DP83800 Software Programmers Guide

National Semiconductor
Application Note 995
Roman Baker
July 1995

The DP83800 Software Programmers Guide is designed to aid in the development of software for the DP83800 10/100 ISA based network adapter. It is recommended that the DP83800 datasheet be read and understood before reading this document.

TABLE OF CONTENTS

1.0 INTRODUCTION TO THE DP83800

1.1 10/100 Mb/s Operation

1.2 Buffer Architecture

1.3 Resource Configuration

1.4 Registers

1.5 Node Management

1.6 EEPROM Interface

1.7 Media Independent Interface

1.8 CAM Interface

2.0 CONFIGURATION

2.1 Plug and Play Configuration Mode

2.2 Legacy ISA Configuration Mode

2.3 EISA Configuration Mode

2.4 Changing Configuration Mode

3.0 INITIALIZATION

3.1 Software Reset

3.2 Transmit Modes

3.3 Normal/Early Transmit

3.4 Automatic Transmit Packet Padding

3.5 Transmit Retries

3.6 Receive Modes

3.7 Interrupt Options

3.8 Node Address Initialization

3.9 Enabling the DP83800

4.0 RUN-TIME OPERATION

4.1 Transmission

4.2 Reception

5.0 OTHER DP83800 FEATURES

5.1 Full-Duplex Operation

5.2 General Purpose Timer

1.0 INTRODUCTION TO THE DP83800

The DP83800 is designed specifically for ISA bus adapter applications. Its design is optimized for high throughput, low CPU utilization and low cost.

It implements a simple FIFO-based slave I/O interface to the ISA bus to simplify the task of writing network device drivers. Additionally, the DP83800 contains many features that are aimed specifically at increasing overall performance in the most popular network environments.

1.1 10/100 Mb/s Operation

The DP83800 is capable of operating as a 10 Mb/s standard Ethernet® controller or as a 100 Mb/s Fast Ethernet controller. When coupled with National Semiconductor's DP83840 10/100 Mb/s physical layer, mode configuration is automatic. The DP83800 provides an interface to the DP83840 through the Media Independent Interface (MII). Through the MII, software can set the physical layer to auto negotiate or it can force any physical layer configuration mode desired.

1.2 Buffer Architecture

The DP83800 provides an easy to use buffer architecture. As packets are received, they are stored sequentially in the 5 kbyte receive FIFO. The status and receive byte count are also stored in a separate FIFOs. The driver need only perform a series of reads from a particular register to move the packet from the FIFO into system memory. Similarly, the driver need only perform a series of writes to the same register in order to transmit a packet. This simplified buffer architecture enables the driver to implement streamlined data transfer routines.

1.3 Resource Configuration

The DP83800 conforms to the Plug and Play 1.0a specification for auto-configuration of Plug and Play ISA devices. Because of this, the DP83800 will be automatically configured when placed in any system which supports Plug and Play isolation and configuration management routines. In these systems, the drivers need only query the resident configuration manager to obtain all necessary configuration information such as base I/O address and IRQ.

The DP83800 also supports a ''legacy'' mode of operation where it behaves like traditional ISA adapters and powers up with pre-assigned resources. These resources are stored in the EEPROM devices on the board and are loaded into the DP83800 on power up. In this mode, the driver can obtain configuration information in several different ways as described below.

RRD-B30M105/Printed in U. S. A.

The driver can obtain the base I/O location using methods inherent in the specific driver specification being written to (such as NET.CFG files for ODI and PROTOCOL.INI files for NDIS drivers) and then read some of the DP83800 configuration registers to obtain the other resources.

The driver can also perform its own Plug and Play isolation to locate the DP83800 adapter. Once isolated, the driver can read the resource information from special Plug and Play registers. This allows the driver to locate the adapter in traditional ISA systems without having to perform dangerous searches of I/O space and without requiring configuration files.

And lastly, the DP83800 supports EISA configuration registers when used in an EISA system. In these systems, resources are assigned by the EISA BIOS at boot time and programmed into the IO/IRQ configuration register and the boot PROM configuration register.

In all modes, the DP83800 will respond to Plug and Play isolation sequences. This is the standard method of changing adapter configuration.

Supporting these three modes of configuration allows the DP83800 to be easily configured and accessed in today's environments as well as future ones.

1.4 Registers

1.4.1 Run-Time Registers

The DP83800 contains over 45 run-time registers used to operate the device. Because the ISA platform has limited I/O space, the DP83800 uses paged registers in order to fit these 16-bit registers in 32 bytes of ISA I/O space. To reduce the need to continually switch register pages, the DP83800 has registers arranged on one ''fixed page'', as well as 5 ''variable pages''. The fixed page is always present in the first 16 I/O locations of the base address. The second 16 I/O locations hold the variable page registers. Which of the variable pages is present is indicated in the master Command Register (CR, fixed page, offset 00h). The grouping is arranged so that after initial driver configuration, the driver can leave page 1 the active variable page and have little or no need to change to another.

