# EtherLink Plus Adapter Technical Reference Manual

**A Member of the EtherLink Product Family**


3165 Kifer Road
Santa Clara, California 95052-8145
Printed in the U.S.A.

Manual Part No. 1569-03
Published January, 1989

## Copyright Statement



## Disclaimer

3Com makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. 3Com shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

# Contents


## Preface v

## Chapter 1: Introduction
About this Document 1-1
Overview 1-2
    Programming 1-2

## Chapter 2: Hardware Reference
Introduction 2-1
Address Maps 2-2
    Adapter I/O Map: 80186 On-board I/O Ports 2-2
    Adapter Memory Map 2-3
    Host I/O Map: Adapter's external I/O ports 2-3
80186 Microprocessor 2-3
82586 Ethernet Coprocessor 2-4
Network Interface 2-4
    82586 Serial Interface 2-4
    8023 Manchester Converter 2-5
    Transceiver 2-5
Adapter Firmware ROM 2-5
Adapter RAM 2-6
Host-Adapter Interface 2-7
    Command Register 2-8
    Data Register 2-8
    Data Register Configuration 2-9
    DMA Transfer 2-9
    Status Flags 2-10
Adapter (80186) Interrupts 2-10
    Internal Interrupts 2-10
    External Interrupts 2-11
Host Interrupts 2-11
Resetting the Adapter 2-12
Station Address 2-13
LED Indicators 2-13
Host ROM 2-13

## Chapter 3: Hardware Interface
Introduction 3-1
Command Register 3-1
Data Register 3-2
Host Control Register 3-2

Host Status Register 3-3
Host AUX DMA Register 3-4
Adapter Control Register 3-5
Adapter Status Register 3-6

## Chapter 4: Command Interface

Introduction 4-1
Primary Command Block Structure 4-1
    Status Flag Usage for PCB Transfer 4-3
    Host to Adapter Request 4-4
    Adapter to Host Request or Response 4-4
PCB Commands 4-5
    Host to EtherLink Plus Adapter PCB Formats 4-5
    EtherLink Plus Adapter to Host PCB Formats 4-9
System ROM Utilities 4-13
    Host I/O Support: INT 80H 4-13
    Network I/O Support: INT 81H 4-16
    Configuration Status: INT 82H 4-17
    Timer Support: INT 83H 4-19
    Download Program Support: INT 84H 4-20
    PCB Command Processor: INT 85H 4-20
    Packet Processor Vector: INT 86H 4-21
    Idle Vector: INT 87H 4-22
    PCB Enqueue Vector: INT 88H 4-22

## Chapter 5: Programming

PCBs 5-1
Interrupts 5-2
Data Transfer and DMA 5-2
ROM Utilities 5-3
Data Structure 5-3

## Appendix A: 80186 Peripheral Control Block Programming

## Appendix B: 82586 Parameter Example

## Appendix C: Diagnostics

Diagnostic Command Format C-2
Requirements for Testing C-3
Running the 3C505.EXE Program C-3
3C501 / 3C505 Diagnose Program Differences C-4

## Appendix D: 3D Debugger

Tile Area D-2
Menu Bar D-2
Typein Area D-3
Using the Mouse to Control the Display D-3
Function Keys D-4

## Appendix E: Software Diskette

## Appendix F: Revision 2.0 ROM

Configure Adapter Memory F-1
Timeout Values F-1
Timestamp and Timer Resolution F-2
DMA Downloading Programs F-2
Zero Offset Problem for Downloaded Programs F-2
Receive/Return Packet functions F-2
PCB Formats F-2
Interrupt Vector Services F-3
TEST Jumper Usage F-3
Configure 82586 Receive Mode F-3
Receive Packet PCB Timeout F-4
Loopback Mode F-4

## Appendix G: Revision 3.0 ROM

Adapter Selftest Command G-1
Transmit Packet Command G-1
Get Adapter Information Command G-1
Packet Processor G-1
Adapter LEDs G-1
Power On Selftest (POST) G-2

## Appendix H: Firmware Idle Loop

# Figures

1-1. EtherLink Plus Adapter Data Flow 1-6
2-1. Block Diagram 2-2
2-2. EtherLink Plus Adapter DRAM Refresh 2-7
5-1. EtherLink Plus Adapter Data Flow 5-5
D-1. The Display Just After a Stop D-2
D-2. Disassembly D-5

# Tables

4-1. PCB Command Code Summary 4-2

# Chapter 1: Introduction

## About this Document

This document is intended for use by sophisticated software engineers who will be either writing application software that will talk to the EtherLink Plus Adapter, or writing software that will actually reside on the card. The user is expected to have a strong background in microcomputer systems. It is recommended that the user also be familiar with the Intel 80186 Data Sheet and the Intel LAN Components User's Manual (they are available through Intel). The manual is divided into the following chapters:

Chapter 1: Introduction
This section is an overview of the features and capabilities of the 3C505 EtherLink Plus adapter

Chapter 2: Hardware Reference
Provides a description of the adapter architecture, system resources and functional operation.

Chapter 3: Hardware Interface
Describes the programmable registers used to control, configure, and communicate with the 3C505

Chapter 4: Command Interface
Describes the function and use of the command level interface software supplied with the card.

Appendix A: 80186 Peripheral Control Block Programming
Provides the values used in the adapter firmware to configure the 80186 internal resources.

Appendix B: 82586 Configuration
Provides the values used by the adapter firmware to configure the 82586.

Appendix C: Diagnostics
Describes the operation of the adapter diagnostic utility program.

Appendix D: 3D Debugger
Describes a host program that uses a special debug mode of the 3C505 to assist in debugging programs running on the card.

Appendix E: Software Diskette
Describes the contents of the diskette that accompanies the developer's kit.

Appendix F: Revision 2.0 ROM
Describes changes made in Revision 2.0 ROM code.

Appendix G: Revision 3.0 ROM
Describes changes made in Revision 3.0 ROM code.

Appendix H: Firmware Idle Loop
Listing of main loop of adapter firmware

## Overview

The EtherLink Plus adapter is a high-performance intelligent Ethernet adapter for IBM AT's, PC's and compatibles. This document describes the low-level programming interface to the adapter and its hardware architecture. This chapter provides an overview of its capabilities and functions. The following chapters present in-depth coverage of specific areas of the adapter's operation.

The adapter contains its own on-board 80186 microprocessor and 256 to 512KB of memory. Network packet reception and transmission is handled by an 82586 Ethernet coprocessor. The board has 16K bytes of ROM installed, which implements firmware to provide a host accessible command structure, initialization, diagnostics, packet transmission and reception, and the capability to load programs onto the board.

The host controls the function of the adapter by sending it Primary Command Blocks (PCB's). These are predefined control structures that initiate functions on the adapter, such as configuration and packet reception. The adapter, in turn, sends response PCB's back to the host. For instance the host might send one PCB to the adapter to configure its Ethernet address, then later send a PCB to request the reception of a packet that has been sent to this address.

The adapter uses some of its memory to provide buffers for holding received packets and to buffer multiple PCB commands that might be in process. The amount of memory used for these functions can be configured.

While not necessary to utilize the adapter, one of its other powerful features is the ability of the host to load a program onto the adapter and have it executed by the on-board 80186. Such a program can modify or replace parts of the default firmware to allow functions such as packet pre-processing or higher level network protocol functions.

## Programming

The following sections of this chapter provide a fairly detailed description of the main areas involved in the programming of the EtherLink Plus adapter. Some of the specifics may be hard to understand in this context, but they are covered separately, and in more detail, in later chapters.

## PCBs

The command interface between the host PC and the EtherLink Plus adapter is accomplished by the host passing defined PCB's (primary command blocks) to the adapter, and the adapter returning response PCB's to the host (if programs are run on the EtherLink Plus adapter, the adapter can also present unsolicited request PCB's to the host, i.e., download data). These PCB's are transferred using programmed I/O to or from the Adapter's Command Register port. Synchronization and control of this process is provided using the Host Control Register and host Status Register. PCB's are provided to gather information or status from the adapter, configure the adapter, initiate transmit or receive functions, pass data or programs to or from the adapter, execute programs on the adapter, and test the adapter.

Some PCB's initiate a data transfer to or from the host in the course of their processing, and the host must be prepared to handle the data transfer through the Data Register port at the appropriate time. This is usually accomplished by the host setting up its DMA to transfer data to or from the Adapter's data port.

When sending PCB's to the adapter, the host should monitor the Host Status Register port for the HCRE bit (Host Command Register Empty) before writing a byte in the Command Register. The host can monitor for response PCB's by polling the Host Status Register port for the ACRF bit (Adapter Command Register Full), then reading the Command Register. Alternately, the host can enable command interrupts from the adapter with the CMDE bit in the Host Control Register. If this bit is set, the adapter will interrupt the host when it fills the command register, in the process of sending a response PCB to the host.

The PCB interface is presented in detail in chapter 3.

## Interrupts

The host can be interrupted by the adapter in two cases:

- When the Command Register is filled by the adapter (PCB response or request).
- When the host DMA reaches terminal count (DMA done).

Each of these interrupts has an enabling bit in the Host Control Register (CMDE and TCEN). Additionally, the DMA Done interrupt assumes DMA has also been enabled with the DMAE bit.

If programs are written to be downloaded and run on the EtherLink Plus adapter, several hardware interrupts are available to the 80186 on the adapter:

- DMA Channel 1 Done. When the onboard DMA reaches terminal count.
- Timer. Every 10 ms.
- Command Register Full. When the host writes into the Command Register.
- 82586 Interrupt. On receive and others (see Intel Microcommunications Handbook).

- Attention (NMI). When the Attention bit in the Host Control Register is set by the host. This is used to initiate a reset.

The interrupt functions are covered in Chapter 1.

### Data Transfer and DMA

Data (other than commands) is passed to or from the adapter through the Data Register port. Data transfers are normally initiated in conjunction with a particular type of PCB process. The host (and EtherLink Plus adapter) are expected to know when data transfers are required and perform the I/O at the appropriate time. The host controls the direction of transfer using the DIR bit in the Host Control Register.

The host can perform data transfer by polling the Host Status Register for the status of the HRDY bit then reading or writing the Data Register port. If DMA transfer is desired, the host needs to initialize its DMA with address, length, direction, etc., then set the Host Control Register to specify direction and DMA Enable. If a DMA Done interrupt is desired, that bit should also be set.

The adapter uses its own onboard DMA in a similar fashion to transfer data between adapter memory and the Data Register port, thus completing the adapter side of the total transfer.

DMA and data transfer functions are described in chapter 1, with register descriptions in chapter 2, and PCB DMA requirements in Chapter 3.

### ROM Utilities

The EtherLink Plus adapter is equipped with a ROM that provides extensive functions for handling the 82586, managing packet buffer and processing queues, gathering information, configuring adapter parameters, and communicating with the host. Many of these functions are available to the host via the PCB interface. Additionally, the host can initiate (with PCB's) a download of a program to the adapter and have it execute on the adapter. A program running in this manner can access or modify many more of the EtherLink Plus adapter functions by respectively calling or replacing software interrupts on the board.

The default resident ROM program on the adapter uses these same interrupts to initiate processing for PCB's, packet management, data transfers, configuration, and so forth. The interrupt functions available on the adapter are described, in detail, in chapter 3. As a guide to how these functions are utilized by the ROM, and how the normal adapter processing is implemented, Appendix H contains a listing of the code of the main processing loop of the firmware on the EtherLink Plus adapter. The listing shows which ROM routines are called by the loop to initiate processing of packets and commands. The comments in the section on the command (PCB) interrupt handler show how processing is initiated for each type of PCB.

### Data Structure

Some of the memory on the adapter is configured to provide queues for received packets and some to queue PCB's from the host. On an EtherLink Plus adapter with default configuration, some memory is free and can be used to allocate more packet or PCB buffers, or to load programs.

Figure 1-1, EtherLink Plus Adapter Data Flow, shows the main data structures and data flow, along with the ROM utility software interrupts, that are used by the firmware to control processing. This figure should help in understanding the discussion that follows. On the figure, the two dashed boxes labeled Downloaded Program represent likely places for a program to take control of the EtherLink Plus adapter packet processing. (Indeed, these functions are implemented in a download program as part of the demo program on the software diskette. See Appendix E). If no program is loaded, the flow will pass directly through these boxes.

The data structures for receiving packets are organized into three queues of receive buffers. All the receive buffers are 1.6 kb in size and can, hold an entire packet. Initially, all receive buffers will be located on the Free List queue of unallocated packet buffers. As the 82586 receives packets and interrupts the processor, packets are moved to the Receive List queue of unprocessed packets. The firmware's main processing loop examines this queue for new packets via INT 81 - 2. If new packets are found, they are processed with INT 86 - 2. If a receive PCB is outstanding, the packet will be sent to the host with INT 80 - 4 and INT 80 - 7, otherwise, INT 86 - 2 enqueues the received packet on the rcvPkt queue. When a receive PCB is available, INT 86 - 3 will dequeue the packet and send it to the host with INT 80 - 4 and INT 80 - 7. After sending the packet the receive buffer is returned to the Free List with INT 81 - 3.

The number of receive buffers (that make up the contents of the 3 queues), and, therefore, the number of packets that can be buffered on the adapter, is a configurable parameter on the adapter. Receive buffers have Frame Descriptors and Receive Buffer Descriptors associated with them (see Intel 82586 documentation for descriptions). The number of Frame Descriptors is also configurable and is normally set equal to the number of receive buffers.

PCB commands are held in two additional queues. When a command is received on the adapter it is either processed immediately or placed in one of two queues for processing. If the PCB is a receive command, it is placed in the RcvPCB queue with INT 86 - 1, otherwise it is placed in the command PCB queue with INT 88. The size of these two queues are individually configurable. Each of the command queue entries is several bytes larger than a PCB (64 bytes).

A few added notes on the somewhat complicated process of handling PCB's are probably in order. When a PCB is received by the adapter, control is passed to the PCB pre-processor, either by a command interrupt or via a direct call to the interrupt handler from the main loop. A few PCB's are processed immediately. Most are enqueued on the PCB queue with INT 88. Receive PCB's are enqueued on their own rcvPCB queue with INT 86 - 1. The action performed for a particular PCB is listed in the comments of the main loop listing (Appendix H). If a PCB is received with a command code greater than those defined in the documentation, the PCB will be enqueued on the PCB queue via INT 88 and later discarded when INT 85 is called to process packets.

Packet transmission is implemented with a single pre-allocated packet buffer (not shown on the figure).

The downloaded program, shown on the figure, implements a packet filter operating on the EtherLink Plus adapter. In one section, the program has inserted itself on the entry to the INT 85 PCB processor. This allows the program to receive control PCB's from the host. New PCB commands are defined for the program, and if it sees one of these PCB's it processes the PCB according to the definition. If the program finds a PCB other than its own, it passes the PCB on to the standard INT 85 PCB processor.

The other section of the downloaded program is the packet filter. The program has inserted itself on the entry to INT 86 - 2. Here it will receive a pointer to the receive buffer for every packet that is received. The program can examine the contents of the packet and has the option to discard or process the packet. If the packet is to be discarded, the program calls INT 81 - 3 to free the packet buffer, then returns to the caller of INT 86 - 2. If the packet is to be processed normally, the program passes control to the normal INT 86 -2 routine. In this case the program could also modify the packet before passing it on.

