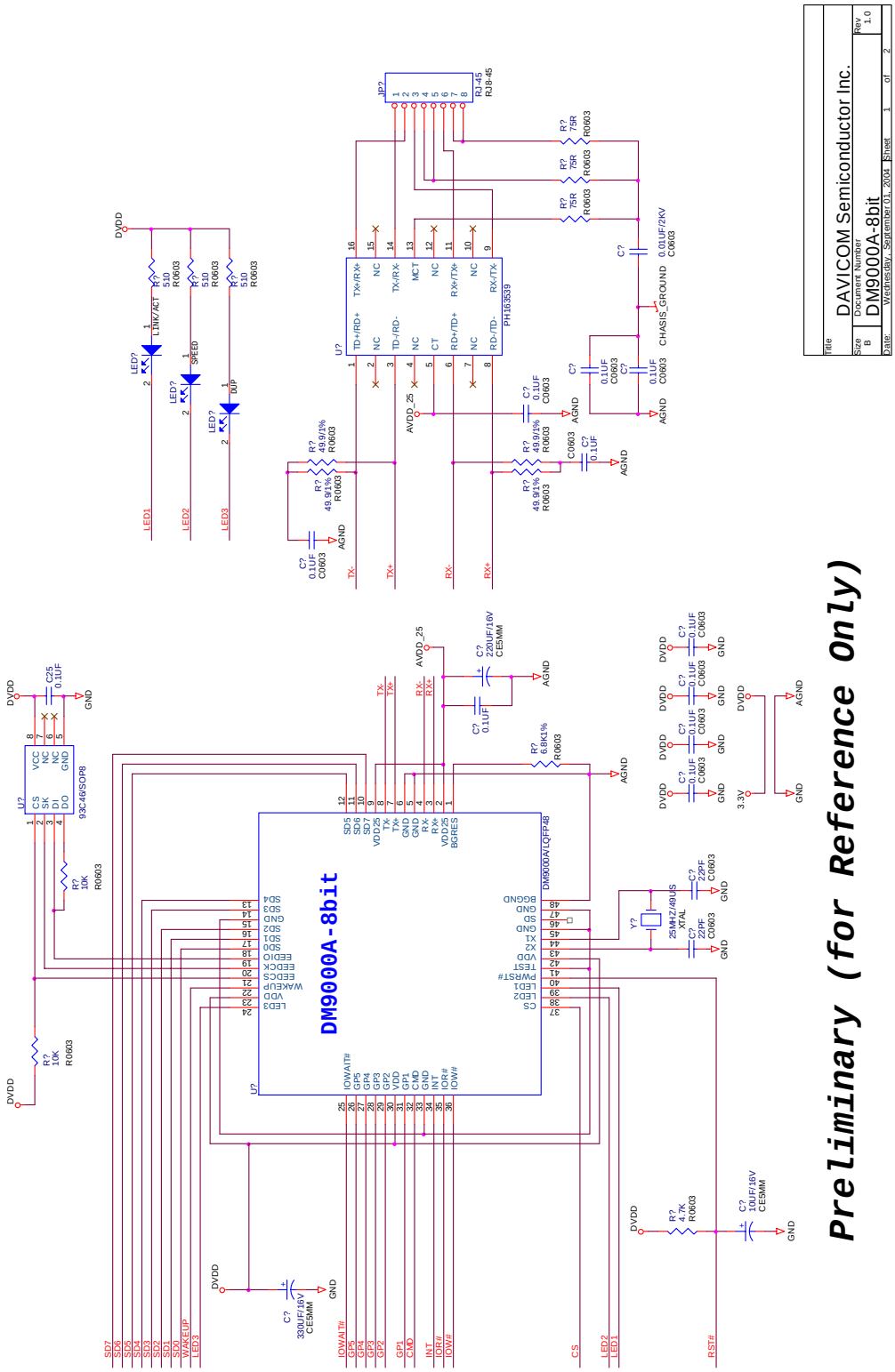


Figure 3.1 Schematic for 8-Bit Processor.