Please refer to the DP83800 datasheet for a detailed description of each of the registers and its bit significance.

1.4.2 Plug and Play Registers

The DP83800 has a set of Plug and Play registers used for configuration of the device. They are available only through the Plug and Play interface after the adapter has been isolated.

The DP83800 has all of the standard Plug and Play registers used for isolation and resource allocation. It also has a mode configuration register used to alter the power-up actions of the DP83800. Please refer to the DP83800 datasheet for a detailed description of all Plug and Play registers.

1.4.3 EISA Configuration Registers

The DP83800 has a set of registers which can be used in an EISA system. These registers allow for detection of the adapter as well as configuration.

The configuration registers are slot specific when placed in an EISA machine. For example, if the DP83800 is placed in slot 3 of the host machine, the EISA Product ID would begin at 3C80h. Please refer to the DP83800 datasheet for a detailed description of all EISA registers.

1.5 Node Management

1.5.1 Management Information Base (MIB) Statistics Counters

The DP83800 contains an extensive block of statistics counters that are automatically updated as each event occurs. The counters provide a set of statistics compliant with the following management specifications: MIB II, Ether-like MIB and IEEE MIB.

With these counters, it is easy for drivers to keep accurate statistics without needing to count them on a per-packet basis. This allows for faster transmit and receive routines and greater overall driver performance.

1.5.2 MIB Interface

To access the MIB statistics counters, the DP83800 provides a register interface to the MIB block. This interface consists of the MIB Control Register (MCR, variable page 3, offset 14h) and the MIB Data Registers (MDR0 and MDR1, variable page 3, offsets 10h and 12h).

The MCR allows the software to choose which set of statistics to enable (see datasheet) as well as clear, test and access all the counters. In order to access the MIB counters, the software need only set the Access Pointer Reset bit (bit 9) to reset the access pointers and then set the offset in bits 5-0 to begin reading the counters. The software then reads from the MDR registers to retrieve the information.

The two MDR registers provide a 32-bit interface to the MIB counters. If the software performs 32-bit input instructions to read the data, the CPU will read both MDR0 and MDR1 consecutively to get the full 32-bit value for each statistic. The software may continue to read from the MDR registers until all statistics have been gathered.

1.6 EEPROM Interface

The DP83800 provides an interface to directly access the NM93C46 (or compatible) device which holds the Ethernet address, Plug and Play information as well as other configuration data. Specifications for the NM93C46 can be obtained from the National Semiconductor Corp. Memory Databook.

The interface consists of the EEPROM Control Register (EECR, variable page 2, offset 10h) and the EEPROM Data Register (EEDR, variable page 2, offset 12h). To read from the EEPROM, software must set the EEPROM command field in bits 7–0 of the EECR to instruct the EEPROM to read from a specified location (see NSC Memory Databook). The software would then poll bit 15, the EEPROM In Use bit of the EECR to determine if the read operation is completed. When bit 15 is cleared, the data is ready to be read from the EEDR.

Write operations are similar to read operations except that the write data needs to be placed in the EEDR before setting bits 7–0 of the EECR to instruct the EEPROM to write to a specific location.

1.7 Media Independent Interface

The DP83800 provides an interface so that software can communicate with the Physical Media Device (PMD) over the Media Independent Interface (MII). Typically the MII is used to determine if the current PMD supports a particular mode or to configure the PMD to operate in a specific mode.

The MII consists of the MII Control Register (MICR, variable page 2, offset 14h) and the MII Data Register (MIDR, variable page 2, offset 16h).

To perform a read operation from a register in the PMD, the software would write to the MICR a value which contains the register address field (bits 4–0), the PMD address field (bits 9–5) and the access mode set to management read (bits 11–10 set to 01 respectively). The software would then poll bit 15 of the MICR to wait for the operation to complete. When the operation is complete, the data is available in the MIDR.

Write operations are similar. Like the EEPROM interface, to perform a write operation the data register (MIDR) first needs to be programmed with the value to write. Next, the software must write to the MICR with the register address field, the PMD address field and the value 10 in bits 11–10 to indicate a management write. When polling of bit 15 of the MICR returns 0, the operation is completed.

Before using the MII, the software should issue a management reset to the MII to synchronize the DP83800 and the PMD. This is achieved by writing an 11 to bits 11–10 in the MICR.

1.8 CAM Interface

The DP83800 provides an interface to the on-board CAM used for address matching. Through the interface, the software can read and write to the CAM, mask certain CAM entries and determine which CAM entry caused the last match.

1.8.1 CAM Read and Write Accesses

Before any software CAM accesses, the CAM must be disabled. To do so, bit 15 of the CAM Control Register (CCR, variable page 2, offset 18h) must be cleared. After the access to the CAM is performed, bit 15 must be set again to resume CAM operation.