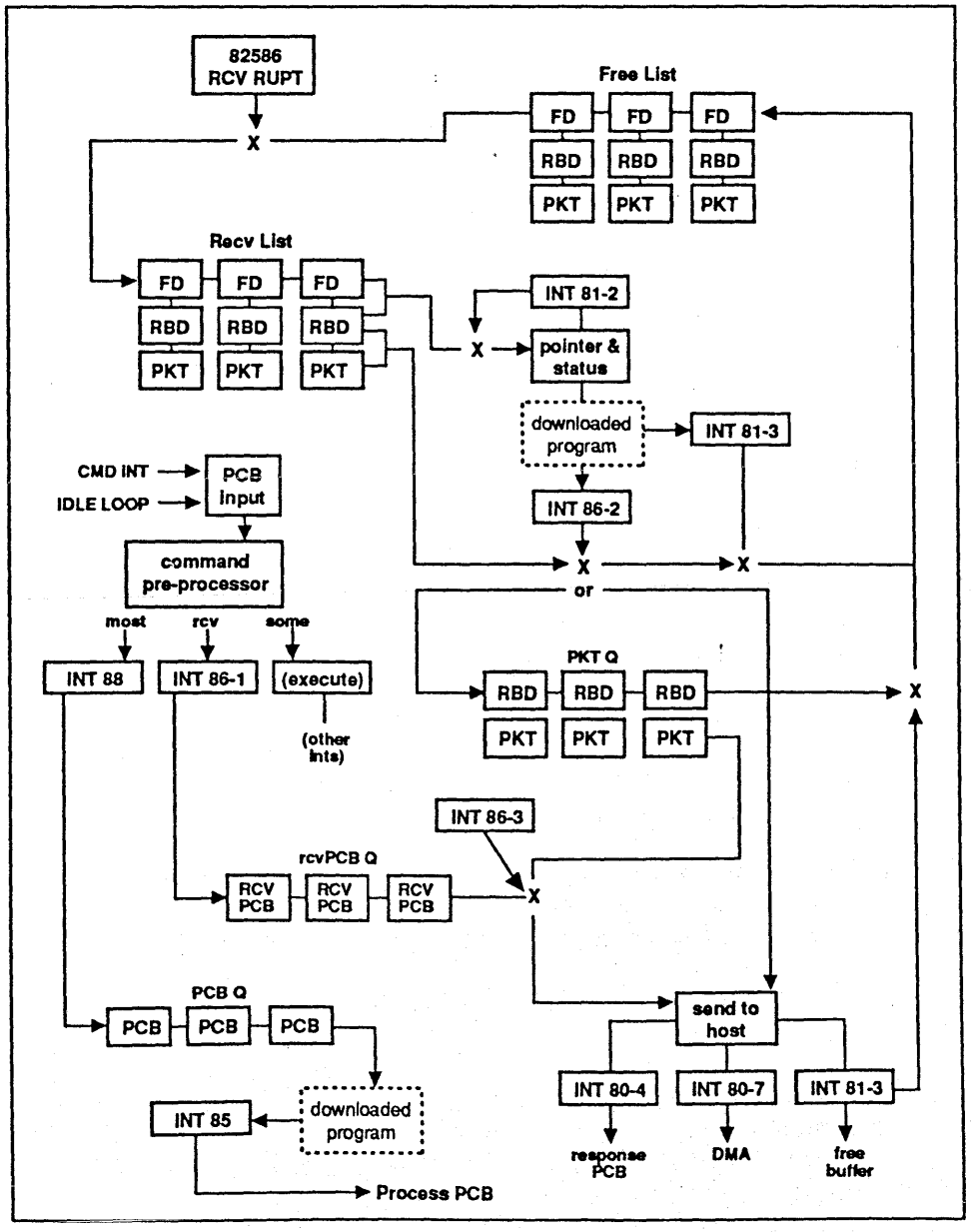



Figure 1-1. EtherLink Plus Adapter Data Flow

# Chapter 2: Hardware Reference

## Introduction

The EtherLink Plus adapter is a high performance Ethernet adapter for IBM PCs and compatibles. It consists of an 80186 16 bit microprocessor, an 82586 Ethernet coprocessor, up to 512KB of user RAM, a high speed 16 bit host interface, and a highly integrated on-board transceiver. The EtherLink Plus adapter is particularly well-suited for server and high performance workstation applications.

- **Resources**

  - 8 Mhz 80186 16 bit microprocessor - no wait states
  - 82586 multi-packet buffer Ethernet coprocessor
  - 16KB to 128KB EPROM
  - 128KB to 512KB packet buffer/program memory
  - 8/16 bit host interface - PIO or DMA
  - 20 byte FIFO to maximize host/adapter data transfer
  - On-board "Thin Ethernet" transceiver/802.3 connector
  - 8KB host EPROM

- **Architecture**

  The EtherLink Plus adapter is a 16 bit microcomputer with a high performance Ethernet I/O channel and an IBM PC AT interface. The 16KB of on-board firmware contain software that supports initialization, program download, and diagnostic software. The 256KB RAM, expandable to 512KB, allows for protocol processing as well as offloading of application programs from the host PC. The 82586 performs all Data Link functions, as well as powerful network diagnostics. It performs all packet buffer management functions and, in a typical environment, will not "drop" packets. The host interface supports high speed, 8 or 16 bit, DMA transfers as well as programmed I/O. The interface is very flexible, yet simple, allowing for easy programming.

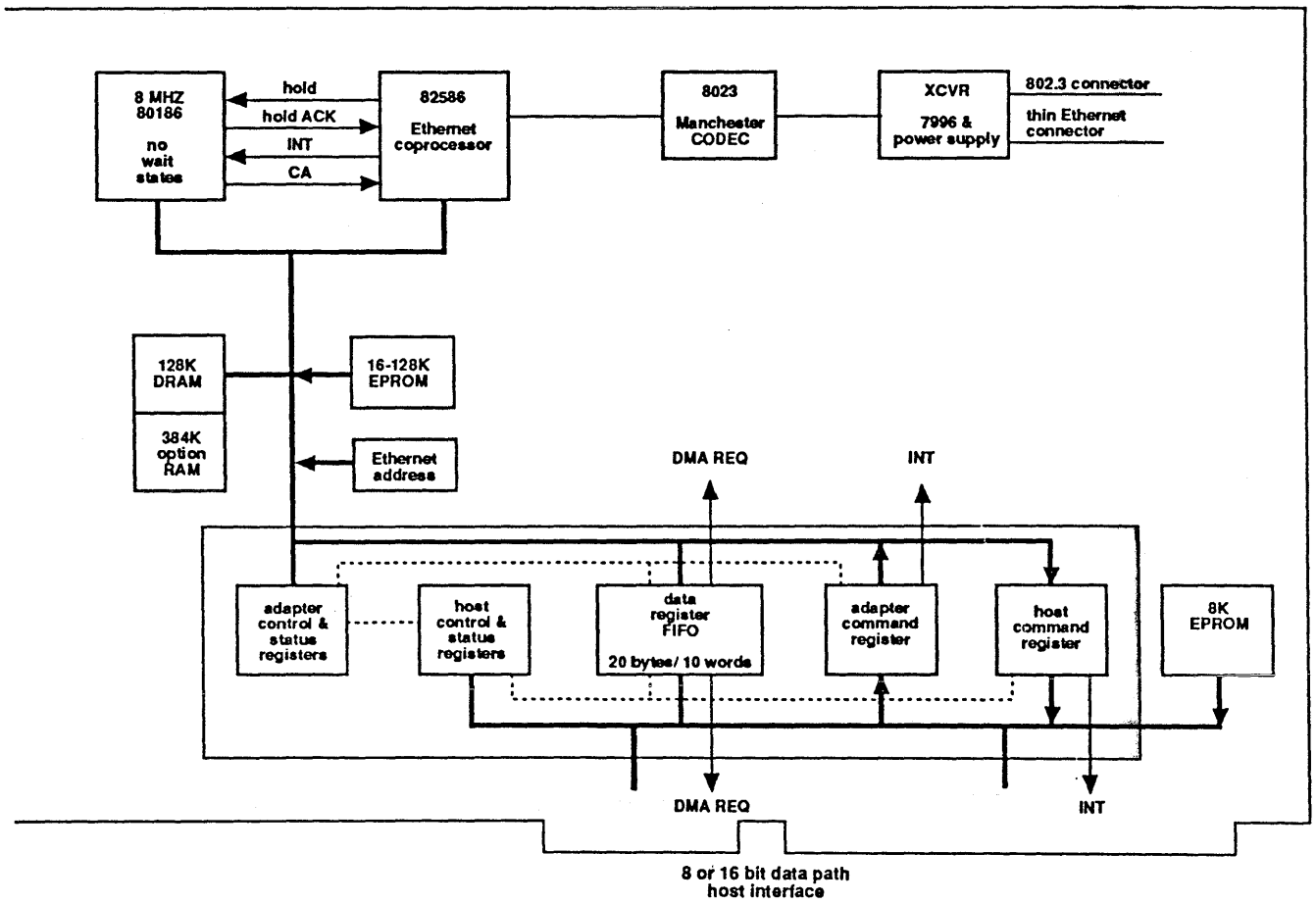



Figure 2-1. Block Diagram

# Address Maps

## Adapter I/O Map: 80186 On-board I/O Ports

| HEX Address | Byte/Word | Description |
|---|---|---|
| 0 | NA (see text) | 82586 Channel Attention |
| 100 | Low Byte | Adapter Command Register |
| 102 | Low Byte - Read | Adapter Control Register |
| 102 | High Byte - Read | Adapter Status Register |
| 102 | High Byte - Write | Adapter Control Register |
| 104 | Word | Data Register |
| 180 - 18F | Low Byte | Station address (6 Bytes) |
| FF00 - FFFF | Word | Peripheral Control Block |

## Adapter Memory Map

| HEX Address | Description |
|---|---|
| 00000-1FFFF | 128KB system RAM: Bank 1 |
| 20000-3FFFF | 128KB system RAM: Bank 2 |
| 40000-5FFFF | 128KB option RAM: Bank 3 |
| 60000-7FFFF | 128KB option RAM: Bank 4 |
| FC000-FFFFF | 16KB system ROM * |
| E0000-FFFFF | 128KB system ROM (If 27512s are installed) |

* Address lines A20-A23 of the 82586 are ignored and the Initialization Root is located at FFFF6 in system ROM.

## Host I/O Map: Adapter's external I/O ports

| HEX Address (factory set) ** | Description |
|---|---|
| Base address + 0 (300) R/W | Host Command Register |
| Base address + 2 (302) Read | Host Status Register |
| Base address + 2 (302) Write | Aux DMA Register |
| Base address + 4 (304) R/W | Data Register *** |
| Base address + 6 (306) R/W **** | Host Control Register |

** The address is given as an offset from the I/O base address which is set using the I/O address jumpers on the card. The factory set base address is 300H.

*** The Data Register is a byte wide register in an 8 bit slot (PC, XT, or AT) and word wide in a 16 bit slot (AT).

**** Host Control Register is Write Only on Rev 2 H/W. (Rev 3 H/W has large gate array chip).

# 80186 Microprocessor

The EtherLink Plus adapter uses the Intel 80186 Microprocessor. This is a highly integrated 16 bit processor with 3 timers, 2 DMA channels, and an interrupt controller on chip. The 80186 is software compatible with the 8086.

The 80186 timing is generated by a 16Mhz crystal. An internal divider generates an 8Mhz clock output which is used for system timing. All 80186 bus cycles are 4 clock cycles long, or 500 nanoseconds, with a system memory bandwidth of 2 Megawords per second. All DMA transfers require 2 bus cycles, or 1 microsecond.

# 82586 Ethernet Coprocessor

The 82586 is a high performance, intelligent communications processor responsible for all network related tasks, including frame reception and transmission, error logging, and diagnostics.

The 82586 has two interfaces: a parallel system bus interface to communicate with the 80186 and to retrieve and store packet data in system RAM; and a serial interface to transmit and receive data from the network. The serial interface is described in theNetwork Interface section. The 82586 bus interface operates from the 8 Mhz system clock and all bus cycles are 500 nanoseconds.

The 80186 and the 82586 operate in a shared bus configuration using the HOLD/HOLDA protocol. This configuration is described in detail in the 82586 sections of the Intel Microcommunications Handbook. In this mode, only one of the processors can use the system bus at a time. All interprocessor communications are via the system RAM. The 80186 can initiate a transaction by asserting the CA (Channel Attention) input to the 82586. A read or write to I/O location 00 will cause an active transition on the CA input. The 82586 initiates a transaction by asserting the 80186 INT1 input.

The 82586 can require the bus to access system RAM in three instances:

1. To read or update the SCB (System Control Block).

2. To transmit a packet.

3. To receive a packet.

When receiving or transmitting, the 82586 uses approximately 35% of the system bandwidth, or 715 KW/second. Thus program execution and DMA transfers, although slowed, do not halt.

The adapter CPU(80186) can reset the 82586 by asserting the R586 bit in the Adapter Control Register. The 82586 remains in the reset state until this bit is cleared.

# Network Interface

The EtherLink Plus adapter network interface consists of the serial interface on the Intel 82586 LAN controller, the SEEQ 8023 Manchester Code Converter, and an on-board transceiver using the AMD 7996 Transceiver IC.

## 82586 Serial Interface

The 82586 performs all parallel to serial and serial to parallel conversion during packet transmission and reception. During transmit, parallel data is retrieved from the adapter RAM through the 82586 bus interface. The 82586 serializes the data, inserts the preamble, source and destination fields, appends a CRC field to the "packet", and outputs the bit stream. The 82586 also performs the CSMA/CD link management algorithm according to the IEEE 802.3 standard. During reception, the 82586 strips off the preamble and compares the destination address field with the station address to see if the frame should be received. If so, the serial bit stream is converted into bytes and stored in the adapter RAM.

## 8023 Manchester Converter

The 8023 is responsible for the Manchester encoding and decoding of the serial bit stream between the 82586 and the transceiver. It also supplies the transmit and receive clocks to the 82586 serial interface. A watchdog timer on the IC prevents continuous transmission of more than 25 milliseconds, thus limiting the maximum packet size to approximately 31KB.

For diagnostic purposes, the 8023 can be placed in "loopback mode" whereby the transmitted data is internally routed to the receive section. This is useful for isolating transceiver problems. Enable loopback by clearing the Loopback bit in the Adapter Control Register. Refer to Chapter 3, Hardware Interface Specification,

## Transceiver

The EtherLink Plus adapter onboard transceiver physically connects the adapter to the "Thin Ethernet" coax cable. It performs the necessary signal conditioning as well as collision detection.

The EtherLink Plus adapter can also be connected to a standard Ethernet network through an external transceiver such as the 3Com 3C102. If so, the user must disable the onboard transceiver and enable the 15 pin connecter on the backplate of the card. To do so, the transceiver select jumper on the card must be moved from the BNC position to the DIX position. The EtherLink Plus Adapter Installation Guide, included with the adapter, illustrates this procedure.

# Adapter Firmware ROM

The EtherLink Plus adapter contains 16KB of firmware contained in two 8KBx8, 2764 type ROMs. These ROMs can be replaced by 27128, 27256, or 27512 ROMs for up to 128KB of firmware. The ROMs must have a maximum address access time of 250 nanoseconds or less.

The EtherLink Plus adapter ROM firmware performs self-test, initialization and configuration, and DRAM refresh. It also provides, through a command block interface, a set of functions which support Host/Adapter I/O, network interfacing and execution of downloaded programs. Refer to Chapter 4, Command Interface Specification, for more details.

The system ROM is mapped to address space FC000H-FFFFFH (E0000H-FFFFFH if 128KB are used) and is accessible to both the 80186 and 82586. The 82586 only accesses ROM following an 82586 reset to fetch the initialization root.

# Adapter RAM

The EtherLink Plus adapter contains 128KB (older version) or 256KB of dynamic memory organized in a 64KB x 16 configuration. Two or three (depending on version) additional 128KB banks can be installed for 256KB, 384KB, or 512KB of RAM memory. Each bank consists of four 64KB x 4 DRAMs. The first additional bank must be installed in socketed locations U31, U33, U35, and U37. The second additional bank must be soldered into locations U40, U42, U44, and U46. The third additional bank must be soldered into locations U41, U43, U45, and U47. These devices must have a maximum RAS access time of 150 nanoseconds and maximum CAS access time of 75 nanoseconds. In addition, these RAMs must support "CAS before RAS refresh", described below. These parts are currently available from Micron, NEC, Fujitsu, and Texas Instruments.