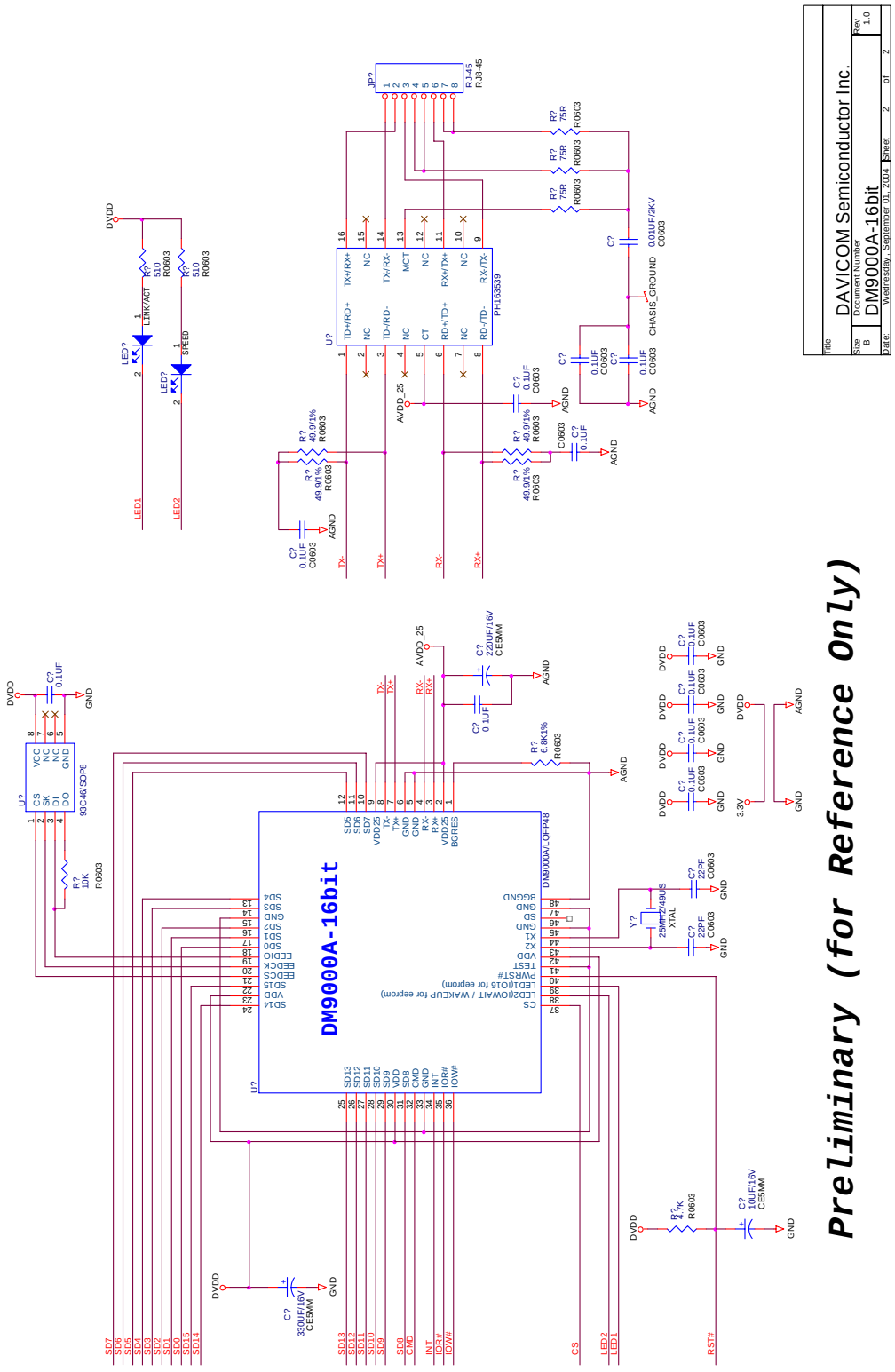


Figure 3.2 Schematic for 16-Bit Processor.

4 Reset Operation and PHY Power-down Mode

The DM9000A can be reset by either hardware reset or software reset. A hardware reset can be accomplished by the power-on reset PWRST# pin 40. A software reset can be accomplished by setting the network control register (NCR REG. 00) RST Bit [0] = 1.

The internal PHY of the DM9000A can be powered down by writing "1" to PHYPD (the Bit [0] of the GPR register), or by setting the Power-down Bit [11] = 1 in the PHY basic mode control register (BMCR PHY REG. 00). In the power-down state, the power consumption is reduced to a minimum under 21mA/ 3.3V.

4.1 Power On Reset

An active low signal is used to reset the DM9000A LAN chip. The PWRST# pin 40 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 DM9000A is ready after 5 us when this pin is de-asserted and then the data will be downloaded from the EEPROM.

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. And the DM9000A needs only 10 us for a software reset.

4.3 PHY Power Down Mode

In the power-down (power-saving) mode, the DM9000A will disable all the TX&RX functions. And, it's almost the same as the PHY SLEEP mode powered down all circuit, except oscillator and clock generator circuit. But, it's different to the Power Reduced mode which the TX circuit still sends out fast link pulse with min. power consumption and wakes up if a valid signal is detected, automatically.

There are two ways to set the power-up/ power-down mode for the internal PHY:

4.3.1 GPR PHYPD Setting

In the MAC registers, the PHYPD bit is used for powering down the internal PHY, and it is default high "1". If the internal PHY is desired to be activated, the system driver needs to clear this power-down bit by writing low "0" to PHYPD in the GPR REG. 1FH.

4.3.2 PHY Register Setting

In the PHY registers, the Bit [11] Power-down of the basic mode control register (BMCR REG. 00) can be set high "1" to enable the PHY power-down mode.

5 How to Program DM9000A

5.1 How to Read/ Write DM9000A Register

There are only two addressing ports through the access of the host interface. One port is INDEX port and the other is DATA port. INDEX port is decoded by the pin CMD=0 and DATA port is decoded by the pin CMD = 1. The content of INDEX port is the address of the register to access DATA port later. Before accessing the 8-bit data in any register or the 8/ 16-bit data in the RX/ TX FIFO SRAM (total 16K bytes), the address of the register must be written into INDEX port. Please refer to the DM9000A datasheet chapter 9.1 about host interface.

Here are the examples to read and write the DM9000A register:

(Where CMD pin is connected to Processor SA2)

```
UINT16 IOaddr;   /* UINT32 IOaddr=0x19000000; for example defined in ARM-base HPI BANK3*/


void iow ( UINT16 reg, UINT8 dataB )
{
   outb(reg, IOaddr);     /* I/O output a byte to INDEX port, select the register*/
   outb(dataB, IOaddr+4); /* I/O output a byte to DATA port, WRITE the register data*/
}


UINT8 ior ( UINT16 reg )
{
   outb (reg, IOaddr);    /* I/O output a byte to INDEX port, select the register*/
   return inb(IOaddr + 4);/*I/O input a byte from DATA port, READ the register data*/
}
```

5.2 Driver Initializing Steps

1. To power on the internal PHY:

iow (0x1F, 0x00); to set GPR (REG. 1FH) Bit [0]: PHYPD = 0

The PHY is powered down at default. The power-up procedure will be needed to enable it by writing iow "0" to PHYPD (GP0 = 0). Please refer to the ch.3.4 about setting the GPIO pins.