To perform a read from the CAM, the software must first set the CAM entry pointer (the CAM has 14 locations, 10 for physical/multicast address and 4 for broadcast locations) to the desired location, then set bit 6 (the CAM read bit) to initiate the read operation. The address in the specified CAM location will then be available in the CAM Data Register (CDR, variable page 2, offset 1Ah). For a physical/multicast address, the full 6 bytes must be retrieved by reading 3 consecutive words from the CDR. The broadcast CAM entries are slightly different and require only 2 word reads from the CDR (see section in datasheet on CDR for further details).

To perform a CAM write operation, the software needs to set the CAM entry pointer, set bit 7 (the CAM write bit) in the CCR instead of bit 6 and then write the data to the CDR instead of reading from it. This will set the specified CAM entry.

1.8.2 CAM Mask Register

Once the address data has been written to the CAM, the appropriate value in the CAM Mask Register (CMR, variable page 2, offset 1Ch) needs to be set in order for the CAM logic to use the entry. With the CMR, addresses can be left in the CAM and enabled or disabled quickly simply by masking it.

1.8.3 CAM Match Register

The CAM Mask Register (CMTR, variable page 2, offset 1Eh) is used to determine which CAM entry caused the last address match. Bits 0-13 in the CMTR correspond to the 14 CAM entries. If, for instance, bit 1 is set in the CMTR, then the address in CAM entry 1 caused the match.

1.8.4 Other CCR Bits

Additionally, the CCR contains bits which control other address match criteria. The Accept All Broadcast bit, when set, allows all broadcast packets to pass address match. The Accept All Multicast does the same for all multicast addresses and the Accept All Physical works for physical addresses. The CAM need not be disable when setting these bits.

2.0 CONFIGURATION

The DP83800 supports three modes of configuration: Plug and Play mode, legacy ISA mode and EISA mode. The configuration requirements vary depending on the configuration mode selected.

2.1 Plug and Play Configuration Mode

Plug and Play configuration mode is generally selected when the adapter is to be placed in a system that has a Plug and Play BIOS, Plug and Play device drivers or a Plug and Play aware operating system. When in such a system, a DP83800 configuration program need only ensure that the EEPROM location containing the mode configuration byte (word 28h, low byte) be set to load only the mode configuration information. (The I/O, IRQ and boot ROM resources are not loaded from the EEPROM.) The host system will isolate and assign resources to all Plug and Play adapters in the system at power-up.

When the driver loads, it must query the resident configuration manager to obtain the resource information. Refer to the Plug and Play Device Driver Kit for information on how to locate the Plug and Play Configuration Manager and how to obtain the resource information.

2.2 Legacy ISA Configuration Mode

Legacy ISA configuration mode is generally used when the adapter is placed into an ISA system that does not support Plug and Play isolation and resource allocation. In this mode, the DP83800 will read the I/O, IRQ and boot ROM configuration information from the EEPROM on power-up. Drivers will need to use native methods, such as changes to the NET.CFG file in the ODI case, to determine the I/O base address of the adapter. Once the base address is determined, the driver can read DP83800 registers to determine IRQ and boot ROM information.

In legacy mode, the DP83800 will still respond to Plug and Play isolation sequences. Therefore, if an adapter is left in legacy mode and is placed in a Plug and Play system, the resources read from EEPROM at power-up will be overwritten by the Plug and Play system. With some Plug and Play systems, an error message may be displayed indicating that the adapter already had resources assigned before Plug and Play isolation was performed.

2.3 EISA Configuration Mode

The DP83800 can be programmed to respond to access to the EISA configuration registers in an EISA system. The EISA BIOS can use the registers to program the DP83800 at power-up using parameters specified in an EISA configuration file. To enable EISA mode, a DP83800 configuration program need only ensure that the EEPROM location containing the mode configuration byte (word 28h, low byte) be set to load only the mode configuration information. The host system will assign resources to the adapter at power-up.

Once EISA mode is enabled, changes to the configuration can be made through the EISA Configuration Utility (ECU) which comes with the EISA machine.

2.4 Changing Configuration Mode

In all configuration modes, the DP83800 will respond to Plug and Play isolation sequences. Once the adapter has been isolated, software can access the PnP Mode Configuration Register (PMCR, index 0F0h) in the Plug and Play register set (see datasheet for complete list of Plug and Play registers). From here, the adapter configuration can be changed by setting the appropriate bits and issuing a configuration ''snapshot'' by setting the CS bit in the PMCR. When this is done, the mode configuration register and the IO/IRQ configuration registers are stored to the EEPROM.

3.0 INITIALIZATION

Several steps need to be performed in order to ready the DP83800 for operation. The following section reviews these steps in detail.

3.1 Software Reset

The first step in initializing the DP83800 is to perform a software reset to the chip. This ensures that all registers are initialized to their reset values even if a driver was previously run on the adapter.