The system RAM is accessible to the 80186 and 82586 and is used for both packet buffering and program storage. No physical partitioning or protection mechanism is used. The RAM is mapped into the adapter memory space 0-7FFFFH, with bank 1 occupying 0-1FFFFH, bank 2 occupying 20000-3FFFFH, bank 3 occupying 40000-5FFFFH, and bank 4 occupying 60000-7FFFFH.

Software must perform two functions for proper RAM operation: initialization and refresh. To refresh the RAM, 256 consecutive locations in each bank must be accessed every 4 milliseconds. Data loss will occur if refresh is not performed. The initialization procedure depends on the refresh technique used.

To facilitate refresh, the EtherLink Plus adapter contains hardware which utilizes the "CAS before RAS" refresh feature of the DRAMs. In this mode, the RAMs generate the refresh address internally after each CAS before RAS cycle, and the internal address counter increments so that the next CAS before RAS cycle will refresh the next address. A read or write to I/O location 80H will produce a CAS before RAS cycle in all banks simultaneously. The 80186 PCS1 Peripheral Chip Select output is programmed for this range. A CAS before RAS cycle, read or write, does not modify RAM data.

To increase reliability and to free the 80186 from involvement in RAM refresh, the EtherLink Plus adapter firmware uses 80186 Timer 2 and DMA Channel 0 to automatically generate refresh cycles. The timer causes a DMA cycle to occur every 30 microseconds. Each DMA cycle performs an I/O read and write to location 80H. Thus each DMA cycle refreshes two memory locations. The DMA controller is not programmed to "stop on terminal count" so that refresh, once initialized, will continue without any CPU involvement. Using this technique, refresh consumes 3.3% of the memory bandwidth. The timer generated DMA will only produce one DMA cycle so that burst mode refresh cannot be used.

Upon power-up, the 80186 must wait 200 microseconds and then perform 8 RAM "initialization" cycles. If CAS before RAS refresh is to be used, then 8 refresh cycles (a read or write to I/O location 80H) will initialize all RAM. If CAS before RAS refresh is not used, then 8 reads or writes to any location in each of the installed banks of memory will initialize the RAM. Refresh should begin immediately after initializing the RAM.

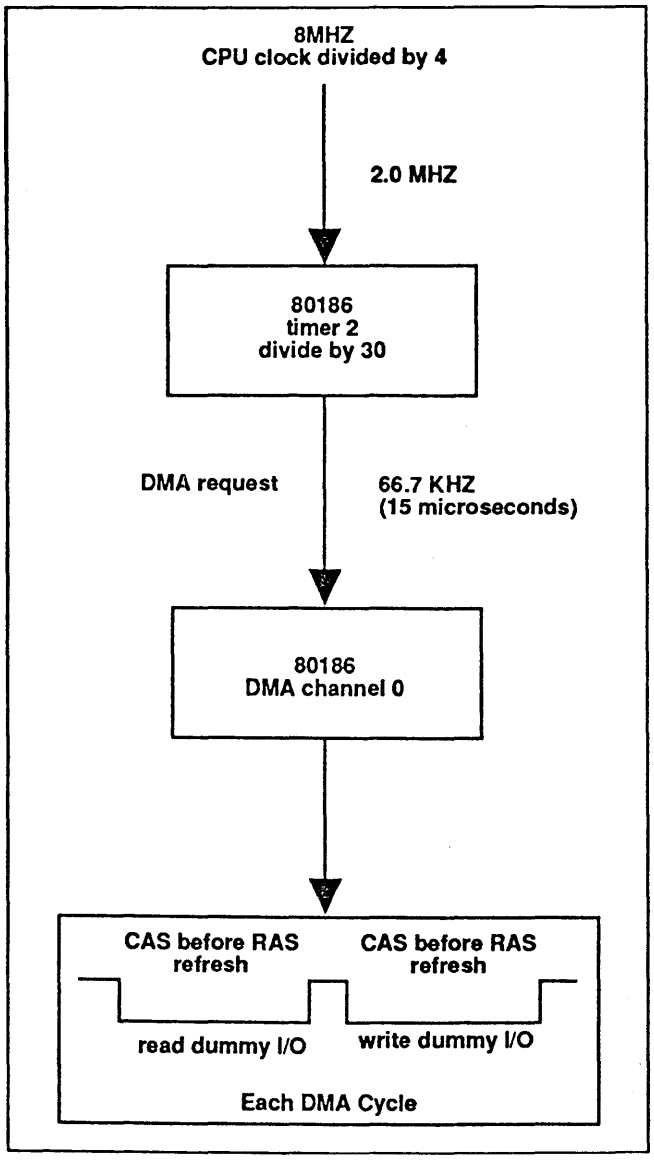



Figure 2-2. EtherLink Plus Adapter DRAM Refresh

# Host-Adapter Interface

The host and the adapter communicate through two I/O mapped registers: the Command Register and the Data Register. In addition, each side has a Control Register and a Status Register that are used for transfer handshaking and interface configuring. A detailed bit level description of these registers is found in Chapter 3, Hardware Interface. The interface requires 16 locations in the host I/O address space. Jumpers are used to position the base address.

## Command Register

The Command Register is a full duplex byte-wide register used to transfer commands and small amounts of data between the host and the adapter. The register can be polled using the Command Register Empty (ACRE and HCRE) and Command Register Full (ACRF and HCRF) bits in the Host and Adapter Status Registers. Alternately, the Command Register can be interrupt driven, so that an interrupt is generated to the host or adapter when the opposing side has loaded a byte into the Command Register. Refer to the Adapter Interrupts section of this chapter for more information.

## Data Register

The Data Register is a half duplex 20 byte FIFO designed for high speed bulk data transfers between the host and the adapter. The direction of the data transfer is controlled by the DIR bit in the host Control Register. If the DIR bit is cleared (0), data transfer is from the host to the adapter, which is referred to as a data download. If the DIR bit is set (1), data transfer is from the adapter to the host and referred to as an upload. The state of the DIR bit can be read in both the Host and Adapter Status Registers.

The Data Register supports both polled I/O and DMA data transfers. In polled operation, the state of the Data Register can be determined by reading the Data Register Ready bit (HRDY and ARDY) in the Host and Adapter Status registers. The meaning of the Ready Bit is determined by its state and the state of the DIR Bit.

| Transfer | Dir | Hrdy | Ardy | Description |
|---|---|---|---|---|
| PIO Download | 0 | 1 | X | Register not full |
| | | X | 1 | Register not empty |
| PIO Upload | 1 | X | 1 | Register not full |
| | | 1 | X | Register not empty |
| Register not full: Write data<br>Register not empty: Read data | | | | |

To clear a stuck byte from the Data Register (see next section), or to ensure that the register is in a known empty state, the FLSH (Flush) bit in the Host and Adapter Control Register is used. By setting and resetting the FLSH Bit, the Data Register Ready Flag is forced to the empty state (the data in the FIFO is not actually cleared). Either the host or the adapter can use this bit, regardless of the state of the DIR Bit.

Careful attention should be paid in the use of the DIR bit. Incorrect and confusing results occur if the bit is not set correctly. The DIR must be in its correct state prior to enabling DMA transfers. When changing the state of the DIR bit from download to upload, the host must make sure that the adapter has actually completed the download, i.e., the FIFO is empty. One solution is to change the DIR bit only as part of the command block sequence. The adapter firmware changes the DIR bit after the adapter has accepted the first word of a command block. This indicates that the adapter has completed execution of the last command block.

## Data Register Configuration

To the adapter, the Data Register is always a 16 bit wide FIFO, 10 words deep. Only 16 bit data transfers are permitted (A0 and BHE are ignored). However, to the host, the Data Register is configured as either an 8 bit FIFO, 20 bytes deep, or a 16 bit FIFO, 10 words deep, depending on where it is installed. The register is automatically configured and no jumpers need be set. Also, the adapter does not need to know whether it is installed in an 8 or a 16 bit slot.

The Data Register is configured as a 16 bit register when installed in a 16 bit I/O slot of an AT. Only word transfers are permitted (A0 and BHE are ignored) and only 16 bit AT DMA channels (5, 6, 7) can be used.

In a PC, XT, or an 8 bit slot of an AT, the Data Register is configured as a 20 byte FIFO to the host. The register performs byte to word conversion so that the 80186 always performs word I/O to the Data Register and adapter performance is not reduced in 8 bit systems. The host must always transfer an even number of bytes to the register; the Adapter Data Register Ready flag (ARDY) indicates the presence of words, not bytes. An odd byte will get "stuck" in the register because the adapter will not know of its presence. In adapter to host transfers, word to byte conversion is performed. A byte cannot get stuck in this direction because the Host Data Register Ready flag (HRDY) indicates the presence of bytes.

## DMA Transfer

DMA transfers by the host to and from the Data Register are enabled using the DMAE bit in the Host Control Register. Since the DMA channel floats when this bit is cleared, caution should be taken to ensure that this channel in the PC DMA controller is not enabled until the DMAE bit is set. When the DMAE bit is cleared, another I/O card may use the same DMA channel.

| Transfer | Dir | Hrdy | Ardy | Description |
|---|---|---|---|---|
| DMA Download | 0 | 1 | X | Write request to host |
| | | X | 1 | Read request to adapter |
| DMA Upload | 1 | X | 1 | Write request to host |
| | | 1 | X | Read request to adapter |

The EtherLink Plus adapter can be programmed to generate an interrupt to the host after the last cycle of a host DMA transfer using the TCEN bit in the Host Control Register. Refer to the section on interrupts for more information.

The adapter performs DMA transfers to and from the Data Register on 80186 DMA Channel 1. Both the DMA Channel Enable and the DMA DONE interrupt are controlled by programming registers internal to the 80186. The DMA Channel 1 input to the 80186 is never in a floating state.

Note: the adapter and the host may perform DMA transfers independent of one another. That is, one may use polled I/O while the other performs DMA. There is little reason for the adapter to use polled I/O and the EtherLink Plus adapter firmware always uses DMA.

The EtherLink Plus adapter contains hardware to support host "demand mode" DMA transfers in PCs where this mode is supported. If the Burst (BRST) bit in the Aux DMA Register is not set, the EtherLink Plus adapter will transfer 9 bytes/words and then relinquish the DMA channel for one host CPU cycle. This will allow the host to refresh its own system DRAM if necessary. The EtherLink Plus adapter will then transfer another 9 bytes/words, and so on. If the Burst bit is set, this pause will not occur. The Burst bit has no effect if single cycle DMA is used. Thus, if the DMAE bit is set, the DMA request signal to the host PC remains active until one of the following conditions is met:

1. The entire host DMA transfer is completed

2. The Data Register FIFO is temporarily full/empty, depending on the transfer direction.

3. The Burst bit is not set and 9 DMA transfers have occurred since the last DMA pause.

## Status Flags

The host and adapter also communicate using general purpose Status Flags. The adapter has three flags, ASF1, ASF2, and ASF3, which are programmed by the Adapter Control Register and directly observable by the Host Status Register. The host has two Status Flags, HSF1 and HSF2, which are programmed by the Host Control Register and observable through the Adapter Status Register. The Status Flags are used between the adapter firmware and the host for command synchronization, completion codes, and other assorted tasks. They are not decoded by the hardware in any way.

# Adapter (80186) Interrupts

The 80186 microprocessor in the EtherLink Plus adapter may be interrupted by both internal and external sources.

## Internal Interrupts

Internal interrupts are interrupts generated by the 80186 itself. These include Timer, DMA, and software generated interrupts. Refer to the 80186 Data Sheet for programming details. A brief description of the internal hardware interrupts follows.

### DMA Channel 1 Done Interrupt

This is used to generate an interrupt after the last cycle of a DMA transfer on the adapter, to or from the Data Register.

### Timer Interrupt

An interrupt is generated every 10 milliseconds from Timer 0. The interrupt is used for general purpose counting and timeouts.

## External Interrupts

There are three sources of external interrupts to the 80186: Command Register Full (INT 0), 82586 Int (INT1), and Attention (NMI). Since each interrupt has a unique channel, there is no need for a corresponding status bit to indicate the cause of the interrupt. Except for NMI, which cannot be disabled, the interrupts are enabled or disabled by setting the appropriate bit in the Interrupt Mask Register in the 80186. These inputs never "float" so that these channels can be enabled at any time. All channels are programmed positive edge triggered.

### Command Register Full (INT0)

If enabled, an interrupt will be generated to the 80186 when the host loads a byte in the command register. This condition is also reflected by the HCRF (Host Command Register Full) bit in the adapter status register. The Command Register Full interrupt and status bit are cleared when the 80186 reads the byte from the Command Register.

### 82586 INT (INT1)

This input is tied directly to the INT output on the 82586. If enabled, the 80186 will be interrupted by the 82586 after the SCB has been modified by the 82586. Refer to the 82586 data sheet for more information.

### Attention (NMI)

When the ATTN bit is set in the Host Control Register, an NMI is generated in the 80186. This NMI is used as a "soft" reset to bring the adapter back to a known state after an interface error occurs. The NMI is positive edge triggered and the ATTN bit must be brought from low to high to force the NMI.

## Host Interrupts

The EtherLink Plus adapter can be programmed to interrupt the host in two situations: DMA complete and Command Register Full. Only one PC interrupt channel is used.

### Host DMA Done

By setting the TCEN (Terminal Count Enable) bit in the Host Control Register, an interrupt will be generated to the host after the last cycle of a DMA transfer to or from the Data Register. If the Command Register Full interrupt is also enabled, the Done bit in the host status register should be used to determine if a DMA Done was the source of the interrupt. The DMA Done interrupt and Done status bit are cleared by disabling the DMA channel using the DMAE (DMA enable) bit in the Host Control Register.

### Command Register Full

By setting the CMDE (Command Enable) bit in the Host Control Register, an interrupt will be generated to the host when the adapter writes a byte in the Command Register. If the DMA Done interrupt is also enabled, the ACRF (Adapter Command Register Full) bit in the Host Status Register should be used to determine the source of the interrupt. The Command Register Full interrupt and ACRF bit are cleared when the host reads the byte from the Command Register.

When installed in a PC, XT, or 8-bit AT slot, interrupt channels 3, 4, 5, 6, 7 or 9 should be used. In this situation, channel 9 is equivalent to channel 2. In a 16-bit AT slot, any interrupt channel can be used.

NOTE: Care must be taken when enabling and disabling the EtherLink Plus adapter interrupts. If both interrupt sources are disabled, the interrupt channel is floated and can cause spurious interrupts if the PC PIC channel is not also disabled. To prevent this, always mask the PIC channel off before disabling both EtherLink Plus adapter interrupts, and enable EtherLink Plus adapter interrupts before enabling the PIC channel. When both EtherLink Plus adapter interrupts are disabled, the interrupt channel can be used by another I/O card.

If both interrupts are to be handled, the interrupt service routines must guarantee that the Adapter's request signal goes inactive sometime after the EOI is issued to the Host's interrupt controller. PC and AT type machines used edge triggering mode on the Intel 8259 PIC. In this mode the Interrupt Request signal must go inactive sometime after the EOI is issued or the channel will not be re-armed. If both adapter interrupts are enabled it is possible to have a case where while handling one interrupt type, the other interrupt type occurs and holds the Adapter's Interrupt Request signal active. The ISR must check for this and in some way cause the request to go inactive after the EOI is issued or interrupts may be lost.

# Resetting the Adapter

### Power On Reset

Upon power up, the EtherLink Plus adapter is put in a reset state. Both the 80186 and 82586 are reset, the Command and Data Registers status indicate empty, and both the Host and Adapter Control Registers are cleared.