2. To do a software reset for the DM9000A initial (see chapter 4.2):

i. iow (0x00, 0x01); to set NCR (REG. 00) RST Bit [0] = 1 for a period time 10 us.

ii. iow (0x00, 0x00); to clear NCR (REG. 00) RST Bit [0] = 0, or let it auto clear.

3. Program the NCR register. Choose normal mode by setting NCR (REG. 00) LBK Bit [2:1] = 00b. The system designer can choose the network operations such as the internal MAC/ PHY loop-back mode, forced the external PHY Full/ half duplex mode or to force collisions, and the wake-up events enable. Please refer to the datasheet chapter 6.1 about the NCR setting.

4. To set the IMR register (REG. FFH) Bit [7] = 1 to enable the Pointer Auto Return function, which is the memory read/ write address pointer of the RX/ TX FIFO SRAM.

5. Read the EEPROM data 3 words, for the individual Ethernet node address (if necessary).

6. Write 6-byte Ethernet node address into the Physical Address Registers (REG. 10H~15H).

7. Write Hash Table 8 bytes into the Multicast Address Registers (REG. 16H ~ REG. 1DH).

8. To clear TX & INTR status by R/W1 the NSR REG. 01 and ISR REG. FEH registers. The Bit [2] TX1END, Bit [3] TX2END, and Bit [5] WAKEST will be cleared automatically by reading it or writing "1" back. Please refer to the datasheet chapter 6.2 & 6.33 about setting NSR & ISR.

9. To handle the NIC interrupts or polling service routines.

10. Program the IMR register (REG. FFH) Bit [0]/ Bit [1] to enable the RX/ TX interrupt.

11. Program the RCR register to enable RX. The RX function is enabled by setting the RXEN Bit [0] = 1 in the RX control register, RCR REG. 05. Please refer to the datasheet ch.6.6 RCR.

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

5.3 How to Read/ Write EEPROM Data

The DM9000A 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. 0BH ~ REG. 0EH), 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] in the EEPROM&PHY address register, EPAR REG. 0CH. Besides, EPAR PHY_ADR Bit [7:6] must be set "00"b to enable the word address for the EEPROM. And, the setting PHY_ADR Bit [7:6] = 01 in EPAR is for the PHY address selected. Please refer to the datasheet ch.6.12 ~ ch.6.14 about setting the EEPROM&PHY registers.

5.3.1 HOWTO Read EEPROM Data

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

Step 1: write the word address into EPAR REG. 0CH.
Step 2: write command = 0x04 into the EEPROM&PHY Control Register (EPCR REG. 0BH) to start the SROM + READ operation:
i. EPCR (REG. 0BH) EPOS Bit [3] = 0: select SROM mode (this is default: 0).
ii. EPCR (REG. 0BH) ERPRR Bit [2] = 1: issue READ command.
Step 3: read EPCR REG. 0BH and wait until ERRE Bit [0] = 0 ok, or just following Step 4.
Step 4: wait 40 us at least, then write "0" into EPCR REG. 0BH to clear READ command.
Step 5: read the SROM data high byte from EE_PHY_H REG. 0EH, and the low byte from EE_PHY_L REG. 0DH in the EEPROM&PHY Data registers.

HOWTO read SROM Word 0x03, Auto Load Control, shown as follows:

1. write word address 0x03 into EPAR REG. 0CH
```
iow ( 0x0C, 0x03 );
```
2. write the SROM + READ command = 0x04 into EPCR REG. 0BH
```
iow ( 0x0B, 0x04 );
```
3. wait until EPCR REG. 0BH Bit [0] = 0 ok, or just following Step 4
```
do { udelay ( 10 ); i++; } while ( ( i < 500 ) && ( inb ( IOaddr + 4 ) & 1 ) );
```
4. delay 40 us at least, then write "0" into EPCR REG. 0BH

```
udelay ( 40 );
iow ( 0x0B, 0 );
```

5. EE_PHY_H REG. 0EH for the high-byte of the SROM data, and

```
(UINT8) e2prom[i*2+1] = ior ( 0x0E );
```

EE_PHY_L REG. 0DH for the low-byte

```
(UINT8) e2prom[i*2] = ior( 0x0D );
```

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

```
for ( i = 0; i < 3; i++ ) {            /* read Ethernet MAC address from SROM */
     iow ( 0x0C, i );
     iow ( 0x0B, 0x04 );
     udelay ( 40 );
     iow ( 0x0B, 0 );
     /* read the SROM word data from EPDR to device address */
     (UINT8) dev_addr[i*2+1] = ior ( 0x0E );/* the high byte of this 16-bit word */
     (UINT8) dev_addr[i*2] = ior ( 0x0D );   /* the low byte of this 16-bit word */
}
```

5.3.2 HOWTO Write EEPROM Data

The following steps are to write data into the SROM:

Step 1: write the word address into EPAR REG. 0CH.

Step 2: write the data high byte into EE_PHY_H REG. 0EH, and the low byte to EE_PHY_L.

Step 3: write command= 0x12 into EPCR REG. 0BH to start the SROM + WRITE operation:

i. EPCR REG. 0BH WEP Bit [4] = 1: enable SROM WRITE operation.

ii. EPCR REG. 0BH EPOS Bit [3] = 0: select SROM mode (this is default: 0).

iii. EPCR (REG. 0BH) ERPRW Bit [1] = 1: issue WRITE command.

Step 4: read EPCR REG. 0BH and wait until ERRE Bit [0] = 0 ok, or just following Step 5.

Step 5: wait 500 us at least, then, write "0" into EPCR REG. 0BH to clear WRITE command.