In order to perform a software reset, the driver must set bit 14, the software reset bit in the Command Register, as well as bit 3, the modify bit. (The modify bit *must* be set to alter any of the bits in the CR except for the page select bits.)

3.2 Transmit Modes

The DP83800 supports three different modes of transmission. The driver can select between blind transmit, halt on error, and transmit acknowledge mode.

3.2.1 Blind Transmit

This is the default transmit mode of the DP83800. In this mode, the DP83800 will, upon completion of transmitting the first packet in the FIFO, move to the next and immediately attempt to transmit it. The DP83800 will continue to do this until there are no more packets in the transmit FIFO. It will move to the next packet regardless of whether the transmission was successful or not.

This mode is particularly useful in environments where unsuccessful transmissions need not be immediately reported to upper layer software.

To enable blind transmit mode, set bits 5–4 in the Transmit Configuration Register (TXCR, variable page 0, offset 16h) to 00.

3.2.2 Halt on Error Transmit

When halt on error mode is enabled, the DP83800 will continue to transmit all of the packets in the transmit FIFO unless there is a fatal transmit error (such as excessive collisions, out-of-window collisions, etc.). If such an error occurs, the transmitter stops and the DP83800 will issue a transmit error event. The driver can then determine the type of error. Once the transmit status is read, the DP83800 will continue to transmit the remaining packets in the FIFO.

This mode allows for faster transmit routines while enabling the driver to handle error events as needed.

To enable halt on error mode, set bits 5–4 of the TXCR to 01.

3.2.3 Transmit Acknowledge Mode

When transmit acknowledge mode is enabled, the DP83800 transmits packets from the FIFO one at a time. After the completion of a packet transmission, the DP83800 issues either a successful or erred transmission interrupt. The driver needs to then read the transmit status for the next transmission to begin.

This mode is the slowest transmission mode because of the need to acknowledge all packets. It does allow for tracking of transmit packets on a per-packet basis.

To enable transmit acknowledge mode, set bits 5–4 of the TXCR to 10.

3.3 Normal/Early Transmit

The DP83800 also supports normal and early transmission of packets in all of the aforementioned transmit modes. With normal and early transmit modes, the driver can be optimized to provide the best possible throughput in the target environment (10 or 100 Mb/s operation, full or half duplex, etc.).

3.3.1 Normal Transmit

In normal transmit mode, transmission of the packet is not started until after the last byte of information is entered into the transmit FIFO. At that point, the DP83800 will begin to transfer information to the physical media device (PMD).

To ensure that the DP83800 is set up for normal transmit, the driver must set the Transmit Minimum Threshold Register (TMTR, variable page 1, offset 12h) to the maximum Ethernet packet size, 1518. With this value, the transmit state machine will not initiate transmit until either the maximum 1518 bytes are entered into the FIFO or the last byte of the packet, if smaller than 1518, is entered.

It is recommended that normal transmit mode be used when the DP83800 is configured for 100 Mb/s operation. At 100 Mb/s, the wire speed is faster than the maximum throughput of the ISA bus. If early transmit mode is used at 100 Mb/s, it is likely that the transmit FIFO will underrun and the packet will be lost. Setting the DP83800 to normal transmit ensures that it will not underrun.

3.3.2 Early Transmit

In early transmit mode, transmission of the packet will begin when the number of bytes specified in the TMTR are entered into the transmit FIFO. At this point, transmission will begin and continue until either the total number of bytes for the packet are transmitted or until the transmit FIFO underruns.

Because the possibility of an underrun still exists at 10 Mb/s, the driver must ensure that either the transmit data transfer is an atomic operation or that the driver will not be interrupted long enough to cause the transmit FIFO to underrun.

When early transmit mode is used, transmission of the packet occurs at the same time data is being transferred from system memory to the transmit FIFO. This can improve overall performance in many instances.

3.4 Automatic Transmit Packet Padding

The DP83800 supports automatic transmit padding of packets. This is useful when the upper layer protocol passes the driver a packet to send where the total byte count is under the minimum Ethernet packet size, 64 bytes. In traditional Ethernet controllers, the driver needed to pad the packet in software before passing it on to the hardware. When automatic transmit packet padding is enabled, this padding in software is not required. The hardware will automatically pad the packet as it is being transmitted.

3.5 Transmit Retries

In compliance with the specification, an Ethernet controller will attempt to transmit a packet up to 16 times. For instance, if a controller tries to transmit and collides, it will try 15 more times. If after 16 attempts it is still unsuccessful, transmission is usually aborted.

Many network environments require that the driver attempt to re-transmit packets that failed. In the past, software needed to keep track of how many times a packet was attempted. The DP83800 has a register called the Transmit Retry Register (TXRR, variable page 0, offset 1Ah) which can be used to program the DP83800 to automatically attempt to transmit these packets again.

The value programmed into the TXRR specifies the number of additional retries in multiples of 16. For example, if it is set to 0, the DP83800 will try the normal 16 times. If set to 1, it will try the normal 16 times plus an additional 16 times.