### Hard Adapter Reset

The host can reset the adapter by simultaneously setting both the ATTN bit and the FLSH bit in the Host Control Register. This reset is similar to the power on reset except that the Host Control Register is not affected. The adapter will remain reset until the ATTN and FLSH bits are reset.

NOTE: After either of the above "hard" resets, the adapter firmware performs configuration and self-test routines which last several seconds. The completion of these tasks is indicated by a transition in the Host Status Flags from state 3 to state 0. Visually, this is indicated by LED #1 turning off. LED1 is closest to the front of the computer. LED2 is closest to the back.

### Soft Adapter Reset

By setting only the ATTN bit of the Host Control Register, the host can initiate a soft reset of the adapter. This reset causes the adapter firmware to clear the Command Register and any commands that are queued on the adapter, flush all packet buffers and queues, and stop any DMA transfers. The soft reset does not perform configuration or self-test functions, so does not incur the several second delay of a hard reset.

# Station Address

The Adapter Station address resides in a PROM in the adapter I/O space. The twelve digits are contained in the low byte of the six consecutive words starting at location 180H.

# LED Indicators

The adapter contains two LEDs which are enabled by the LED1 and LED2 bits in the Adapter Control Register. The LEDs are active high so that setting the bit turns the LED on and clearing the bit turns the LED off.

**LED #1**
The EtherLink Plus adapter firmware turns this LED on during the self test and initialization following a hard reset. The LED is turned off at the conclusion of these routines. Application software may call EtherLink Plus adapter firmware routines to use the LED for debug and status indications.

**LED #2**
The EtherLink Plus adapter firmware turns this LED on or off at approximately a 1 HZ rate. It serves as a "heartbeat" signal and is a visual indicator that the card is alive. If the LED should stop blinking, a software or hardware error has occurred. It is not recommended that application software use this LED.

---

⚠ **CAUTION:** When using EtherLink Plus adapter firmware, downloaded software must control the LEDs by calls to the firmware routines provided in ROM. Otherwise incorrect operation will result.

---

# Host ROM

A socket is provided on the card for an 8KB x 8 (2764) ROM which resides in the host memory space. This ROM can be used for applications such as BIOS extensions. The maximum address access time for these devices must not be greater than 250 nanoseconds. The ROM is only accessible to the host.

**NOTE:** The PC AT will execute from 8 bit ROMs on I/O cards.

The ROM can be mapped on any 8KB boundary in the host address space. The base address for the ROM is programmed using the memory address jumpers on the card. The EtherLink Plus adapter does not support DMA access to this ROM; incorrect data will be read.

To enable the ROM, set the Enable jumper on the card to the ON (0) position. If no ROM is present, or to disable a ROM that is present, place the jumper in the off (1) position.

# Chapter 3: Hardware Interface

## Introduction

The EtherLink Plus Adapter Hardware Interface Specification describes in detail the EtherLink Plus adapter interface registers accessible by the PC host and the EtherLink Plus adapter processor.

Briefly, the host and EtherLink Plus adapter communicate using four registers: Command, Data, Status, and Control. The Command Register is full duplex and used for command block transfers. The Data Register is a half duplex, 16-bit wide FIFO, and can be used with a DMA channel for efficient bulk data transfer. The Control Register allows programmed configuration of the interface. The Status Register contains interface state flags and programmable flags. The host and adapter access these registers in their I/O space relative to a base I/O address:

| Register | Base Offsets Host | Base Offsets Adapter | Access |
|---|---|---|---|
| Command | 0 | 0 | Read/Write |
| Data | 4 | 4 | Read/Write |
| Status | 2 | 3 | Read only |
| Control | 6 | 3 | Write only |
| Control | 6 | 2 | Read only * |
| AUX DMA | 2 | X | Write only |

* The host and adapter will read the contents of their own Control Registers.

The host base I/O address can be modified with jumpers, while the adapter base address is fixed at 100 hex.

Refer to Chapter 2 for a more detailed explanation of the hardware architecture. Refer to Chapter 4, Command Interface Specification, for a description of how this interface can be programmed.

## Command Register

| CMD7 | CMD6 | CMD5 | CMD4 | CMD3 | CMD2 | CMD1 | CMD0 |
|---|---|---|---|---|---|---|---|

The Command Register (CMDR) is a bidirectional 8-bit data register used for passage of primary command blocks between the host and the EtherLink Plus adapter. Programmed and interrupt driven I/O can be used to read/write this register; DMA is not supported.

## Data Register

| DR15 | DR14 | DR13 | .... | .... | DR2 | DR1 | DR0 |
|---|---|---|---|---|---|---|---|

The Data Register (DR) is a half duplex, 20 byte FIFO used for high speed data transfers. DMA or programmed I/O methods can be used to read/write this register; interrupt driven I/O is not supported. From an 8-bit host, the Data Register appears as an 8-bit wide register. Only an even number of bytes can be transferred. To the adapter or to a 16-bit host, the register appears as a 16-bit wide register (10 words deep) and only word transfers are supported.

## Host Control Register

The Host Control Register (HCR) is an 8-bit register used by the host to cause EtherLink Plus adapter hard or soft resets, to control interrupt and DMA requests to the host, and to provide synchronization control signals between the PC host and EtherLink Plus adapter processors. The contents of this register can be read back by the host. This register is cleared upon power-up.

| ATTN | FLSH | DMAE | DIR | TCEN | CMDE | HSF2 | HSF1 |
|---|---|---|---|---|---|---|---|

**ATTN:** Attention
When the host sets ATTN, a non-maskable interrupt (NMI) is generated to the adapter's 80186 processor. The Host Control and Status Registers on the adapter are not affected. The interpretation of NMI is intended to be "soft reset", where the adapter resets itself into an idle state ready to accept commands.

**FLSH:** Flush Data Register
Setting the FLSH bit flushes all data words from the Data Register regardless of the state of the DIR (direction) bit. The FIFO assumes an empty condition, although the actual data in the FIFO is unchanged. The Data Register remains in this state until the FLSH bit is cleared.

**ATTN+FLSH:** Reset adapter
When the host simultaneously sets both ATTN and FLSH, the adapter hardware decodes it as a "hard reset". The Data Register, Adapter Status and Control Registers, and the Host Status Register are reset. A reset signal to the 80186 processor is generated which resets all 80186 internal registers and transfers control to the power up reset location. The 82586 is also reset. The adapter will stay in this reset state until the ATTN and FLSH bits are cleared.

**DMAE:** DMA enable
Used in conjunction with the DIR bit, DMAE enables DMA transfers to or from the Data Register. DMA requests to the host can occur only if this bit is set. With the DMAE bit cleared, the DMA request output to the host "floats" and another I/O card may use the channel. A Terminal Count interrupt request to the host is cleared by clearing this bit.

**DIR:** Direction flag
The host has exclusive control of the direction of the half-duplex Data Register. If DIR is clear, data transfers are to the adapter (download). If DIR is set, data transfers are to the host (upload).

⚠ **CAUTION:** After completing a download, the host must make sure that the adapter has completed its transfer (FIFO empty) before changing the DIR bit to the upload state. This can take 1 to 30 microseconds, depending on the network activity occurring on the adapter.

**TCEN:** Terminal Count interrupt enable
TCEN enables an interrupt to the host at the completion of a DMA transfer to or from the Data Register. After an interrupt, the request is cleared by clearing DMAE.

**CMDE:** Command Register interrupt enable
The CMDE control bit allows the host to be interrupted when the adapter has written the Command Register. The interrupt request is cleared when the Command Register is read.

When neither TCEN nor CMDE are set, the host should disable the interrupt channel because the interrupt request line will float.

**HSF1, HSF2:** Host Status Flags 1 and 2
The HSF1 and HSF2 status bits are routed directly to the Adapter Status Register. They are general purpose in nature and can be used by host and adapter interface drivers to synchronize data transfer or pass command completion status.

# Host Status Register

The Host Status Register (HSR) is an 8-bit register used by the host to determine causes of interrupts, check status of both Data and Command Register programmed I/O, and provide a way to synchronize the host and EtherLink Plus adapter processors.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| HRDY | HCRE | ACRF | DIR | DONE | ASF3 | ASF2 | ASF1 |

**HRDY:** Data Register ready
The HRDY bit indicates whether the Data Register is not full or not empty, depending on the Direction Flag. When the host is downloading data to the adapter, HRDY set means that the Data Register is not full, i.e., ready for more data. When the host is uploading data from the adapter, HRDY set means that the Data Register is not empty, i.e., input data is available.

**HCRE:** Host Command Register empty
The HCRE flag is used to handshake data transfer through the Command Register from the host to the adapter. When the host writes the Command Register, HCRE is cleared indicating the register is not empty. When the adapter has read the Command Register, HCRE is set, indicating that the register is empty.

**ACRF:** Adapter Command Register full
The ACRF flag is used to handshake data transfer through the Command Register from the adapter to the host. When the adapter writes the Command Register, ACRF is set, indicating the register is full. When the host reads the Command Register, ACRF is cleared, indicating that the register is not full.

**DIR**: Direction flag
The DIR status bit is the current value of the DIR control bit in the Host Control Register. It specifies in which direction data is allowed to pass through the Data Register. When DIR is clear, transfers are from the host to the adapter (download). When DIR is set, transfers are from the adapter to the host (upload). The DIR bit also determines how HRDY should be interpreted.

**DONE:** DMA done
The DONE flag is set when a DMA transfer between the host and the Data Register is complete. An interrupt to the host will also be generated if the TCEN bit in the Host Control Register is set. The DONE bit is cleared by clearing the DMAE bit in the Host Control Register.

**ASF1, ASF2, ASF3**: Adapter Status Flags
The ASF1, 2 and 3 status bits are routed directly to the Host Status Register from the Adapter Control Register. They are general purpose in nature and can be used by host and adapter interface drivers to synchronize data transfer or pass command completion status.

---

⚠ **CAUTION**: These bits are set asynchronously with respect to the host processor and it is possible to read these bits while they are in transition. This is only a problem if the state of more than one flag is tested simultaneously. For example, if the present state is ASF1 = ASF2 = 0 and you are testing for state ASF1=1 and ASF2=0, you could actually read this state during a state transition to ASF1 = ASF2 =1, if the ASF2 flag changed state slower than the ASF1 flag. The solution is to read the Adapter Status Register twice when checking the state of more than one flag to ensure that you have not read a flag in transition.

---

# Host AUX DMA Register

The Host Aux DMA Register is used to support demand mode DMA transfers. This register is cleared upon power-up. It doesn't exist on older Rev 2 hardware boards.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | BRST |

**BRST**: DMA Burst
If the Burst bit is not set, demand mode DMA transfers by the host will pause every 9 transfers to allow the PC to refresh its dynamic RAMs. If the Burst bit is set, no such pause will occur. This bit has no effect during single cycle DMA transfers.

---

⚠ **CAUTION**: Do not use burst or demand mode DMA in PC or XT type PC's to transfer from the host to the adapter. Hardware implementations of DMA on these machines can cause data errors in this mode.

---

# Adapter Control Register

The Adapter Control Register (ACR) is an 8-bit register used by the adapter to reset the 82586, flush the Data Register, blink the LEDs, and set the state of synchronization flags between the PC host and EtherLink Plus adapter processor. The contents of this register can be read back by the adapter. This register is cleared upon power-up.

| LPBK | FLSH | R586 | LED2 | LED1 | ASF3 | ASF2 | ASF1 |
|---|---|---|---|---|---|---|---|

**LPBK:** Loopback control LPBK specifies a diagnostic mode in which transmitted data is not placed on the network, but is looped back into the adapter. This controls loopback at the 8023 Manchester Code Converter. If CLEAR, loopback mode is enabled.

**FLSH:** Flush Data Register
Setting the FLSH bit flushes all data words from the Data Register regardless of the state of the DIR (direction) bit. The FIFO assumes an empty condition, although the actual data in the FIFO is unchanged. The Data Register remains in this state until the FLSH bit is cleared.

**R586:** Reset 82586
When the adapter sets R586, a hardware reset is applied to the 82586 coprocessor chip. All major 82586 hardware components are reset to an inactive state and remain reset until R586 is cleared. The 82586 then waits for the Channel Attention signal before completing initialization.

**LED2:** LED control bit 2
LED2 determines the state of LED 2. Setting the bit turns the LED on, and clearing the bit turns the LED off.

**LED1:** LED control bit 1
LED1 determines the state of LED 1. Setting the bit turns the LED on, and clearing the bit turns the LED off.

**ASF1, ASF2, ASF3:** Adapter Status Flags
The ASF1, 2 and 3 status bits are routed directly to the Host Status Register. They are general purpose in nature and can be used by host and adapter interface drivers to synchronize data transfer or pass command completion status.

# Adapter Status Register

The Adapter Status Register (ASR) is an 8-bit register used by the adapter to determine causes of interrupts, check status of both Data and Command Register programmed I/O, and provide a way to synchronize the host and EtherLink Plus adapter processors.

| ARDY | ACRE | HCRF | DIR | 8/16 | SWTC | HSF2 | HSF1 |
|---|---|---|---|---|---|---|---|

**ARDY**: Data Register ready
The ARDY bit indicates whether the Data Register is not full or not empty, depending on the Direction Flag. When the host is downloading data to the adapter, ARDY set means that the Data Register is not empty, i.e., input data is available. When the adapter is uploading data to the host, ARDY set means that the Data Register is not full, i.e., ready to accept more data.

**ACRE**: Adapter Command Register empty
The ACRE flag is used to handshake data transfer through the Command Register from the adapter to the host. When the adapter writes the Command Register, ACRE is cleared, indicating that the register is not empty. When the host reads the Command Register, ACRE is set, indicating that the register is empty.

**HCRF**: Host Command Register full
The HCRF flag is used to handshake data transfer through the Command Register from the host to the adapter. When the host writes the Command Register, HCRF is set, indicating the register is full. When the adapter reads the Command Register, HCRF is cleared, indicating that the register is not full.

**DIR**: Direction flag
The DIR status bit specifies in which direction data is allowed to pass through the Data Register. The direction can be set only by the host using the DIR bit in the Host Control Register. When DIR is clear, transfers are from the host to the adapter. When DIR is set, transfers are from the adapter to the host.

**8/16**: 8/16 bit
The 8/16 bit flag indicates whether the adapter is installed in an 8 or 16 bit expansion slot. If the 8/16 bit is set, the adapter is in a sixteen bit slot, i.e., an IBM AT or AT-compatible.

**SWTC**: External switch
The SWTC flag in the Adapter Status Register represents the state of the TEST jumper on the adapter. When the TEST jumper is set to one, the Revision 3.0 ROM code will:

1. Ignore power up memory test error. Memory errors detected during power up normally prevent the adapter from entering the main ROM idle loop. Ignoring errors is useful when using ICE systems that need to modify the NMI vector location in order to operate.

2. Ignore ROM checksum error. During ROM development, it is convenient not to checksum since the code is changing frequently.

3. Install 3D interrupt vectors. The interrupt vectors known as "exceptions" (basically INT 0 to 7) and all unused interrupt vectors are made to point to the 3D slave in the Revision 2.0 ROM. When an exception occurs, 3D becomes active and attempts to communicate with the 3D Debugger program.

**HSF1, HSF2**: Host Status Flags
The HSF1 and HSF2 status bits are routed directly from the Host Control Register. They are general purpose in nature and can be used by host and adapter interface drivers to synchronize data transfer or pass command completion status.

---

⚠ **CAUTION:** These bits are set asynchronously with respect to the 80186 and it is possible to read these bits while they are in transition. This is only a problem if the state of both flags is tested simultaneously. For example, if the present state is HSF1=HSF2=0 and we are testing for state HSF1=1 and HSF2=0, we could actually read this state during a state transition to HSF1=HSF2=1, if the HSF2 flag changed state more slowly than the HSF1 flag. The solution is to read the Host Status Register twice when checking the state of both flags to ensure that you have not read a flag in transition.