HOWTO write 0x1E7 into SROM Word 0x06, Pin Control, shown as follows:

1. write word address = 0x06 to EPAR REG. 0CH

```
iow ( 0x0C, 0x06 );
```

2. write the high-byte 0x01 into EE_PHY_H, and the low-byte 0xE7 into EE_PHY_L

```
iow ( 0x0E, 0x01 );
iow ( 0x0D, 0xE7 );
```

3. write the SROM + WRITE command = 0x12 into EPCR REG. 0BH

```
iow ( 0x0B, ( 0x02 | 0x10 ) );
```

4. wait until EPCR (REG. 0BH) ERRE Bit [0] = 0 ok, or just following Step 5

```
do { udelay ( 10 ); i++; } while ( ( i < 500 ) && ( inb ( IOaddr + 4 ) & 1 ) );
```

5. delay 500 us at least, then write "0" into EPCR REG. 0BH

```
udelay ( 60000 );
iow ( 0x0B, 0 );
```

For example, to write Ethernet node address 00:E0:63:0E:56:78 into SROM Word 0& 1& 2:

```
srom_write ( 0x00, 0xE000 );           /* Word 0: low-byte = 00, high-byte = E0 */
srom_write ( 0x01, 0x0E63 );           /* Word 1: low-byte = 63, high-byte = 0E */
srom_write ( 0x02, 0x7856 );           /* Word 2: low-byte = 56, high-byte = 78 */

void srom_write ( int offset, UINT16 dataW )
{
     UINT16    i, tmpv;
     /* write the SROM word address into EPAR REG. 0CH */
     iow ( 0x0C, offset );

     /* write the high-byte to EE_PHY_H REG. 0EH, low-byte to EE_PHY_L REG. 0DH */
     iow ( 0x0D, dataW & 0xff );       /* the low-byte  of this 16-bit word */
     iow ( 0x0E, (dataW >> 8) & 0xff ); /* the high-byte of this 16-bit word */

     /* issue the SROM + WRITE command = 0x12 into EPCR REG. 0BH */
     iow ( 0x0B, 0x02 | 0x10 );

     /* wait >500 us for SROM + WRITE completed then write "0" to clear command */
     do { udelay (10); tmpv= inb (IOaddr+4); i++; } while ( (i < 500) && (tmpv & 1) );
     udelay ( 60000 );
     iow( 0x0B, 0 );
}
```

5.4 How to Read/ Write PHY Register

The DM9000A PHY supports only 32 registers, which are mapped to EPAR (REG. 0CH) Bit [4:0]. The default value of EPAR (REG. 0CH) PHY_ADR Bit [7:6] is "01" to select the PHY mode. Please refer to the datasheet ch.6.12~6.14 about the EEPROM&PHY registers setting.

5.4.1 HOWTO Read PHY Register

The following steps are shown to read data from the PHY register:

Step 1: write the PHY register offset into EPAR REG. 0CH Bit [4:0], and the PHY address "01"b into EPAR REG. 0CH Bit [7:6].

Step 2: write command = 0x0C into EPCR REG. 0BH to start the PHY + READ operation:

i. EPCR (REG. 0BH) EPOS Bit [3] = 1: select PHY mode (0: select SROM, see ch.5.3)

ii. EPCR (REG. 0BH) ERPRR Bit [2] = 1: issue READ command.

Step 3: read EPCR REG. 0BH and wait until ERRE Bit [0] = 0 ok, or just following Step 4.

Step 4: wait 5 us maximum, then write "0x08" into EPCR REG. 0BH to clear READ command.

Step 5: read the PHY data high byte from EE_PHY_H REG. 0EH, and the low byte from EE_PHY_L REG. 0DH in the EEPROM&PHY Data registers.

For example, to read the PHY register of BMCR at address offset = 0x00:

1. write offset = 0 into EPAR REG. 0CH Bit [4:0], and "01"b into EPAR REG. 0CH Bit[7:6]

```
iow ( 0x0C, ( 0x00 | 0x40 ) );/* issue PHY address+ 40H (EPAR PHY_ADR = "01"b) */
```

2. write the PHY + READ command = 0x0C into EPCR REG. 0BH

```
iow ( 0x0B, ( 0x04 | 0x08 ) );
```

3. wait until EPCR REG. 0B ERRE Bit [0] = 0 ok, or just following Step 4

```
do { udelay ( 1 ); i++; }  while ( ( i < 50 ) && ( inb ( IOaddr + 4 ) & 1 ) );
```

4. delay 5 us maximum, then write "0x8" into EPCR REG. 0BH to clear it and keep PHY

```
udelay ( 5 );              /* wait 1~5 us for the PHY+ READ command completion */
iow ( 0x0B, 0x08 );                        /* clear this PHY "READ" command */
```

5. read the high byte from EE_PHY_H REG. 0EH, and the low byte from EE_PHY_L

```
(UINT8) phy[offset*2+1] = ior ( 0x0E );    /* the high byte of this 16-bit word */
(UINT8) phy[offset*2] = ior ( 0x0D );      /* the low byte of this 16-bit word */
```

5.4.2 HOWTO Write PHY Register

The following steps are to write data into the PHY register:

Step 1: write the PHY register offset into EPAR REG. 0CH Bit [4:0], and the PHY address Bit [1:0] = "01"b into EPAR REG. 0CH Bit [7:6].

Step 2: write the PHY high byte data to EE_PHY_H REG. 0EH, and low byte to EE_PHY_L.

Step 3: write command = 0x0A into EPCR REG. 0BH 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.

Step 4: read EPCR REG. 0BH and wait until ERRE Bit [0] = 0 ok, or just following Step 5.

Step 5: wait 5 us maximum, then, write "0x8" into EPCR REG. 0BH to clear WRITE command.