3.6 Receive Modes

As with transmit, the DP83800 supports multiple modes of reception so that the driver can optimize the hardware operation for the target environment. The DP83800 supports three modes of reception: normal, early and burst.

3.6.1 Normal Receive

In normal receive mode, an interrupt for the receive packet is not issued to the host until the entire packet is received into the receive FIFO. At this point, the driver may move the packet from the receive FIFO into system memory.

To ensure normal receive mode, the driver must set bit 15 of the receive threshold register (RTR, variable page 1, offset 18h) to 0. This disables all receive threshold modes.

3.6.2 Early Receive

In early receive mode, the DP83800 is instructed to issue an interrupt to the host after a certain number of bytes are available in the receive FIFO. At this point, the driver may move some of the bytes into system memory.

Early receive mode is particularly useful in environments where data at the beginning of the packet, called lookahead data, can be used to determine if any upper-layer protocol needs the packet. In this environment it can be determined, before the packet is fully received, whether it is needed. If not, the DP83800 can be instructed to abort the current reception and reclaim the space in the FIFO. If the packet is needed by some protocol, the driver can begin to move the data before it is entirely received.

To enable early receive mode, the enable bit in the RTR must be set to 1 to enable the receive threshold logic. Then bit 14 must be set to 0 to instruct the DP83800 to use early thresholds instead of burst thresholds. Lastly, the number of bytes necessary before an interrupt is generated should be specified in bits 0–11. Typically, this is set to be the same as the number of lookahead bytes required by the upper layer software.

When early transmit mode is used, retrieval of the packet from the receive FIFO can be done at the same time the packet is being received from the wire. Again, overall performance can be improved.

3.6.3 Burst Receive

In burst receive mode, the DP83800 will not issue an interrupt for every packet that is received. Instead, an interrupt is generated when the number of receive packets specified in the RTR is reached.

To enable burst threshold mode, bit 15 of the RTR needs to be set to enable the receive threshold logic. Then bit 14 needs to be set to instruct the DP83800 to use burst receive instead of early receive. Lastly, the number of packets needs to be specified in bits 11–0.

Because interrupts are not issued for each packet received, it is possible that a number of packets received could be less than the threshold set in the RTR. An interrupt would not be issued for the packets that are received. For example, let's say that a server normally sends out 3 packets to a workstation to query its status. If the threshold was set to 4 on the workstation, the upper-layer protocol on the workstation would not receive the packets to process until one more packet is received. The server would timeout waiting for the reply and would try to contact the workstation again. When it does, more packets would be received and the threshold would be met. The upper layer protocols on the workstation would then process the first request from the server and reply. There would also be another request in the FIFO which might confuse the protocols.

To help ensure the error condition doesn't occur, the DP83800 has a receive timeout register (RTOR, variable page 0, offset 1Ch). This register can be initialized so that if some packets are received but the burst threshold is not met in some specified time interval, an interrupt is generated anyway. This helps ensure that the workstation or server need not wait for some long software timeout to occur.

Burst receive threshold is typically used in environments where there are a large number of small packets and where hardware interrupts can cause task switches to occur. In these environments, enabling burst receive can greatly decrease the number of time-consuming task switches that occur and boost overall system performance (compared to using normal or early transmit mode).

3.7 Interrupt Options

3.7.1 Interrupt Vector Register

The DP83800 provides a slightly different interrupt reporting mechanism than previous generation Ethernet products. Instead of having an interrupt status register, where each bit represents an event, the DP83800 provides an Interrupt Vector Register (IVR, fixed page, offset 02h). With the IVR, the driver need not read a status register, test each bit, perform an action if that bit is set and then clear the bit. Instead, the driver need only read from the IVR. It can then use the value to determine what to do next. The act of reading the IVR clears the event from the event queue so the driver need not clear any status bits. To service all of the events pending, the driver just reads the IVR, services the event and reads again until the IVR returns 0.

For drivers written in assembly language, it is possible to set up a jump table in memory to further streamline the servicing of interrupts. With a jump table, the driver can simply use the value returned by the IVR to jump directly to a subroutine. The following code illustrates how it can be done:

Jump table initialization for x86 system:

```
JumpTable  label word
ISRNoPending            dw      0
ISRReceiveError         dw      0
ISRReceiveThreshold     dw      0
ISRReceive              dw      0
ISRTransmitError        dw      0
ISRTransmit             dw      0
ISRTxDataSpace          dw      0
ISRTimer                dw      0
ISRSoftware             dw      0
ISRMIB0ver              dw      0
```

(The values for all ISR variables need to be initialized to the offset of the appropriate service routine.)