---

# Chapter 4: Command Interface

## Introduction

The 16KB of EPROM on the EtherLink Plus adapter contains firmware that supports the following:

- Bootup initialization and diagnostics
- Software memory refresh
- Network I/O
- Packet buffer control
- Host I/O
- System timer
- Host PC primary command interface

After adapter bootup initialization, host-based application programs or drivers can access the network or resources of the adapter through the primary command block interface described in the next sections. Additionally, programs can be downloaded into the adapter and executed there. Downloaded programs can access the adapter resources through a set of interrupt vectored utilities or directly through registers, the 80186 and other hardware functions.

## Primary Command Block Structure

The EtherLink Plus firmware idles waiting for a Primary Command Block (PCB) from the PC host. The PCB structure is expected during command/response sequences. The following shows the format of a PCB:

**PCB Format**

| | |
|---|---|
| PCB command code | (byte) |
| PCB data length | (byte) |
| PCB data | (up to 62 bytes) |
| PCB total length | (byte) |

The PCB total length is not explicitly part of the PCB structure but it is passed as the last byte on all PCB transfers, just before setting status bits to end the transfer.

The maximum PCB size the adapter can accept in this version ROM is 64 bytes. The PCB data length field does not include the PCB command code or the length field itself. The maximum data field is 62 bytes long. The valid PCB command codes are summarized in Table 4-1 and are explained in detail in the next section.

The PCB is passed using programmed I/O through the Command Register. In most command sequences, the host transfers a command PCB and receives a response PCB from the adapter. Some commands pass all required data in the PCB structure. Many commands also have a data transfer portion (DMA or programmed I/O) associated either with the request or response PCB. A few command sequences have only one PCB rather than the normal request/response PCB pair. Typically, these single PCB commands will only be used in conjunction with a program that has been downloaded to the adapter.

An example PCB might contain an 82586 configuration command, a length field that counts the number of bytes in the data field, and a data field that has configuration data needed to set up the 82586 coprocessor. The adapter will return a corresponding PCB with the completion status of the configuration command in its data field.

**Table 4-1. PCB Command Code Summary**

| Host -> EtherLink Plus Adapter Commands | | |
|---|---|---|
| 00: | n/a | |
| 01: | Configure Adapter Memory | set adapter buffer requirements |
| 02: | Configure 82586 | set 82586 receive mode |
| 03: | Station Address | get adapter station address |
| 04: | Download Data to adapter | download using adapter DMA * |
| 05: | Upload Data to host | upload to host using adapter DMA* |
| 06: | Download Data to adapter | download using adapter PIO * |
| 07: | Upload Data to host | upload to host using adapter PIO* |
| 08: | Receive Packet | receive a packet |
| 09: | Transmit Packet | transmit a packet |
| 0a: | Network Statistics | includes 82586 error counts |
| 0b: | Load Multicast List | perform 82586 MC-setup command |
| 0c: | Clear Downloaded Programs | release download program memory |
| 0d: | Download Program | download program to adapter |
| 0e: | Execute Program | execute program in adapter |
| 0f: | Self-Test | perform adapter self-test |
| 10: | Set Station Address | set station address in 82586 |
| 11: | Adapter Info | get adapter information |
| 12: | reserved | |
| : | \| | |
| 2f: | reserved | |

| EtherLink Plus Adapter -> Host Commands | | |
|---|---|---|
| 30: | n/a | |
| 31: | Configure Adapter Memory | returns success or failure |
| 32: | Configure 82586 Response | returns success or failure |
| 33: | Address Response | returns station address |
| 34: | Download Data Request | request DMA download to adapter ** |
| 35: | Upload Data Request | request DMA upload to host ** |
| 36: | n/a | |
| 37: | n/a | |
| 38: | Receive Packet Complete | receive packet request complete |
| 39: | Transmit Packet Complete | transmit packet request complete |
| 3a: | Network Statistics Response | returns network statistics |
| 3b: | Load Multicast Response | returns success or failure |
| 3c: | Clear Program Response | returns success or failure |
| 3d: | Download Program Response | returns program id |
| 3e: | Execute Response | returns variable length data |
| 3f: | Self-Test Response | returns self-test results |
| 40: | Set Address Response | returns success or failure |
| 41: | Adapter Info Response | returns adapter information |
| 42: | reserved | |
| : | \| | |
| 5f: | reserved | |

* No response PCB associated

** Adapter initiated PCB

## Status Flag Usage for PCB Transfer

The adapter uses a 64-byte circular buffer to store the host PCB byte stream sent through the Command Register. For protection against stray bytes (from host aborted PCB transfers), the adapter does not consider a PCB transfer complete until the Host Status Flags (HSF2 and HSF1) are set by the host to state 11. Simultaneously, the TOTAL length of the PCB should be in the Command Register so the true beginning of PCB can be calculated. (This last total length byte is NOT included in the PCB data length field.) The adapter uses its status flags (ASF2 and ASF1) similarly to signal "end of PCB" when sending a PCB to the host.

The adapter is always ready to read a PCB but it might not always be able to accept it. To indicate the acceptance of the PCB, the adapter uses status flag state 01 after the host signals end-of-PCB. To indicate rejection, the adapter uses status flag state 10. When the adapter sends a PCB to the host, it expects the host to set its status flags similarly to signal acceptance or rejection.

In summary, the adapter uses and expects the host to use the following conventions:

**Adapter or Host Status Flags**

| SF2 | SF1 | Meaning |
|---|---|---|
| 0 | 0 | None (State 0) |
| 0 | 1 | PCB accepted |
| 1 | 0 | PCB rejected |
| 1 | 1 | End of PCB |

The state 11 is accompanied by the total length of the PCB just transmitted. After a PCB is received, the state 01 or 10 is used to signal acceptance or rejection of the PCB.

## Host to Adapter Request

The following method is suggested to send a host PCB to the EtherLink Plus adapter:

- Load the PCB command byte into the Command Register; this will interrupt the EtherLink Plus adapter, synchronizing it to the PC host for the data transfer.
- Poll the Command Register Empty flag (HCRE) in the Host Status Register. Abort the I/O if it does not go empty within 40 ms.
- Output the remainder of the PCB similarly, reducing the timeout period to 500µs. The adapter remains in interrupt context to read PCB data.
- After the last actual PCB data byte is transferred, the host must send one last byte signifying the TOTAL length of the PCB (excluding this byte). Set the host status flags to state 11 (PCB End) before writing the length.
- Wait for adapter status flags 01 (Accept) or 10 (Reject) from the adapter. Assume a reject if a 50ms timeout occurs.

## Adapter to the Host Request or Response

The EtherLink Plus adapter to PC host request is made when the adapter needs to read or write a block of host memory. The adapter usually sends a response PCB after it has executed a host request.

The following method is used by the adapter to send a PCB to the host:

- Load the PCB command byte into the Command Register; this interrupts the PC host, synchronizing it to the adapter for the data transfer.
- Poll the Command Register Empty flag (ACRE) in the Adapter Status Register. Abort the I/O if it does not go empty within 20 ms.

- Output the remainder of the PCB similarly, reducing the timeout to 500µs. The host should remain in interrupt context to read PCB data or poll ACRF.
- After the last actual PCB data byte is transferred, the adapter must send one last byte signifying the TOTAL length of the PCB. The Adapter Status Flags are set to state 11 (PCB End) before writing the length.
- The adapter waits for Host Status Flags state 01 (Accept) or 10 (Reject).

# PCB Commands

## Host to EtherLink Plus Adapter PCB Formats

### 01H: Configure Adapter memory

The Adapter allocates memory for the PCB command queue, receive command queue, multicast address list, 82586 frame descriptors, receive buffers, and download program data structures. Each PCB and receive command queue entry is large enough to buffer the maximum size PCB of 64 bytes. A multicast list is kept in adapter memory to be loaded into the 82586 LAN coprocessor when in multicast mode. Receive and transmit buffers of 1.6Kb are always used to decrease buffer management and DMA overhead. The number of transmit buffers is fixed at one and is not configurable. If this command is not issued, the adapter uses the default values shown in parentheses below. The host should expect the adapter response PCB 31H to confirm execution.

```
db      01      ;command code
db      0C      ;length of data portion of PCB
dw      ?       ;# command PCB queue entries  (10)
dw      ?       ;# receive PCB queue entries  (20)
dw      ?       ;# multicast addresses        (0)
dw      ?       ;# frame descriptors          (20)
dw      ?       ;# receive buffers            (20)
dw      ?       ;# download programs          (10)
```

### 02H: Configure 82586

Instructs the adapter to configure the 82586 LAN coprocessor into the given receive mode. If this configure command is not issued, the adapter will use the default values shown in parentheses below. The host should expect the adapter response PCB 32H to confirm execution

```
db      02      ;command code
db      02H     ;length of data portion of PCB
dw      ?       ;receive mode
                ;   bit 2,1,0: receive mode (000)
                :       000 = station only
                ;       001 = plus broadcast
                ;       010 = plus multicast
                ;       100 = promiscuous
                ;   bit 4,3  : loopback mode (00)
                ;       00 = none (default)
                ;       01 = 82586 internal loopback
                ;       10 = 82586 external loopback
```

Multiple mode bits can be set, i.e., broadcast and multicast together would be a valid combination.

### 03H: Station Address

Requests adapter to return the station address stored in its address PROM. The adapter sends the PROM address in PCB 33H.

```
db      03      ;command code
db      00      ;length of data portion of PCB
```

### 04H: Download Data To Adapter

Requests the adapter to DMA download data through the data register. The direction bit must be set to the download direction before issuing the command. If the command is accepted, the adapter sets up the DMA transfer and expects the host to supply the required number of bytes. There is no adapter response PCB for this command.

```
db      04      ;command code
db      06      ;length of data portion of PCB
dw      ?       ;data block byte length (must be even)
dw      ?       ;adapter destination offset
dw      ?       ;adapter destination segment
```

### 05H: Upload Data To Host

Requests the adapter to use its DMA channel to upload data through the data register. The direction bit must be set to the upload direction before issuing this command. If the command is accepted, the adapter sets up the DMA and expects the host to read the given number of bytes. There is no adapter response PCB for this command.

```
db      05      ;command code
db      06      ;length of data portion of PCB
dw      ?       ;data block byte length (must be even)
dw      ?       ;adapter source offset
dw      ?       ;adapter source segment
```

### 06H: Download Data To EtherLink Plus Adapter

Operates as command code 04H, except that the adapter uses programmed input/output (PIO) instead of DMA. The direction bit must be set to the download direction before issuing this command. There is no adapter response PCB for this command.

```
db      06      ;command code
db      06      ;length of data portion of PCB
dw      ?       ;data block byte length (must be even)
dw      ?       ;adapter destination offset
dw      ?       ;adapter destination segment
```

### 07H: Upload Data To Host

Operates as command code 05H, except that the adapter uses PIO instead of DMA. The direction bit must be set to the upload direction before issuing this command. There is no adapter response PCB for this command.

```
db      07      ;command code
db      06      ;length of data portion of PCB
dw      ?       ;data block byte length (must be even)
dw      ?       ;adapter source offset
dw      ?       ;adapter source segment
```

### 08H: Receive Packet

Requests the adapter to receive an Ethernet packet. The packet type of interest is defined previously by the Configure PCB 02H. After a packet has been received, the adapter responds with PCB 38H. The adapter will then DMA upload the packet through the Data Register. The host should wait for the response PCB then set up to input the packet data.

```
db      08      ;command code
db      08      ;length of PCB data portion
dw      ?       ;offset of host receive buffer
dw      ?       ;segment of host receive buffer
dw      ?       ;host receive buffer length in bytes
dw      ?       ;timeout in 10ms increments (zero is
                ;no timeout; maximum is 32767 ticks)
```

### 09H: Transmit Packet

Requests the adapter to transmit a packet. If the PCB is accepted, the host should download the packet data through the Data Register immediately after this PCB is accepted. When the transmit is complete, the adapter responds with PCB 39H.

```
db      09H     ;command code
db      06      ;length of PCB data portion
dw      ?       ;offset of host transmit buffer
dw      ?       ;segment of host transmit buffer
dw      ?       ;packet length in bytes (must be even)
```

### 0AH: Network Statistics

This command requests the adapter to send the cumulative 82586 error statistics and the packet counters kept by the adapter. The values are returned through the command register in the adapter response PCB with command code 3AH. The adapter clears all statistics after sending the response.

```
db      0AH     ;command code
db      00      ;length of PCB data portion
```

### 0BH: Load Multicast List

The adapter will add the given list of multicast addresses to the 82586 multicast list. The adapter must be configured (PCB 01H) for an appropriate number of multicast addresses. A zero length list will cause the adapter to clear all multicast addresses and multiple PCB's can be issued to create lists greater than ten entries. The maximum number of addresses in the PCB is ten. The response PCB 3BH will contain command completion status.

```
db      0BH         ;command code
db      6*n         ;length of PCB data portion
db      6 dup(?)    ;Multicast address 1
        ...
db      6 dup(?)    ;Multicast address n (10 maximum)
```

### 0CH: Clear Downloaded Programs

This command releases all adapter memory previously allocated to downloaded programs. The adapter response PCB 3CH will contain the number of paragraphs of program memory available.

```
db      0CH     ;command code
db      0       ;length of PCB data portion
```

### 0DH: Download Program

Downloaded programs occupy adapter memory not used for packet buffers or system overhead. If this request is accepted, the adapter will DMA download the program. The host should set up to download the program immediately after this PCB is accepted. When done, the adapter responds with PCB 3DH containing a "program ID". The adapter always provides paragraph alignment to each downloaded program. It is suggested that the adapter be hard reset before a Download Program sequence.

```
db      0DH     ;command code
db      02      ;length of PCB data portion
dw      ?       ;program length in bytes
```

### 0EH: Execute Program

The adapter will pass control to the program defined by the program id. The first executable code is assumed at offset zero relative to the beginning of the downloaded program. The adapter routine employed is described in the Download Program Support section, 02H: Execute Program. The adapter, when done, responds with PCB 3EH.

```
db      0EH         ;command code
db      02+n        ;length of PCB data portion
dw      ?           ;program id
db      n dup(?)    ;variable length parameter list
```

The parameter list (if used) may be referenced by the downloaded program at the offset specified by es:bx registers. Parameter meanings are user definable. One 64 byte buffer is allocated for passing and returning parameters.

⚠ **CAUTION:** If more than one program is to be downloaded and executed this single buffer could cause conflicts.

### 0FH: Self-Test

The adapter will execute its self-test. The adapter, when done, responds with PCB 3FH.

```
db      0FH     ;command code
db      0       ;length of PCB data portion
```

### 10H: Set Station Address

The adapter will issue an IA-setup command to the 82586 coprocessor specifying an Ethernet station address. If this command is not used, the address from the Adapter's Ethernet address PROM is used to configure the 82586.

```
db      10H         ;command code
db      6           ;length of PCB data portion
db      6 dup(?)    ;station address
```

### 11H: Adapter Info

Requests the EtherLink Plus adapter to send general information that describes the adapter configuration. The adapter, when done, responds with PCB 41H.

```
db      11H     ;command code
db      0       ;length of PCB data portion
```

## EtherLink Plus Adapter to Host PCB Formats

### 31H: Configure Adapter Response

After the adapter has initialized the PCB command queue and the multicast address storage area, it responds with status in this PCB.

```
db      31H     ;command
db      02      ;length of PCB data portion
dw      ?       ;status  0 = successful
                ;   -1 = failure
```

### 32H: Configure 82586 Response

After the adapter has initialized the 82586 coprocessor using parameters in the PCB 02H command, it responds with status in this PCB.

```
db      32H     ;command code
db      02      ;length of PCB data portion
dw      ?       ;status  0 = successful
                ;   1 = failure
```

### 33H: Station Address Response

The adapter returns the 6-byte Station address in this response PCB. The address has previously been read from the Ethernet address PROM and stored into memory.

```
db      33H         ;command code
db      06          ;length of PCB data portion
db      6 dup(?)    ;station address
```

### 34H: Download Data To EtherLink Plus Adapter

In this PCB, the adapter requests that the host download a block of host memory to the EtherLink Plus adapter. If the command is accepted by the host, the adapter will use DMA to transfer the data through the data register.