For example, to write data = 0x101 into ANAR REG. 04 for the 100M Full-duplex support:

1. write offset= 4 into EPAR REG. 0CH Bit [4:0], and "01" into EPAR REG. 0CH Bit [7:6]

```
iow ( 0x0C, ( 0x04 | 0x40 ) );/* issue PHY address+ 40H (EPAR PHY_ADR = "01"b) */
```

2. write the PHY high byte 0x01 into EE_PHY_H, and low byte 0x01 into EE_PHY_L

```
iow ( 0x0E, 0x01 );
iow ( 0x0D, 0x01 );
```

3. write the PHY + WRITE command = 0x0A into EPCR REG. 0BH

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

4. wait until EPCR (REG. 0BH) ERRE Bit [0] = 0 ok, or just following Step 5

```
do { udelay ( 1 ); i++; }  while ( ( i < 50 ) && ( inb ( IOaddr + 4 ) & 1 ) );
```

5. delay 5 us maximum, then write "8" into EPCR REG. 0BH to clear it and keep PHY

```
udelay ( 5 );              /* wait 1~5 us for the PHY+WRITE command completion */
iow ( 0x0B, 0x08 );                        /* clear this PHY "WRITE" command */
```

5.5 How to Transmit Packets

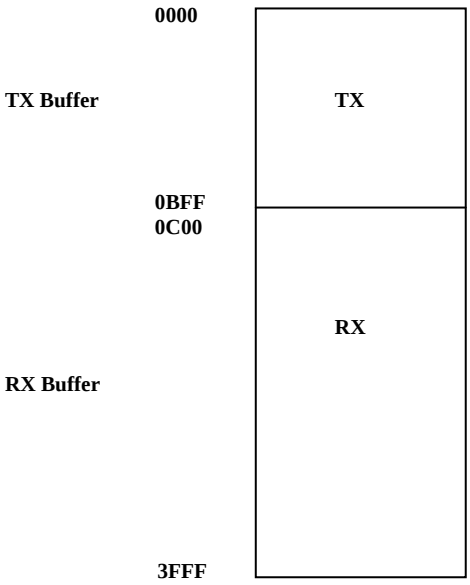


Figure 5.1 Packet Transmitting Buffer.

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 DM9000A MAC and to write F8H (MWCMD REG. F8H port latched) into INDEX port firstly. Then, the length of the packet is put in MDRAH REG. FDH for the high byte and MDRAL REG. FCH for the low byte. The final step is to set the TXREQ (Transmit Request), Bit [0] in TCR REG. 02, for transmitting this packet.

The DM9000A will generate an interrupt at PTS Bit [1] = 1 in ISR REG. FEH, if setting Bit [1] = 1 in IMR REG. FFH, and also to set a completion flag to either TX1END Bit [2] = 1 or TX2END Bit[3] = 1 in NSR REG. 01 in toggle to indicate that the packet is transmitted completely.

5.5.1 Packet Transmission

Step 1: check the memory 8/ 16 DATA width,

```
(u8) io_mode = ior ( 0xFE ) >> 7;  /* ISR Bit [7] IOMODE indicating I/O DATA mode */
```

Step 2: write the packet's data into TX FIFO SRAM,

```
outb ( IOaddr, 0xF8 );            /* trigger MWCMD REG. F8H with write_ptr++ */
/* u8 TX_data[]: the transmitting data, int TX_length: the length of TX_data[] */
if ( io_mode == DM9000A_BYTE_MODE ) { /* I/O 8-bit Byte mode if ( io_mode == 1 ) */
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 == DM9000A_WORD_MODE ) { /* IO 16-bit Word mode if(io_mode== 0) */
(int) length_tmp = ( TX_length + 1 ) / 2;    /* depended on Word mode for loop */
for ( i = 0; i < length_tmp; i++ ) /* loop to write a Word data into TX FIFO SRAM */
outw ( ( (u16 *) TX_data)[i], IOaddr + 4 );       }
```

Step 3: write the transmitting length into MDRAL REG. FCH and MDRAH REG. FDH,

```
/* write high byte of the TX data length into MDRAH REG. FDH */
iow ( 0xFD, (TX_length >> 8) & 0xff );
/* write low byte of the TX data length into MDRAL REG. FCH */
Iow ( 0xFC, TX_length & 0xff );
```

Step 4: start to transmit a packet out,

```
iow ( 0x02, 1 );    /* set a TX request command, TXREQ, Bit[0] of TCR REG.02 */
```

5.5.2 To 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 ( 0x01 );     /* read NSR REG. 01 status 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 ( 0xFE );         /* got DM9000A interrupt status in ISR */
if ( INT_status & 0x02 ) { /* TX success */  /* check if PTS Bit [1] = 1, TX ok */
     iow ( 0xFE, 0x02 ); }              /* clear PTS Bit [1] latched in ISR */
// if ( INT_status & 0x01 )   { /* RX success */  iow ( 0xFE, 0x01 ); }
```

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 size) in the DM9000A MAC. There are four bytes for the MAC header of each packet saving in the RX FIFO SRAM, and using the two registers of MRCMDX REG. F0H and MRCMD REG. F2H 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" or LSB Bit [0] = 1. But, if Bit [1:0] is neither "01"b nor "00"b, the DM9000A MAC/ PHY must do a software-reset in order to recover from the un-stable states between the system bus and the DM9000A LAN chip. The second byte 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 either 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:

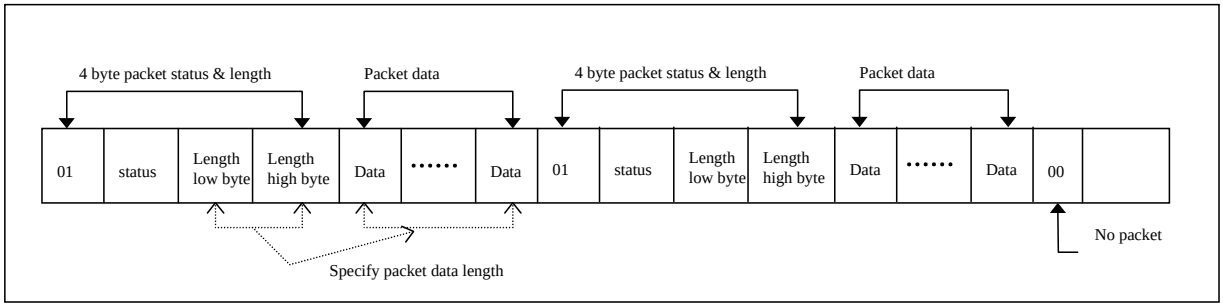


Figure 5.2 Block Diagram of the Received Packets.

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 ( 0xFE );          /* got DM9000A interrupt status in ISR */
if ( INT_status & 0x01 ) { /* doing ch.5.6.2 */ /* check if PRS Bit [0]=1, RX ok */
iow ( 0xFE, 0x01 ); }         /* clear PRS latched in ISR Bit [0] (see ch.5.5.2) */
```