Sample section of driver interrupt service routine to handle IVR return values as indices into a jump table of service routines. RHPA is a macro that returns the value of the named register.

```
DriverISR0:
      RHPA ax, HPA_int_vector
      mov  bx, ax
      jmp  JumpTable[bx]
```

3.7.2 Vector Sizes

The IVR can be programmed to return either 16- or 32-bit vector values. This choice is often made depending on what environment the driver will be used in. For example, some drivers run in DOS real mode environments. Here, only the offset needs to be specified for the jump table. Therefore, 16-bit values should be used. Other drivers run in 386 enhanced mode where 32-bit values would be desirable.

The IVR size can be set by setting bit 0 of the Mode Configuration Register (MDCR, variable page 0, offset 12h). A 1 will enable 32-bit IVR values while a 0 enables 16-bit values.

3.7.3 Interrupt Mask Register

The Interrupt Mask Register (IMR, variable page 0, offset 14h) is used to inform the DP83800 what events should cause an interrupt and be reported in the IVR.

Choices for the IMR should be based on environments, transmit operating mode, receive operating mode, etc.

3.8 Node Address Initialization

The Ethernet node address is stored beginning at location 24h on the EEPROM device. Most operating systems require that the driver configure itself to accept packets addressed to its own Ethernet address. To do this, the data must be moved from the EEPROM into the CAM.

Setting up the DP83800 to accept its own Ethernet address is a two part process. First the data needs to be moved from the EEPROM into a temporary storage location. Once completed, the CAM can be programmed with the data read from the EEPROM. The sample code below illustrates how this might be done:

```
int Address[3];
int Count;
// First read 3 words from EEPROM
for (Count = 0; Count < 3; Count++)
{
   putReg(EECR, READ | (24h+Count));
   while (getReg(EECR) & InUse)
      ;
   Address[Count] = getReg(EEDR);
}
// Next write 3 words to CAM. First
// disable CAM and the write to
// CAM entry 0.
putReg(CCR, CAM_DISABLE);
putReg(CCR, WRITE | 0);
for (Count = 0; Count < 3; Count++)
{
   putReg(CDR, Address[Count]);
}
// Enable CAM again
putReg(CCR, CAM_ENABLE);
```

3.9 Enabling the DP83800

To complete initialization of the DP83800, the driver must enable the transmitter, the receiver and the generation of interrupts.

Enabling of the transmitter and receiver entails setting bits 9 and 12 (Receive Enable and Transmit Enable respectively) in the Command Register.

Enabling interrupt generation is done by writing a 1 to the Interrupt Enable Register (IER, variable page 1, offset 1Eh).

Once these steps are done, the DP83800 will be ready to send and receive packets over the network.

4.0 RUN-TIME OPERATION

After initialization and configuration, a driver's main purpose is to transmit and receive packets. Although there may be instances when a driver may need to reset or reconfigure hardware, transmission and reception of packets constitutes a large majority of the tasks a driver performs during run-time.

4.1 Transmission

As previously stated, the DP83800 can be configured to operate in a variety of transmit modes. Depending on the mode, transmit operation can vary.

The most common DP83800 transmit configuration will be illustrated below (blind transmit, non-early with automatic transmit padding). Following this will be an explanation of variations required for different modes.

4.1.1 Common Transmit Routine

When the driver gets a packet to be transmitted, it must first assemble the transmit command word. The transmit command word informs the transmitter how to transmit the next packet in the FIFO. Bits 11–0 of the transmit command word indicate the total length of the transmit data. This does not include the transmit command word, handle word or any pad bytes (see below). The other bits in the transmit command word are seldom used. Refer to the datasheet for further explanation.

After the transmit command word is assembled, the driver sends it to the transmit FIFO by writing to the Transmit Receive Data Registers (TRDR0, TRDR1, fixed page, offset 0Ch and 0Eh). Next the transmit handle word should be written. If handles are not being used by the driver, just write any 16-bit value.

Next, the actual packet data should be written to the TRDR registers. (If 32-bit OUT instructions are available, they should be used as they increase performance.) The driver may do 8-, 16- or 32-bit out instructions without need for data steering. After all packet data has been moved to the transmit FIFO, the driver needs to dword align the FIFO. The total number of bytes passed to the FIFO (including the command word, handle packet data) must be divisible by 4. If it is not aligned, the driver needs to write the appropriate number of bytes to the TRDR registers in order to align it.

Once the FIFO is aligned, transmission will begin and continue until the entire packet is sent. Refer to *Figure 1* on the following page for a diagram of the transmit data structure.

4.1.2 Halt on Error and Transmit Acknowledge

Unlike with blind transmit mode, halt on error and transmit acknowledge require that the driver read the Transmit Status Register (TSR, fixed page, offset 06h) to continue transmission after the generation of a transmit interrupt.

In halt on error mode, the driver need only configure the DP83800 to issue interrupts when a fatal transmit error occurs (FIFO underruns, excessive deferrals, out-of-window collisions, etc.). After the transmit error interrupt is generated, the driver would read the vector and then read the TSR to retrieve the status. Upon reading the TSR, the DP83800 transmitter will begin transmitting the next packet in the FIFO.