This PCB is an asynchronous request from a downloaded program without a prior host PCB.

```
db      34H     ;command code
db      06      ;length of data portion of PCB
dw      ?       ;data block byte length (must be even)
dw      ?       ;host data block source offset
dw      ?       ;host data block source segment
```

### 35H: Upload Data To Host

In this PCB, the adapter requests that the host upload a block of data into host memory. If the command is accepted, the adapter will set up a DMA to transfer the appropriate data through the data register. This PCB is an asynchronous request from a downloaded program without a prior host PCB.

```
db      35H     ;command code
db      06      ;length of data portion of PCB
dw      ?       ;data block byte length (must be even)
dw      ?       ;host data block destination offset
dw      ?       ;host data block destination segment
```

### 38H: Packet Received Response

When the adapter receives a packet and there is an outstanding host request to receive a packet (as a result of PCB 08H), the adapter sends this response PCB and follows it with a DMA upload. The number of bytes DMA'ed will not exceed the buffer length specified in the receive packet command PCB 8H; extra packet data is discarded. The host has the option of rejecting this PCB, in which case, the packet will be discarded and no DMA upload will take place.

```
db      38H     ;command code
db      10H     ;length of PCB data portion
dw      ?       ;offset of host receive buffer
dw      ?       ;segment of host receive buffer
dw      ?       ;number of bytes to be DMA'ed
dw      ?       ;actual packet length
dw      ?       ;completion status     0 = successful
                ;                     -1 = timeout
dw      ?       ;82586 receive status
dd      ?       ;double word time tag in 15µs ticks
```

### 39H: Transmit Packet Complete

The status of a packet transmission is returned to the host in this response PCB.

```
db      39H     ;command code
db      08H     ;length of PCB data portion
dw      ?       ;offset of host transmit buffer
dw      ?       ;segment of host transmit buffer
dw      ?       ;completion status     0 = successful
                ;                     -1 = Xmit failed
                ;                     -2 = DMA timeout
dw      ?       ;82586 transmit status
```

### 3AH: Network Statistics Response

The adapter returns the total packet counters and the 82586 error counters in this response PCB. The adapter statistics are cleared.

```
db      3AH     ;command code
db      0CH     ;length of PCB data portion
dd      ?       ;total receive packets
dd      ?       ;total transmit packets
dw      ?       ;CRC error counter
dw      ?       ;alignment error counter
dw      ?       ;no resources error counter
dw      ?       ;overrun error counter
```

### 3BH: Load Multicast Complete

After the multicast list is loaded into the 82586, the adapter responds with this PCB.

```
db      3BH     ;command code
db      02      ;length of PCB data portion
dw      ?       ;status  0 = successful
                ;       -1 = failure
```

### 3CH: Clear Downloaded Program Response

To clear the down-loadable program memory, the adapter reinitializes the structures and variables describing each downloaded program. The adapter sends the amount of program memory available in paragraphs in this response PCB.

```
db      3CH     ;command code
db      02      ;length of PCB data portion
dw      ?       ;amount of downloadable program
                ;memory in paragraphs
```

### 3DH: Download Program Response

A downloaded program is assigned a "program ID" by the adapter. The ID is used by the host and adapter when specifying which downloaded program to execute or has executed.

```
db      3DH     ;command code
db      08      ;length of PCB data portion
dw      ?       ;program ID:  > 0 , if allocated
dw      ?       ;program offset in adapter memory
dw      ?       ;program segment in adapter memory
dw      ?       ;remaining memory in paragraphs
```

### 3EH: Execute Program Response

After a downloaded program has executed, it sends this response PCB to the host. The return status and parameters are program dependent.

```
db      3EH         ;command code
db      02+n        ;length of PCB data portion
dw      ?           ;program ID:  -1 if bad ID in request
db      n dup(?)    ;return status and parameters
```

Parameters and return status are user defined. See notes in PCB 0EH.

### 3FH: Self-Test Response

The adapter self-test consists of a ROM checksum, non-destructive RAM test, and internal loopback test on the 82586. Status of the test is returned in this PCB.

```
db      3FH         ;command code
db      2+2n        ;length of PCB data portion
dw      ?           ;self-test status
dw      n dup(?)    ;and optional failure data
```

The self-test status codes and failure data for each are:

| # | Status | Failure Data |
|---|---|---|
| 0 | no errors | none |
| 1 | ROM checksum | computed checksum value |
| 2 | RAM test | failed memory offset:segment |
| 3 | 82586 test | status word:<br>bit 14 = internal loopback failure<br>13 = external loopback failure<br>12 = configure error |

### 40H: Set Address Response

After the adapter sets the station address in the 82586, this response is sent to the host.

```
db      40H     ;command code
db      2       ;length of PCB data portion
dw      ?       ;status  0 = successful
                ;       -1 = failed
```

### 41H: Adapter Info Response

The adapter formats a response containing the ROM revision code, ROM checksum value, total amount of memory in kilobytes, and the segment/ offset pointer to free memory.

```
db      41H     ;command code
db      10      ;length of PCB data portion
dw      ?       ;ROM revision level (0x0300 = rev 3.0)
dw      ?       ;checksum value in rom
dw      ?       ;amount of memory in kbyte
dw      ?       ;free memory offset
dw      ?       ;and segment
```

# System ROM Utilities

Programs downloaded into the EtherLink Plus adapter can access the adapter resources directly or through a set of utilities available in ROM code. To simplify and standardize usage, a set of soft interrupt vectors have been allocated to support the following:

- Host I/O
- Network I/O
- Configuration/Status
- System Timer
- Download Program Support
- PCB Command Processing
- Receive Packet Processing
- Timed execution
- PCB Enqueuing

These vectors may be replaced or chained to by user downloaded programs. To chain to a vector, the downloaded program should first save the current vector before replacing it with a pointer to itself. A program then gains control when the vector is invoked. As appropriate, the program should pass control to the next program in the chain.

The EtherLink Plus adapter ROM utilities always save and restore the calling program's SS, DS, ES and SP. It is suggested that downloaded programs also maintain these registers.

# Host I/O Support: INT 80H

This group of functions allows upload and download of data and command blocks between the adapter and PC host. Both DMA and PIO methods of IO are supported.

### 01H: Download Request

The adapter formats a download request PCB, sends it to the host, and waits for host acknowledgement (HSF2 and HSF1 state 01 or 10). If the host accepts the request (state 01), the caller should proceed with the data transfer.

```
ax = 1    function code
es:bx     host source buffer address
cx        buffer length bytes (must be even; maximum 64Kb)
dx        initial timeout in 10ms increments
          (maximum is 32767 ticks)

Return:   carry clear if successful
          carry set if error, ax = error code
```

### 02H: DMA Upload Request

The adapter formats a upload request PCB and sends it to the host. If the request is acknowledged by the host (HSF2 and HSF1 state 01), the adapter should perform the data transfer.

```
ax = 2    function code
es:bx     host destination buffer address
cx        buffer length bytes (must be even; maximum 64Kb)
dx        initial timeout in 10ms increments
          (maximum is 32767 ticks)

Return:   carry clear if successful
          carry set if error, ax = error code
```

### 03H: Get Primary Command Block From Host

This function reads a PCB through the Command Register and stores it into the destination buffer.

```
ax = 3    function code
es:bx     buffer address
cx        buffer length bytes
dx        initial timeout in 10ms increments
          (maximum is 32767 ticks)

Return:   carry clear if successful
          carry set if error, ax = error code
```

### 04H: Send Primary Command Block To Host

This function sends the given PCB buffer to the host and waits for host acknowledgement either accept or reject.

```
ax = 4    function code
es:bx     buffer address
cx        total buffer length bytes
dx        initial timeout in 10ms increments
          (maximum is 32767 ticks)

Return:   carry clear if successful and accepted
          carry set if error, ax = error code
```

### 05H: Host Data DMA Input,

### 06H: Host Data PIO Input

Assuming that the host is already configured to perform a download, this function transfers host data into the passed buffer using DMA or PIO. When DMA is used, the timeout value passed in register DX is the amount of time the adapter waits for the DMA semaphore to signal "available". If a timeout occurs, the adapter assumes some sort of error has occurred and takes over use of the DMA channel. Also, the function initiates DMA transfer but does not wait for completion. Use INT 80H function 0BH to poll for DMA completion.

```
ax = 5    DMA download or
ax = 6    PIO download
es:bx     buffer address
cx        buffer length in bytes to transfer (must be even)
dx        timeout in 10ms increments
          (maximum is 32767 ticks)

Return:   carry clear if successful
          carry set if error, ax = error code
```

### 07H: Host Data DMA Output,

### 08H: Host Data PIO Output

Assuming that the host is already configured to perform an upload, this function transfers adapter data (using DMA or PIO) from the adapter buffer to the host. The comments describing DMA in int 80H function 05H also apply here.

```
ax = 7    DMA upload or
ax = 8    PIO upload
es:bx     buffer address
cx        buffer length in bytes to transfer (must be even)
dx        initial timeout in 10ms increments
          (maximum is 32767 ticks)

Return:   carry clear if successful
          carry set if error, ax = error code
```

### 09H: Accept PCB,

### 0AH: Reject PCB

When a PCB is read using int 80H function 3, the protocol described in the Status Flag Usage for PCB Transfer section requires that the PCB be accepted or rejected. The following function codes provide this ability to downloaded programs:

```
ax = 09H Accept host PCB
ax = 0AH Reject host PCB
```

Function 9, accept a PCB, also includes a Data Register flush operation to prepare for a DMA or PIO data transfer.

### 0BH: Check DMA Complete

When a DMA transfer has been initiated, this function can be called to check if the DMA has completed.

```
ax = 0BH Check DMA Complete
dx       timeout in 10ms ticks
         (maximum is 32767 ticks)
Return:  carry clear if DMA done
         carry set if timeout
```

## Network I/O Support: INT 81H

### 1H: Transmit Packet

To transmit a packet, this function links the given packet buffer to the first 82586 transmit buffer descriptor, links the descriptor to the one and only transmit command block, links the command block to the system control block, and then signals channel attention to the 82856. The transmit packet function will poll for transmit complete before returning to the calling routine.

```
ax = 1   function code
es:bx    buffer address
cx       buffer length bytes
dx       timeout in 10ms increments

Return:  carry clear if successful
         carry set if error, ax = 82586 transmit status
```

### 2H: Receive Packet

During execution of the adapter and 82586 configure commands the 82586 Receive Frame Area is initialized and the Receive Unit is commanded to start frame reception. A receive packet is first detected by the 82586 interrupt service routine, which time tags it and then updates global pointers to an "received packet" list. This function checks that list for an entry and gives it to the caller.

```
ax = 2   function code
dx       timeout in 10ms increments
         (maximum is 32767 ticks)

Return:  carry clear if successful
               es:bx = packet buffer address
               cx:dx = double word time tag (high:low order)
                   di = packet length in bytes
                   si = 82586 receive status
         carry set if error, ax = error code
```

### 03H: Return Buffer

After a packet buffer is processed, it must be returned to the system so that the buffer can be relinked to a Receive Buffer Descriptor which is in turn relinked to the free RBD list.

```
ax = 3    function code
es:bx     Buffer address

Return:   carry clear if successful
          carry set if error, ax = error code
```

## Configuration Status: INT 82H

### 01H: Configure Adapter Memory

The adapter allocates memory for the PCB command queue, receive command queue, multicast address list, frame descriptors, receive buffers, and download program structures. Each PCB or receive command queue entry is large enough to buffer a maximum size PCB of 64 bytes. Each multicast address occupies 6 bytes. Receive and transmit buffers are fixed at 1.6Kb in order to decrease management overhead. The number of transmit buffers is fixed at one.

```
ax = 1    function code
es:bx     pointer to configuration control block
          num_PCB_entries                    DW ?    (10)
          num_receive_Q_entries              DW ?    (20)
          num_multicast_entries              DW ?    (0)
          num_frame_descriptors              DW ?    (20)
          num_receive_buffers                DW ?    (20)
          num_download_programs              DW ?    (10)

Return:   carry clear if successful
          carry set if error, ax = error code
```

### 02H: Configure 82586 Receive Mode

Instructs the adapter to set the 82586 coprocessor into the given receive mode.

```
ax = 2    function code
bx        receive mode
          bit 2, 1, 0: receive mode (000)
                 000 = station only
                 001 = plus broadcast
                 010 = multicast
                 100 = promiscuous
          bit 4, 3: loopback mode (00)
                 00 = none (default)
                 01 = internal loopback
                 10 = external loopback

Return:   carry clear if successful
          carry set if error, ax = error code
```

### 03H: Return Station Address

Read the station address PROM and store the 6-byte address into the caller's buffer.

```
ax = 3    function code
es:bx     buffer address

Return:   carry clear if successful, buffer es:bx updated
          carry set if error, ax = error code
```

### 04H: Set Station Address

Use the 6 byte station address supplied in the buffer and issue an IA-setup command to the 82586 coprocessor.

```
ax = 4    function code
es:bx      buffer address

Return:   carry clear if successful
          carry set if error, ax = error
```

### 05H: Set LEDs

This function allows downloaded programs to control the state of the Adapter's two LEDs. LED #2 is periodically flashed by the adapter to indicate normal operation; this is called the "heartbeat".

```
ax = 5    function code
bx        control word
          bit 1, 0 = LED2, LED1 value; 1=ON
          bit    2 = enable/disable heartbeat; 1=enable

Return:   ax = current control register value in high byte
          carry clear if successful
          carry set if error
```

**NOTE:** The heartbeat refers to flashing LED 2 by the firmware about once per second, and is not related to 802.3 heartbeat.

### 06H: Get Adapter Info

Retrieve general adapter information.

```
ax = 6    function code
es:bx      buffer address

Return:   es:bx = revision id
                  rom checksum
                  memory size in kbytes
                  free memory offset
                  free memory segment
          cx       = data length in bytes
          carry clear if successful
          carry set if error
```

# Timer Support: INT 83H

The EtherLink Plus adapter maintains both a 10ms and 15µs double word time tick counter using two 16-bit timers in the 80186 microprocessor. The 15µs timer is meant for high resolution timeout or time tag applications and generates a timer interrupt every 0.98 seconds. The 10ms timer generates an interrupt every 10ms and can be used for timeout calculations. Every fifth 10ms tick the adapter sets a flag allowing the Idle vector to be called. The default Idle vector handler just resets the flag. The Idle vector allows downloaded programs that are "chained" through the Idle vector a chance to execute approximately every 50ms.

### 01H: Set 10ms Double Word Time

Set the global double word 10ms tick counter to given value.

```
ax = 1   function code
cx       high portion of 10ms count
dx       low portion of count
```

### 02H: Read 10ms Double Word Time

Retrieve the current double word 10ms tick counter.

```
ax = 2   function code
Return:  cx     high portion of 10ms count
         dx     low portion of count
```

### 03H: Set 15µs Double Word Time

Set the global double word 15µs tick counter to given value.

```
ax = 3   function code
cx       high portion of 15µs count
dx       low portion of count
```

### 04H: Read 15µs Double Word Time

Retrieve the current double word 15µs tick counter.

```
ax = 4   function code

Return:  cx     high portion of 15µs count
         dx     low portion of count
```

# Download Program Support: INT 84H

The adapter uses low memory for data, stack, packet buffers, and PCB command queue. Remaining memory is available for downloaded programs. Programs must have statically allocated memory for global data. Programs can use the one kiloword stack segment setup by the EtherLink Plus adapter ROM. Downloaded programs should not reconfigure adapter memory.