5.6.2 Packet Reception

The definition of MRCMDX (REG. F0H) is the memory data read command without address increment register. MRCMDX is only used to read the received packet ready flag "01", or LSB Bit [0] = 1 and Bit [7:5] & Bit [4:2] indicated the IP/TCP/UDP checksum status and type, from the RX FIFO SRAM.

Here is an example to get the RX ready message:

```
(u8) RX_ready = ior ( 0xF0 );      /* dummy READ the packet RX ready flag */
RX_ready = (u8) inb( IOaddr + 4 ); /* got the most updated data */
if ( (RX_ready & 0x01)==1 ) { /* 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 & 0x2)!=0 ) { /* stop interface and wait to *reset device */
iow ( 0xFF, 0x80 );           /* stop INT request */
iow ( 0xFE, 0x0F );           /* clear ISR status */
iow ( 0x05, 0x00 );           /* stop RX function */
(u8) device_wait_reset = TRUE;     /* raise the MAC/ PHY software-reset flag */
/* iow ( 0x00, 0x01 );        // it's quick software-reset to replace above*
udelay ( 10 );
iow ( 0x00, NCR_set );
iow ( 0xFF, 0x80 );
iow ( 0x05, RCR_set | 1 );    */
/* then, re-new system variables and counters for dropped and queued packets... */
}
```

5.6.3 To Check the Packet Status and Length

MRCMD (REG. F2H): 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. F2H). The size, the increment of the RX read pointer, is depended on the system application, which the I/O bus width is Byte/ Word to increase one or two bytes respectively. MRCMD (REG. F2H) 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 ( 0xFE ) >> 7;  /* ISR Bit [7] IOMODE indicating DATA I/O mode */
outb ( 0xF2, IOaddr );             /* trigger MRCMD (REG. F2H) with read_ptr++ */
/* int RX_status: the RX packet status, int RX_length: the length of the RX packet*/

if ( io_mode == DM9000A_BYTE_MODE ) { /* I/O 8-bit Byte mode if ( io_mode == 1 ) */
     RX_status = inb ( IOaddr + 4 ) + ( inb ( IOaddr + 4 ) << 8 );
     RX_length = inb ( IOaddr + 4 ) + ( inb ( IOaddr + 4 ) << 8 );    }

else if ( io_mode== DM9000A_WORD_MODE ) { /* I/O 16-bit Word mode if(io_mode== 0) */
     RX_status = inw ( IOaddr+ 4 ); /* the high byte is the status as RSR REG. 06 */
     RX_length = inw ( IOaddr + 4 );         }
```

5.6.4 Receive the Packet's Data

The read pointer will be increased after reading the memory read command MRCMD REG.F2. 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 == DM9000A_BYTE_MODE ) { /* I/O 8-bit Byte mode if ( io_mode == 1 ) */
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== DM9000A_WORD_MODE ) { /* I/O 16-bit Word mode if(io_mode== 0) */
(int) tmp_length = ( RX_length + 1 ) / 2;    /* Word mode */
for ( i = 0 ; i < tmp_length ; i++ ) /* loop to READ a Word data from RX FIFO SRAM*/
 ( (u16 *)RX_data)[ i ] = inw ( IOaddr + 4 );          }
```

6 The Others

6.1 How to transmit and receive more than 2048-byte packets

If a system transmits or receives more than 2048-byte packets, which size is although over 802.3 spec., the TCR (REG. 02) Bit [6] TJDIS and RCR Bit [6] WTDIS should be set to "1".
For example,

```
#define   TCR  0x02
#define   RCR  0x05
iow ( TCR , ior ( TCR ) | 0x40 );
iow ( RCR , ior ( RCR ) | 0x40 );
```

6.2 The performance of DM9000A

The performance of the DM9000A is depended on the capability of MPU/ MCU. If the MPU/MCU is so fastest to match the timing of the maximum speed for the DM9000A IOR#/ IOW#, the performance will be: 10ns + 10ns = 20ns -> 50Mbps.

8-bit mode	Receive or transmit data	->	8 * 50Mbps	->	400 Mbps
	RX & TX data at the same time	->	8 * 50Mbps/ 2	->	200 Mbps
16-bit mode	Receive or transmit data	->	16 * 50Mbps	->	800 Mbps
	RX & TX data at the same time	->	16 * 50Mbps/ 2	->	400 Mbps

Note: The DM9000A is the 10/100 Mbps Ethernet NIC, so the maximum speed is 100 Mbps.

6.3 WOL (Wake-up on LAN)

The DM9000A LAN chip also supports the WOL function and the pin 22 WAKE (or the pin 38 LED2, if setting the EEPROM Word 7 Bit [13:12] = 10 in 16-bit mode) output the wake-up signal to the embedded system. There are three methods to generate WOL signals, which are the Magic Packet, the Link Change, and the Sample Frame types. This section will discuss how to implement the WOL function in the DM9000A MAC.