For transmit acknowledge mode, all transmit status words must be read, regardless of whether the transmission was successful or erred. The driver must program the DP83800 to interrupt when there is a transmit error or transmit complete event. Again, upon reading the TSR, the DP83800 will begin to transmit the next packet in the FIFO.

4.1.3 Transmit Free Space Registers

When a driver needs to send a packet, it can determine if there are enough bytes in the transmit FIFO by reading the Transmit Free Space Register (TFSR, variable page 1, offset 16h). Reading the TFSR will return how many bytes are currently free in the transmit FIFO. If there are enough for the transmit packet, transfer can begin immediately.

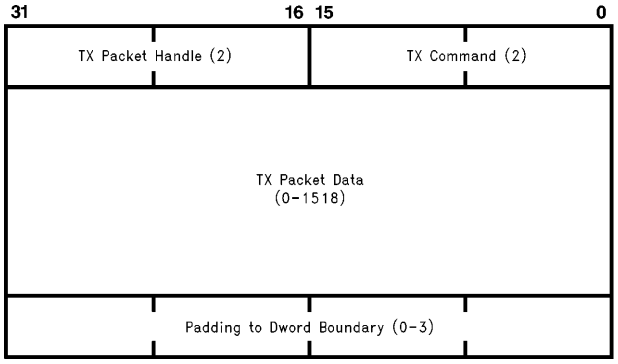
If there are not enough bytes, the driver has two choices: it can wait for enough to become available or it can request that the DP83800 interrupt it when enough space is available.

To have the DP83800 generate an interrupt when a certain amount of free space is available, the driver must program the Transmit Free Space Threshold Register (TFTR, variable page 1, offset 14h) with the desired number of free bytes. When the desired number of bytes are available, the DP83800 will generate an interrupt and the driver can then transmit the packet.

4.1.4 Miscellaneous Transmit Options

When automatic transmit padding is disabled, the DP83800 can be programmed to issue runt packets (packets less than 64 total bytes). This is usually undesirable on the part of a driver but can be useful to diagnostics software. Many times protocols will pass packets where the useful data is smaller than 64 bytes. In this case, the protocol assumes that the packet will be padded with bytes so that it is of the minimum length when transmitted. With automatic transmit padding disabled, the driver will need to do this padding before it is passed to the transmit FIFO.

With early transmits, the driver need only ensure that the data transfer to the FIFO is not interrupted so long as to cause a transmit FIFO underrun. If this happens, it will severely degrade network performance.



TL/F/12460–1

FIGURE 1. Transmit Data Structure

4.2 Reception

As with transmission, the DP83800 can be configured to receive packets in a variety of different manners. The simplest form of operation is when no threshold mode (early or burst) is being used.

4.2.1 Normal Receive Mode

Figure 2 shows the data structure for each packet received. Following reception, the receive packet data is available from the FIFO. Following the receive data are pad bytes (to dword align the packet) as well as one reserved word and the Rx management status. The Rx management status is retrieved when the Receive Packet Advance Register (RPAR, fixed page, offset 0Ah) is read. Also stored for each packet is the status/current size in FIFO (this size value will decrement as data is moved from the FIFO).

When a receive complete interrupt is generated by the DP83800 the packet data is available through the TRDR registers. By simply reading these registers, the data from the packet is moved from the receive FIFO to either a register or into system memory, depending on the instructions used.

The receive FIFO can be accessed byte-wide, word-wide or dword-wide so the driver does not need to perform any byte-steering functions. Because of this, receive fragments presented to the driver by the protocols are easy to handle. Through the Receive Status Register (RSR, fixed page, offset 08h), the driver has access to statistics such as the completion status, whether it was received OK or bad, and the number of bytes remaining to be moved.

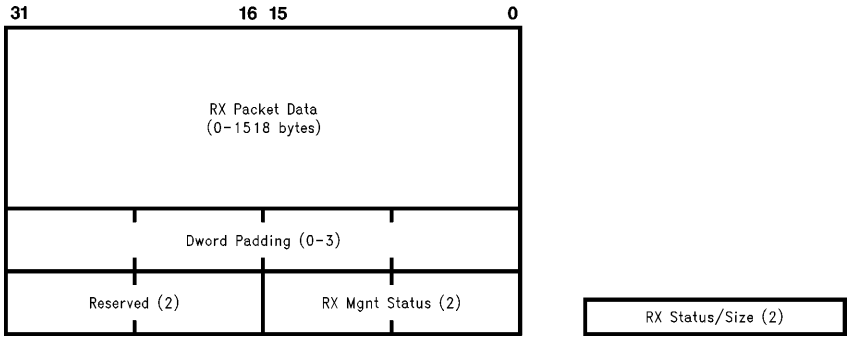
Reading of the RPAR will unconditionally move the receive FIFO pointers to the next packet. Reading of this register is usually performed after the entire packet has been moved from the FIFO. However, if interrogation of the lookahead data indicates that no protocol needs the packet, the driver can simply read the RPAR at any time to advance the FIFO to the next packet.