### 01H: Clear Downloaded Programs

This command releases all adapter memory previously allocated to downloaded programs. Soft interrupt vectors are restored to the reset state.

```
ax = 1    function code

Return:   carry clear if successful, ax = # free paragraphs
          carry set if error, ax = error code
```

### 02H: Execute Program

Control is passed to the program defined by the program ID. Executing a program is a far subroutine call to the program with the following registers setup:

```
ax = 2    function code
es:bx     address of parameter list
cx        length of parameter list in bytes
dx        program ID

Return:   carry clear if successful
                es:bx    = address of return buffer
                cx   = length of return buffer
          carry set if error, ax = error code
```

# PCB Command Processor: INT 85H

PCB's that are removed from the command PCB queue are processed by calling this interrupt. There are no sub-functions.

```
es:bx     pointer to PCB
```

The adapter Command Register ISR (interrupt service routine) reads host PCB's and places most PCB's into a command queue. Even PCB's with command values not defined in this specification will be placed into the command queue. Receive commands are placed into their own separate queue. A few PCB commands will be processed immediately, without being enqueued.

The EtherLink Plus adapter ROM idle loop monitors the command queue and dequeues each PCB for execution. The PCB is passed to a PCB Command Processor whose address is stored in this interrupt vector. The Command Processor is given an ES:BX pointer to the PCB entry which is obtained by the idle loop via INT 81 function 2.

The Command Processor uses the PCB command field to key into a jump table of command processing subroutine addresses. The selected command processing subroutine is also passed the PCB (often containing parameters for the subroutine). The Command Processor ignores PCB's with command numbers not defined in this specification, and immediately returns to the foreground idle.

A downloaded program can chain to this vector; that is, the program saves the current interrupt vector contents and replaces it with a pointer to itself. Then the downloaded program has an opportunity (not necessarily the first) to examine the PCB. If the program does not want to execute the PCB, the program must pass it to the command processor that it replaced. In this case the program must be careful that register values are not modified. This becomes a mechanism for creating new commands or replacing existing ones.

## Packet Processor Vector: INT 86H

This software interrupt vector defines the address of the Packet Processor. The Packet Processor centralizes handling and queuing of host receive command PCB's and received packets. The management of the 82586 LAN coprocessor is performed outside the Packet Processor. A downloaded program can replace this vector in order to implement a more specialized scheme than the ROM-based functions described below.

### 01H: Enqueue Receive Command PCB

Receive packet command PCB's are placed into a separate receive PCB queue. The Command Register ISR uses this function to place the receive PCB in the receive PCB queue for the Packet Processor to handle.

```
ax = 1    command code
es:bx     pointer to PCB
```

### 02H: Enqueue Receive Packet

When packets are received, the 82586 ISR time tags the frame with a double word 15µs time and updates global pointers to the frame and exits. The foreground idle loop obtains a newly received packet with Network I/O vector INT 81H function 2. Using this function, the idle loop places the packet into the receive packet queue for the Packet Processor to handle or sends the packet to the host.

```
ax = 2       command code
 es:bx       pointer to receive buffer
 cx:dx       double word time tag (high:low)
    di       packet length
```

When this function is called, the firmware checks the queue of receive PCB's. If there is a pending receive command, a receive response PCB is sent to the host, packet data is DMA uploaded, and the packet buffer is returned to the system. Otherwise, the packet is enqueued in the receive packet queue. This vector/function can be replaced by downloaded programs enabling them to receive and process all incoming packets.

**03H: Receive Scan**
Since receive commands have timeout values, the foreground idle loop will periodically call the Packet Processor with this function. The Packet Processor will then scan the receive command queue to check if any request has timed out, or if there is a pending receive command and a packet in the packet queue. If a request has timed out, a receive response PCB with failure status is formatted and sent to the host. If there is a receive request PCB and a packet has been queued, the packet will be sent to the host as described in INT 86 function 2.

```
ax = 3    NOP
```

## Idle Vector: INT 87H

The Idle vector is called approximately every 50 ms from the main adapter ROM idle loop. Programs chained through this vector will have a chance to execute.

To chain to the vector, a downloaded program should first save the current vector before replacing it with a pointer to itself. It is also important that the program pass control (using a far jump) to the next program in the chain (this will normally be the default INT 87 handler). Remember that SS, SP, DS and ES should always be saved on entry and restored on exit. In this way, each downloaded program in the chain has an opportunity to execute.

A modulo five counter runs continuously, incrementing every 10ms. When the counter reaches 5, a global flag is set allowing an Idle interrupt to occur. The default ROM Idle vector routine clears the flag that allows the Idle interrupt to take place. In a situation where a download program has chained to the Idle vector but never passes control to the default Idle vector routine, this flag is never cleared. The effect is that the download program will be called on every pass through the main adapter ROM idle loop.

## PCB Enqueue Vector: INT 88H

The PCB Enqueue Vector is called to add a PCB entry into the system PCB queue. PCB's will, later, be dequeued by the foreground idle loop and given to the PCB Command Processor for execution with INT 85. The calling sequence for this function is:

```
es:bx    pointer to PCB
Return:  carry clear if successful
         carry set if error, ax = error code
```

The Command Register interrupt service routine (ISR) uses this vector after it receives a PCB. If a downloaded program has directly read a PCB from the host and does not recognize it, it should use this vector to enqueue it for later execution. Alternatively, a downloaded program can chain to this vector to receive PCB's from the Command Register ISR before they are enqueued.

# Chapter 5: Programming

The following sections of this chapter provide a fairly detailed description of the main areas involved in the programming of the EtherLink Plus adapter. Some of the specifics may be hard to understand in this context, but they are covered separately, and in more detail, in later chapters.

## PCBs

The command interface between the host PC and the EtherLink Plus adapter is accomplished by the host passing defined PCB's (primary command blocks) to the adapter, and the adapter returning response PCB's to the host (if programs are run on the EtherLink Plus adapter, the adapter can also present unsolicited request PCB's to the host, i.e., upload data). These PCB's are transferred using programmed I/O to or from the Adapter's Command Register port. Synchronization and control of this process is provided using the Host Control Register and host Status Register. PCB's are provided to gather information or status from the adapter, configure the adapter, initiate transmit or receive functions, pass data or programs to or from the adapter, execute programs on the adapter, and test the adapter.

Some PCB's initiate a data transfer to or from the host in the course of their processing, and the host must be prepared to handle the data transfer through the Data Register port at the appropriate time. This is usually accomplished by the host setting up its DMA to transfer data to or from the Adapter's data port.

When sending PCB's to the adapter, the host should monitor the Host Status Register port for the HCRE bit (Host Command Register Empty) before writing a byte in the Command Register. The host can monitor for response PCB's by polling the Host Status Register port for the ACRF bit (Adapter Command Register Full), then reading the Command Register. Alternately, the host can enable command interrupts from the adapter with the CMDE bit in the Host Control Register. If this bit is set, the adapter will interrupt the host when it fills the command register, in the process of sending a response PCB to the host.

The PCB interface is presented in detail in chapter 3.

# Interrupts

The host can be interrupted by the adapter in two cases:

- When the Command Register is filled by the adapter (PCB response or request).
- When the host DMA reaches terminal count (DMA done).

Each of these interrupts has an enabling bit in the Host Control Register (CMDE and TCEN). Additionally, the DMA Done interrupt assumes DMA has also been enabled with the DMAE bit.

If programs are written to be downloaded and run on the EtherLink Plus adapter, several hardware interrupts are available to the 80186 on the adapter:

- DMA Channel 1 Done. When the onboard DMA reaches terminal count.
- Timer. Every 10 ms.
- Command Register Full. When the host writes into the Command Register.
- 82586 Interrupt. On receive and others (see Intel Microcommunications Handbook).
- Attention (NMI). When the Attention bit in the Host Control Register is set by the host. This is used to initiate a reset.

The interrupt functions are covered in Chapter 1.

# Data Transfer and DMA

Data (other than commands) is passed to or from the adapter through the Data Register port. Data transfers are normally initiated in conjunction with a particular type of PCB process. The host (and EtherLink Plus adapter) are expected to know when data transfers are required and perform the I/O at the appropriate time. The host controls the direction of transfer using the DIR bit in the Host Control Register.

The host can perform data transfer by polling the Host Status Register for the status of the HRDY bit then reading or writing the Data Register port. If DMA transfer is desired, the host needs to initialize its DMA with address, length, direction, etc., then set the Host Control Register to specify direction and DMA Enable. If a DMA Done interrupt is desired, that bit should also be set.

The adapter uses its own onboard DMA in a similar fashion to transfer data between adapter memory and the Data Register port, thus completing the adapter side of the total transfer.

DMA and data transfer functions are described in chapter 1, with register descriptions in chapter 2, and PCB DMA requirements in Chapter 3.

# ROM Utilities

The EtherLink Plus adapter is equipped with a ROM that provides extensive functions for handling the 82586, managing packet buffer and processing queues, gathering information, configuring adapter parameters, and communicating with the host. Many of these functions are available to the host via the PCB interface. Additionally, the host can initiate (with PCB's) a download of a program to the adapter and have it execute on the adapter. A program running in this manner can access or modify many more of the EtherLink Plus adapter functions by respectively calling or replacing software interrupts on the board.

The default resident ROM program on the adapter uses these same interrupts to initiate processing for PCB's, packet management, data transfers, configuration, and so forth. The interrupt functions available on the adapter are described, in detail, in chapter 3. As a guide to how these functions are utilized by the ROM, and how the normal adapter processing is implemented, Appendix H contains a listing of the code of the main processing loop of the firmware on the EtherLink Plus adapter. The listing shows which ROM routines are called by the loop to initiate processing of packets and commands. The comments in the section on the command (PCB) interrupt handler show how processing is initiated for each type of PCB.

# Data Structure

Some of the memory on the adapter is configured to provide queues for received packets and some to queue PCB's from the host. On an EtherLink Plus adapter with default configuration, some memory is free and can be used to allocate more packet or PCB buffers, or to load programs.

Figure 5-1, EtherLink Plus Adapter Data Flow, shows the main data structures and data flow, along with the ROM utility software interrupts, that are used by the firmware to control processing. This figure should help in understanding the discussion that follows. On the figure, the two dashed boxes labeled Downloaded Program represent likely places for a program to take control of the EtherLink Plus adapter packet processing. (Indeed, these functions are implemented in a download program as part of the demo program on the software diskette. See Appendix E). If no program is loaded, the flow will pass directly through these boxes.

The data structures for receiving packets are organized into three queues of receive buffers. All the receive buffers are 1.6 kb in size and can, hold an entire packet. Initially, all receive buffers will be located on the Free List queue of unallocated packet buffers. As the 82586 receives packets and interrupts the processor, packets are moved to the Receive List queue of unprocessed packets. The firmware's main processing loop examines this queue for new packets via INT 81 - 2. If new packets are found, they are processed with INT 86 - 2. If a receive PCB is outstanding, the packet will be sent to the host with INT 80 - 4 and INT 80 - 7, otherwise, INT 86 - 2 enqueues the received packet on the rcvPkt queue. When a receive PCB is available, INT 86 - 3 will dequeue the packet and send it to the host with INT 80 - 4 and INT 80 - 7. After sending the packet the receive buffer is returned to the Free List with INT 81 - 3.

The number of receive buffers (that make up the contents of the 3 queues), and, therefore, the number of packets that can be buffered on the adapter, is a configurable parameter on the adapter. Receive buffers have Frame Descriptors and Receive Buffer Descriptors associated with them (see Intel 82586 documentation for descriptions). The number of Frame Descriptors is also configurable and is normally set equal to the number of receive buffers.

PCB commands are held in two additional queues. When a command is received on the adapter it is either processed immediately or placed in one of two queues for processing. If the PCB is a receive command, it is placed in the RcvPCB queue with INT 86 - 1, otherwise it is placed in the command PCB queue with INT 88. The size of these two queues are individually configurable. Each of the command queue entries is several bytes larger than a PCB (64 bytes).

A few added notes on the somewhat complicated process of handling PCB's are probably in order. When a PCB is received by the adapter, control is passed to the PCB pre-processor, either by a command interrupt or via a direct call to the interrupt handler (hcmdf_proc) from the main loop. A few PCB's are processed immediately. Most are enqueued on the PCB queue with INT 88. Receive PCB's are enqueued on their own rcvPCB queue with INT 86 - 1. The action performed for a particular PCB is listed in the comments of the main loop listing (Appendix H). If a PCB is received with a command code greater than those defined in the documentation, the PCB will be enqueued on the PCB queue via INT 88 and later discarded when INT 85 is called to process packets.

Packet transmission is implemented with a single pre-allocated packet buffer (not shown on the figure).

The downloaded program, shown on the figure, implements a packet filter operating on the EtherLink Plus adapter. In one section, the program has inserted itself on the entry to the INT 85 PCB processor. This allows the program to receive control PCB's from the host. New PCB commands are defined for the program, and if it sees one of these PCB's it processes the PCB according to the definition. If the program finds a PCB other than its own, it passes the PCB on to the standard INT 85 PCB processor.

The other section of the downloaded program is the packet filter. The program has inserted itself on the entry to INT 86 - 2. Here it will receive a pointer to the receive buffer for every packet that is received. The program can examine the contents of the packet and has the option to discard or process the packet. If the packet is to be discarded, the program calls INT 81 - 3 to free the packet buffer, then returns to the caller of INT 86 - 2. If the packet is to be processed normally, the program passes control to the normal INT 86 -2 routine. In this case the program could also modify the packet before passing it on.