(1) Magic Packet

If the DM9000A receives a broadcast packet which content is "FF:FF:FF:FF:FF:FF"+"16 times of the Node address", then the wake-up pin will be activated. Please note that the Node address must be the same as the Physical address in the PAR REG. 10H~REG. 15H.

For example:

```
#define   NCR  0x00
#define   WCR  0x0F
iow ( WCR , ior ( WCR ) | 0x08 );  /* Magic Packet event enable */
iow ( NCR , ior ( NCR ) | 0x40 );  /* Wake-up remote enable */
/* If a physical address in DM9000A PAR REG. 10H ~ REG. 15H is 00:60:6e:90:00:01 */
/* DM9000A WOL would be active, while received the Magic packet as follows, */
/* 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 (+ another DM9000A TX auto 4-byte CRC appends)        */
```

(2) Link Change

If the link status of the internal PHY has been changed, the wake-up pin will be active.

For example,

```
#define   NCR  0x00
#define   WCR  0x0F
iow ( WCR , ior ( WCR ) | 0x20 );  /* Link Change event enable */
iow ( NCR , ior ( NCR ) | 0x40 );  /* Wake-up function enable */
```

(3) Sample Frame

The Sample Frame would be made up of 6 packets and the maximum size of each packet can be 2048-byte. The wake-up pin will be active if the DM9000A receives only one set of the sample frames. The following description will show how to set the Sample Frame in detail.

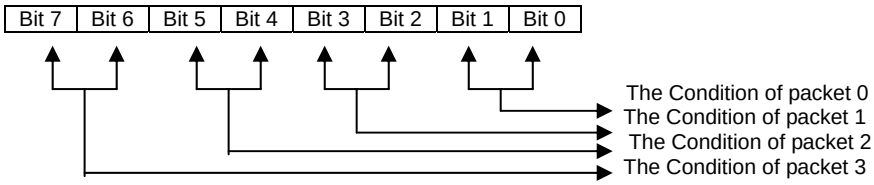
- Firstly close the DM9000A receiver function and set the Bit [0] in RCR REG. 05 to "0".
- Stop the self recover function. Set the Bit [7] PAR in IMR REG. FFH to be "0".

- Save the 6 packets max into the DM9000A FIFO SRAM. The format is shown below,

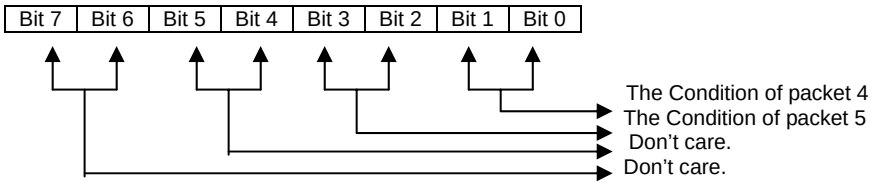
m-byte3 / packet 3	m-byte2 / packet 2	m-byte1 / packet 1	m-byte0 / packet 0	DM9000A TX&RX Memory location
packet 3-byte 0	packet 2-byte 0	packet 1-byte 0	packet 0-byte 0	0000h
packet 3-byte 1	packet 2-byte 1	packet 1-byte 1	packet 0-byte 1	0004h
:	:	:	:	0008h
:	:	:	:	000Ch
:	:	:	:	0010h
:	:	:	:	:
packet 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	DM9000A TX&RX Memory location
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 5-byte 2047	packet 4-byte 2047	Cond. 1-byte 2047	Cond. 0-byte 2047	3FFCh

The format of Condition 0



The format of Condition 1



If the condition is "00" or "01", don't care the byte; if "10", check the byte is matched or not; and if "11", the DM9000A MAC stops checking the sample frame.

- Restart the Sample Frame function by setting the Bit [4] SAMPLEEN in WCR REG. 0FH to "1" and enable the WOL function by setting the WAKEEN Bit [6]=1 in NCR (REG. 00).
- Restart the DM9000A receiver function by setting the Bit [0] in RCR REG. 05 to be "1".

For example,

```
#define   NCR  0x00
#define   RCR  0x05
#define   WCR  0x0F
#define   MWRL 0xFA
#define   MWRH 0xFB
#define   IMR  0xFF
iow ( RCR , ior ( RCR ) & 0xfe );
iow ( IMR , ior ( IMR ) & 0x3f );

/* Here, the Sample Frame put in the DM9000A FIFO SRAM 16KB: 0x0000~ 0x3FFF */
for (i = 0, sample_ptr = 0; i < sample_length; i++, sample_ptr += 4) {
/* set sample_ptr to MWRL as low byte and to MWRH as high byte in DM9000A SRAM */
     iow ( MWRL, sample_ptr & 0xff );
     iow ( MWRH, (sample_ptr >> 8) & 0xff );
     outb ( 0xF8, IOaddr );   /* Output port number MWCMD=0xF8 TX WRITE locking */
     if (i % 2) outw ( ( ((u16 *) sample_frame)[(i+1)/2] >> 8), IOaddr + 4 );
     else outw ( ((u16 *) sample_frame)[(i+1)/2], IOaddr + 4 );  }/* 16-bit mode */

/* set the condition Byte [0] & Byte [1] of the Sample Frame in 0x2000~ 0x3FFD */
for (i = 0, sample_ptr = 0x2000; i < sample_length; i++, sample_ptr += 4) {
/* set sample_ptr to MWRL as low byte and to MWRH as high byte in DM9000A SRAM */
     iow ( MWRL, sample_ptr & 0xff );
     iow ( MWRH, (sample_ptr >> 8) & 0xff );
     outb ( 0xF8, IOaddr );   /* Output port number MWCMD=0xF8 TX WRITE locking */
     outw( 0x0002, IOaddr + 4 ); } /* Fill Bit [1:0]="10" of condition 0 in pkt 0 */
// iow ( MWRL, (sample_ptr+4) & 0xff); iow ( MWRH, ((sample_ptr+4) >> 8) \
// & 0xff); outw( 0x0003, IOaddr + 4 ); /* Fill Bit [1:0]="11" to stop checking */

iow ( WCR , ior ( WCR ) | 0x10 );  /* Sample Frame enable */
iow ( NCR , ior ( NCR ) | 0x40 );  /* Wake-up remote enable */
iow ( RCR , ior ( RCR ) | 0x01 );
```