4.2.2 Early Receive Mode

With early receive threshold mode, the DP83800 will generate interrupts before the entire packet is received in the FIFO. The number of bytes required before an interrupt is generated is specified in the RTR.

Typically, the threshold programmed into the RTR matches the number of lookahead bytes required by the protocols. When these bytes are available, they can be moved from the FIFO into system memory for interrogation. If the packet is not needed by any protocols, the driver can read the RPAR to advance the FIFOs to the next packet.

If the packet is needed by at least one of the protocols, the driver can either wait for blocks of data to be received or can wait for the end of packet interrupt to move the data.



TL/F/12460–2

FIGURE 2. Receive Data Structure

If the driver waits for blocks of data to be received, it can read the RSR to see how many bytes are available in the FIFO. When the desired number of bytes are available, they can be moved. Then the driver will wait again until either the desired number of bytes are available or until the packet reception is complete. This method of reception provides high raw throughput but can ''hog'' processor cycles in a multi-threaded operating system.

Alternatively, the driver can exit the receive routine and wait for the end of packet interrupt. This allows for other processes to use the CPU while the reception is completing but results in multiple interrupts being generated for each packet. This could affect system performance in some environments.

4.2.3 Burst Receive Mode

With burst receive mode, interrupts are not generated for each packet received. Instead, the number of packets specified in the RTR must be received before a receive complete interrupt is generated.

This is helpful if the DP83800 is to be used in an operating system when interrupt hits are costly. (See section 3.6.3 for more information on burst receive mode.)

5.0 OTHER DP83800 FEATURES

5.1 Full-Duplex Operation

With standard Ethernet controllers, only one node may transmit at a time. If more than one attempts to transmit, collisions occur. If a workstation has a packet to transfer but another workstation is already doing so, it has to defer. Collisions and deferrals cause a node to be idle when it could otherwise be active.

With full-duplex operation, a node can begin transmission even if it is receiving a packet from another node. This configuration, when combined with a full-duplex switching hub, eliminates the need to collide or defer and boosts total network performance.

The DP83800 supports full-duplex at both 10 and 100 Mb/s. To configure it to operate in full-duplex, several registers need to be accessed.

First, there needs to be a full-duplex capable PMD attached to the DP83800, such as the DP83840. The PMD can be interrogated over the MII to determine if it can support full-duplex and if it is configured to operate in that mode.

If so, the driver then needs to alter some bits in the Transmit Configuration Register (TXCR, variable page 0, offset 16h) and the Receive Configuration Register (RXCR, variable page 0, offset 18h).

Two bits in the TXCR need to be set. Bit 9 (carrier sense ignore) and bit 8 (heartbeat ignore). And a single bit in the RXCR, bit 3 (accept transmit packets) needs to be set. Once set, the DP83800 will operate in full-duplex mode.

If the DP83800 needs to be changed back to normal mode (for instance, if the PMD is in auto configuration mode and it detects a change from full- to half-duplex), all of the bits mentioned above need to be cleared. If they are not, the DP83800 may cause errors on other nodes.

5.2 General Purpose Timer

The DP83800 contains a general purpose timer which can be used to generate a single or periodic interrupts.

The timer is controlled by the Timer Control Register (TCR, variable page 3, offset 16h), the Timer Max Count Registers (TMR0 and TMR1, variable page 3, offsets 18h and 1Ah) and the Timer Count Registers (TCR0 and TCR1, variable page 3, offsets 1Ch and 1Eh).

The TCR is used to specify whether the timer will run once or in continuous mode. If it is run once, it will generate a single interrupt. If it is set to continuous mode, it will generate periodic interrupts.

The TMR registers control how high the timer must count before issuing an interrupt. The TCR0 and TCR1 registers give a current reading of the timer value.

The timer increments once every 800 ns, or the time it takes to move a single byte over Ethernet.

National Semiconductor Corporation
1111 West Bardin Road
Arlington, TX 76017
Tel: 1(800) 272-9959
Fax: 1(800) 737-7018

National Semiconductor Europe
Fax: (+49) 0-180-530 85 86
Email: cnjwge@tevm2.nsc.com
Deutsch Tel: (+49) 0-180-530 85 85
English Tel: (+49) 0-180-532 78 32
Français Tel: (+49) 0-180-532 93 58
Italiano Tel: (+49) 0-180-534 16 80

National Semiconductor Hong Kong Ltd.
13th Floor, Straight Block,
Ocean Centre, 5 Canton Rd.
Tsimshatsui, Kowloon
Hong Kong
Tel: (852) 2737-1600
Fax: (852) 2736-9960

National Semiconductor Japan Ltd.
Tel: 81-043-299-2309
Fax: 81-043-299-2408