6.4 IP/TCP/UDP checksums Offload

The DM9000A chip supports the IP/TCP/UDP checksum generations and status checking. And enable the IP/TCP/UDP checksums offload function in the TCP/IP upper layers of the embedded OS, while setting the DM9000A IP/TCP/UDP checksum generations and checking. List powerful function bits in TCSCR & RCSCSR:

TX Check Sum Control Register (TCSCR REG. 31H):

Bit	Name	Description
2	UDPCSE	UDP Check Sum Generation Enable
1	TCPCSE	TCP Check Sum Generation Enable
0	IPCSE	IP Check Sum Generation Enable

RX Check Sum Control Status Register (RCSCSR REG. 32H):

Bit	Name	Description	
7	UDPS	UDP Check Sum Status	0: checksum OK, if UDP packet received.
6	TCPS	TCP Check Sum Status	0: checksum OK, if TCP packet received.
5	IPS	IP Check Sum Status	0: checksum OK, if IP packet received.
4	UDPP	UDP Packet if indicating 1	
3	TCPP	TCP Packet if indicating 1	
2	IPP	IP Packet if indicating 1	
1	RCSEN	Receive Check Sum Enable Checking	
		When set, the checksum status will be stored in packet first byte of status header.	
0	DCSE	Discard Check Sum Error Packets	
		When set, if IP/TCP/UDP checksum field is error, this packet will be discarded.	

For example,

```
#define   TCSCR     0x31
#define   RCSCSR    0x32
iow ( TCSCR , 0x07 );         /* TX IPCSE + TCPCSE + UDPCSE cs-generation enable */
iow ( RCSCSR , 0x03 );        /* RX checksums checking bad-pkt-discard enable */
/* IPS/TCPS/UDPS indicating check-sum status, while RX IP/TCP/UDP (good) packets */
```

6.5 AUTO-MDIX and Application circuit

When design AUTO-MDIX circuit of the DM9000A in the embedded system, the following layout guide must be cared: Traces routed from RXI± and TXO± to the transformer should run in close pairs directly to the transformer such as YCL-PH163539. The network interface should be void of any signals other than the TXO± and RXI± pairs between the RJ-45 to the transformer and the transformer to the DM9000A.

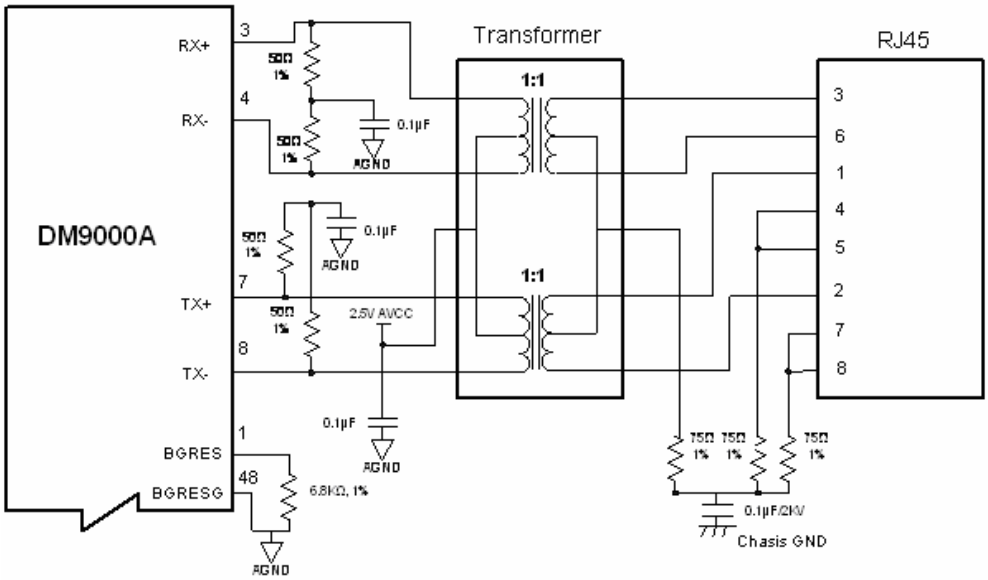


Figure 6.1 AUTO-MDIX 10Base-T/100Base-TX Application.

The DM9000A is default turning on AUTO-MDIX feature, and there are two ways to disable AUTO-MDIX function:

1. To set the SROM Word 7, Wake-up Mode Control, Bit [14] = 1 then zero to re-load it,

```
srom_write (0x07, 0x180); /* PHY power-on but disable Auto-MDIX setting into SROM */
iow ( 0x0B, 0x20 );      /* REEP Bit [5] of EPCR REG. 0BH to re-load EEPROM value */
iow ( 0x0B, 0x00 );      /* REEP Bit [5]= 0 normal after 400 us delay for re-load */
```

2. To write high "1" to the Bit [4] Mdix-down in the PHY REG. 20,

```
phy_write(20, 0x0010);/* Mdix_down Bit[4]= 1 with Mdix_fix_Value Bit[5]=0 for MDI */
phy_write(20, 0x0030);/* Mdix_down Bit[4]=1 with Mdix_fix_Value Bit[5]=1 for MDIX */
```