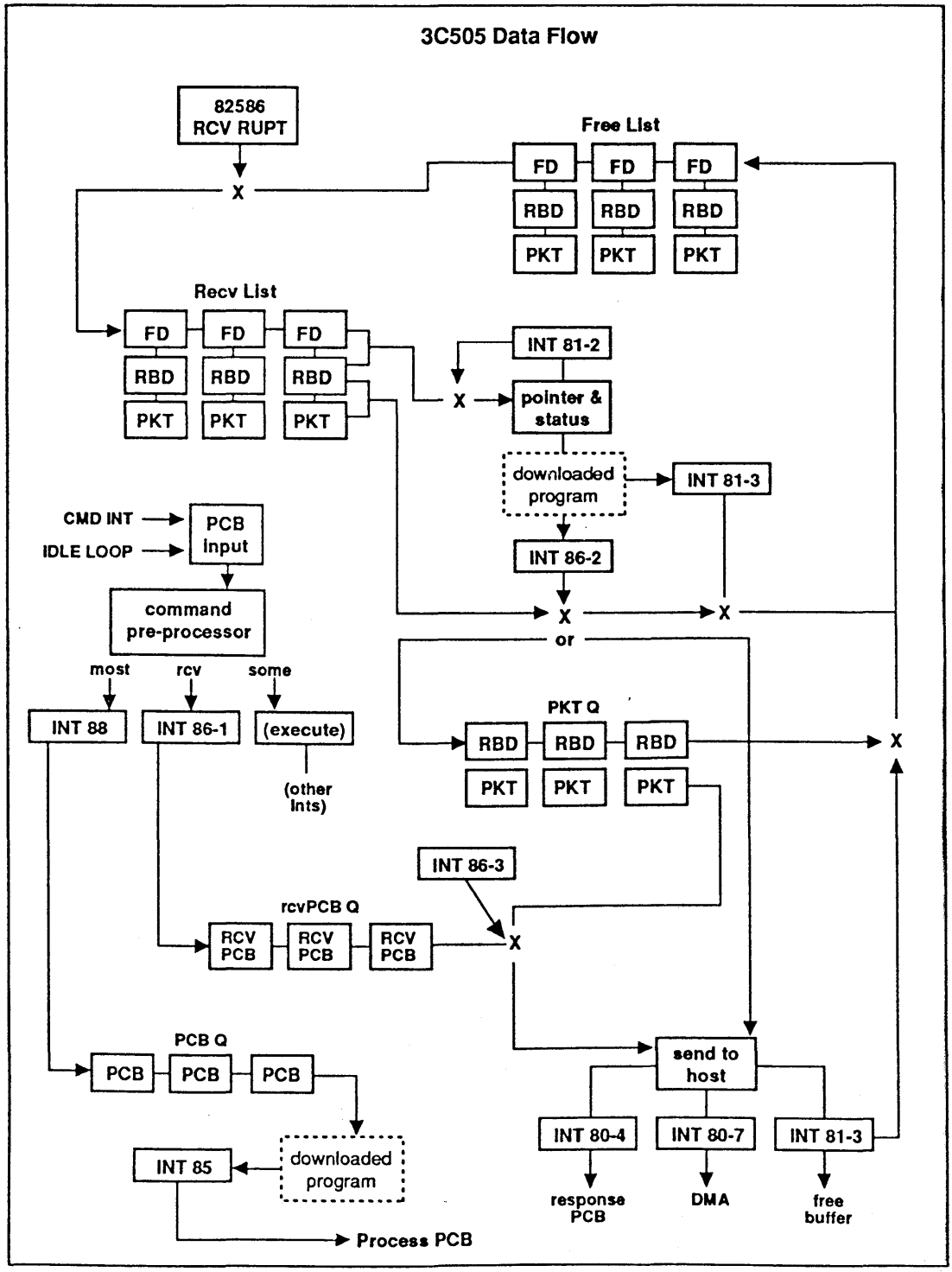



Figure 5-1. EtherLink Plus Adapter Data Flow

# Appendix A: 80186 Peripheral Control Block Programming

```
;
;These are the default values set by the adapter firmware
;
reloc_reg_cont equ     00ffh            ;initialization constant of
umcs_cont      equ     0fc3ch           ;chip selects
lmcs_cont      equ     3ffchmmcs_cont
               equ     81fchmpcs_cont
               equ     0a0bch
pacs_cont      equ     003ch
;
pic priority assignment:
;
level_int0     equ     4                ;int0 priority, Command Register
                                        ; Full
level_int1     equ     2                ;int1 priority, interrupt from
                                        ; 82586
level_int2     equ     5                ;not used
level_int3     equ     6                ;not used
level_dma0     equ     0                ;dma0 priority, DRAM refresh
level_dma1     equ     3                ;dma1 priority, Data Register
level_timer    equ     1                ;timer priority
;
;initialization of pic control registers
;
pic_timer_cont equ     level_timer
pic_dma0_cont  equ     level_dma0
pic_dma1_cont  equ     level_dma1
pic_int0_cont  equ     level_int0
pic_int1_cont  equ     level_int1
pic_int2_cont  equ     000dh
pic_int3_cont  equ     000eh
;
;initialization of timer 2 registers; 15 microsecond for DRAM
;refresh and Hi RES Timer
;
t2cnt_cont     equ     0
t2maxra_cont   equ     30
t2cntrl_cont   equ     0c001h
;
```

```
;initialization of timer 0 registers; 10 millisecond interrupt
;
t0cnt_cont          equ     0
t0maxra_cont        equ     20000/t2maxra_cont
t0maxrb_cont        equ     0
t0cntrl_cont        equ     0e029h
;
;initialization of timer 1 registers; Hi res system Timer
;
t1cnt_cont          equ     0
t1maxra_cont        equ     0ffffh
t0maxrb_cont        equ     0
t0cntrl_cont        equ     0e029h
;
;initialization of dma0 registers; DRAM refresh
;(Rev 3 ROM)
;
dma0srclo_cont      equ     0080h
dma0srchi_cont      equ     0000h
dma0dstlo_cont      equ     0080h
dma0dsthi_cont      equ     0000h
dma0cnt_cont        equ     0ffffh
dma0cntrl_cont      equ     0077h
;
;initialization of dma0 registers; DRAM refresh
;(these values for Rev 2)
;
dma0srclo_cont      equ     0000
dma0srchi_cont      equ     0000h
dma0dstlo_cont      equ     0800h
dma0dsthi_cont      equ     0008h
dma0cnt_cont        equ     0ffffh
dma0cntrl_cont      equ     0157fh
;
;DMA 1 control register values
;
dma1_from_dr_cntrl  equ     0a347h              ;DMA input from host
dma1_to_dr_cntrl    equ     1787h               ;DMA output to host
```

# Appendix B: 82586 Parameter Example

This is an example of the parameters used to configure the Intel 82586 LAN Coprocessor. Please refer to the Intel LAN Components User's Manual for the description of the abbreviations used.

| | |
|---|---|
| FIFO LIM | = 6 |
| BYTE CNT | = 0CH |
| EXT LP BCK | = 0 |
| INT LP BCK | = 0 |
| PREAM LEN | = 2 |
| AT LOC | = 1 |
| ADDR LEN | = 6 |
| SAV BF | = 0 |
| SRDY | = 1 |
| INTERFRAME SPACING | = 60H |
| BOF MET | = 0 |
| ACR | = 0 |
| LIN PRIO | = 0 |
| RETRY NUM | = 0FH |
| SLOT TIME | = 200H |
| CDT SRC | = 0 |
| CDTF | = 0 |
| CRS SRC | = 0 |
| CRSF | = 0 |
| PAD | = 0 |
| BT STF | = 0 |
| CRC16 | = 0 |
| NCRC INS | = 0 |
| TONO CRS | = 0 |
| MANCH/NRZ | = 0 |
| BC DIS | = 1 (reconfigurable by host command) |
| PRM | = 0 |
| MIN FRM LEN | = 40H |

# Appendix C: Diagnostics

The diskette supplied with your EtherLink Plus adapter card includes a diagnostic program called 3C505.EXE, which can be used to help you identify hardware problems on the EtherLink Plus adapter.

This appendix describes the 3C505.EXE program, the equipment and tools required to run the program, and a step-by-step procedure on how to use the diagnostic.

3C505.EXE takes about twelve minutes or less (five minutes in an AT) from start to finish. As it runs, it reports its progress by displaying the name of the group of tests being performed and a pass count. The test stops and displays an error message if a hardware error is detected.

0 - Adapter self test
1 - Preliminary test
2 - DMA test
3 - Packet test
4 - Recognizer test
5 - Message exchange test
6 - Passive receive test
7 - NS echo server

When the standard 3C505.EXE program is run, Test 0 through Test 4 are performed in sequence. Test 5 through Test 7 are run individually and must be specified separately. Tests 3 and 4 require the use of a loopback plug to prevent network activity from producing erroneous test results, and to keep test packets from polluting the network. Tests 5 through 7 must be run while the adapter is connected to the network. A brief description of each test follows:

**TEST 0:**
Resets the adapter causing the self-test routines to be executed. These include 80186 and 82586 initialization, memory and internal and external loopback tests. The results are passed back to the host and displayed. If communication between the host and adapter can not be established, an error message is displayed.

**TEST 1:**
Tests the interface between the host and the adapter using Programmed I/O data transfers.

**TEST 2:**
Tests the interface between the host and the adapter using DMA data transfers.

**TEST 3:**
Performs a transmit test and, if successful, a loopback test. The transmit test checks for the correct status from the 82586 LAN controller following transmits. Loopback test further compares the received packets with those transmitted.

**TEST 4:**
Tests the 82586 LAN controller address matching functions. The receiver is configured to various modes; station only, multicast, broadcast, and promiscuous. In each mode, packets of differing destination address and size are transmitted and the ability of the adapter to reject or accept packets properly is tested.

**TEST 5:**
Performs packet exchange with another PC or server on the network. The PC transmits an "echo request packet" into the network. A responding server or PC will transmit the packet (echo) back to the PC under the test.

**TEST 6:**
Detects legal packets on the network and counts them. This tests the adapter's receive function and provides a diagnostic tool for locating problems elsewhere on the network. This test is "passive" to the network and can be used to check the transmit capability of another PC on the network.

**TEST 7:**
Designates this PC as an "echo server", which is used to exchange packets with PCs running Test 5. The PC remains in this mode until a key is depressed.

# Diagnostic Command Format

The program is called using the command format:

3C505 [-Ix][-Dx][-Bxxx][-#][-E][-T]

where:

Ix = Test uses interrupt x, default (factory setting) is Interrupt 3.

Dx = Test uses DMA channel x, default is DMA 1.

Bxxx = Sets the base address of the EtherLink card to xxx (three hexadecimal digits). This option should be used if the I/O address jumpers on the EtherLink card have been changed. The default value is 300 hex.

\# = Enter 5, 6 or 7 to select one of the following tests.
5 - Message exchange test
6 - Passive receive test
7 - NS echo server

E = Used with test 5 only. Use NS echo protocol to access remote nodes during the message exchange test. This option should be used on a Xerox NS 8000 network that have echo servers or if there is another PC with a 3Com EtherLink Plus adapter running 3C505 -7 to reply to the echo request. If this option is not specified, the diagnostic will generate EtherSeries echo requests, and any EtherSeries server on the network will reply using EtherSeries protocol.

T = Specifies that the host computer is a TI Professional which requires special treatment.

# Requirements for Testing

For testing, you need:

1. A loopback plug;

2. Another IBM PC on the network OR an EtherSeries network server that is connected to the network.

   The second PC will be used as an echo server, which will exchange packets over the network with the computer under test.

# Running the 3C505.EXE Program

To start the 3C505.EXE program, disconnect your PC from the network, attach a loopback plug to the BNC connector, insert the diskette in drive A and type:

3C505

Remember to give the option -I, -D, -B if the factory settings were changed. As the program runs, it prints a message indicating which test is being performed and the progress. Once it detects an error, the test stops and displays a message. Test 0 thru 4 mentioned above will start one after another.

To run Test 5, connect the PC to the network with either:

1. An EtherSeries network server

2. Another PC, with an EtherLink Plus adapter as a echo server. Start the server running 3C505.EXE on this diskette by typing:

   3C505 -7

   OR

3. Another PC with another type of 3Com EtherLink acting as an echo server. Use the diagnostic program supplied with that specific EtherLink such as the 3C501 and type:

   A> 3C501-7

   If the echo server is case 2 mentioned above, then type:

   3C505 -5 -E

   If the echo server is case 3 mentioned above, then type:

   3C505 -5

## 3C501 / 3C505 Diagnose Program Differences

The 3C505.EXE diagnostic is modeled after the 3C501 version, 3C501.EXE (or DIAGNOSE.COM in earlier versions). For those of you familiar with the 3C501 diagnostic, all test types (Preliminary, DMA, Packet, Recognizer, Message exchange, Passive receive, NS Echo server) have been carried forward into the EtherLink Plus adapter diagnostic. The actual interpretation and implementation of these tests for the EtherLink Plus adapter are, however, very different. It is worth noting that in the EtherLink Plus adapter diagnostic,

- There is an additional test 0 (Adapter self test);
- Command line switches (to set adapter base address, DMA channel, etc. for the test) are always preceded by a dash "-";
- Tests 0 through 4 (Adapter self test, Preliminary, DMA, Packet, Recognizer) are always executed one after another (i.e., you cannot use "3C505 -3" to run Packet test only);
- Tests 0 through 4 are skipped when test 5, 6, or 7 (Message exchange, Passive receive, NS Echo server) is selected.

After running 3C505.EXE, you might find it necessary reboot your system or reinitialize the EtherLink Plus adapter. This is especially true if you use download software or if a particular EtherLink Plus adapter configuration is expected.

# Appendix D: 3D Debugger

3D is a program for loading and debugging programs that run on the EtherLink Plus adapter. 3D runs on an IBM PC (or compatible), under PC-DOS; the 3D host program communicates through the Command Register with the "slave", a small program in the EtherLink Plus adapter ROM. 3D can start, stop, and single-step the EtherLink Plus Adapter's 80186 processor; set, and clear breakpoints; download, modify, and examine memory.

3D requires an IBM PC with 256KB of memory, one serial port, and a Mouse Systems PC mouse. 3D also runs on IBM compatibles such as the Compaq portable and Zenith Z-150. Attach the mouse to the COM2 port. Recall that COM2 always uses interrupt level 3 so beware of conflicts. The EtherLink Plus adapter must be set to a base I/O address of 310 hex in this version of 3D. The TEST jumper on the EtherLink Plus adapter can also be set to the "1" position; this causes the 3D Slave to install itself at boot time. All "exception" and unused interrupt vectors will point to the 3D Slave. The ability to single step and to use breakpoints will be enabled.

3D divides the display into four regions: from top to bottom these are tile area, menu bar, typein, and message area. The tile area is for display and alteration of EtherLink Plus adapter internal registers, I/O registers, and memory. The menu bar is for control of the EtherLink Plus adapter. The typein area displays the accumulated typein from the user. The message area is for the display of status and error messages from 3D.

```
*AX         0000   BP       300C   CS             03F0   *IP        6A76
*BX         6892  *SP       3000   SS             3C6D   *FLAGS     F046
 CX         0000   SI       04C4   DS             03F0
 DX         0000  *DI       6806  *ES             03F0

*40:8       0040  *3C6D:3000     6A76
 40:0A      6892  *3C6D:3002     03F0
 40:0C      0000  *3C6D:3004     F046
 40:0E      0040  *3C6D:3006     F206
 40:10      0000  *3C6D:3008     6A85
 40:12      0300  *3C6D:300A     6FD7
 40:14      2E2E  *3C6D:300C     3C6D
 40:16      0000  *3C6D:300E     3C6D
                  *3C6D:3010     3C6D
 40:18      0014  *3C6D:3012     0000
 40:1A      0000  *3C6D:3014     04C4
 40:1C      0000  *3C6D:3016     303A
 40:1E      2E2E  *3C6D:3018     3022
 40:20      2E2E  *3C6D:301A     064D
 40:22      2F0A  *3C6D:301C     0000
 40:24      2F5A  *3C6D:301E     0000
Boot!  Load!  Go!  Continue!  Stop!  Step!  Break!  Unbreak!  UnbreakAll!
Probe!
ss:sp
popf >> Non maskable interrupt at 3F0:6A76
```

**Figure D.1 The Display Just After a Stop**

# Tile Area

The tile area is divided into a 21 by 4 array of tiles. Each tile displays the name of a register or memory location and its contents. A tile consists of three fields; in reading order these are flag, left, and right. The left field holds the register name and the right field displays it contents. A tilde, "~", in the flag means the data in the right field is invalid for some reason. An asterisk, "*", means the right field changed since 3D last read the register. When the cursor enters the tile area, 3D inverts the tile occupied by the cursor.

# Menu Bar

The menu bar provides access to functions for controlling the EtherLink Plus adapter. To select a function, move the cursor over the name of the function and click the left mouse button. Functions on the menu bar are Boot, Load, Go, Continue, Stop, Step, Break, Unbreak, UnbreakAll, and Probe.

**Boot** issues a hard reset to the EtherLink Plus adapter.

**Load** downloads a file from the IBM PC to the EtherLink Plus adapter through the Command Register. 3D always uses the last file specified unless the typein ends with a slash, "/". In this case, 3D uses the typein, minus the slash, as the file name; thus, if the same file is loaded many times, the file name need not be supplied every time. The slash is not necessary for the first file loaded. The file is downloaded into adapter memory at the address that is specified in the typein area when function key F6 is pressed. Please note that this Load operation is NOT THE SAME as the download program PCB described in Chapter 4.

**Go** evaluates the typein and starts the EtherLink Plus adapter running at the specified address.

**Continue** starts the EtherLink Plus adapter running at the current CS:IP. Once Go or Continue starts the EtherLink Plus adapter running, 3D becomes a "dumb" terminal for the EtherLink Plus adapter; 3D monitors both the IBM PC keyboard and the Command Register. 3D transmits any character typed on the keyboard through the Command Register. The bottom sixteen lines of the tile area become the display area for any characters transmitted by the EtherLink Plus adapter through the Command Register.

**Stop** establishes communication between 3D and the Slave through the Command Register. If the Slave does not respond, 3D displays an error in the message area. No other commands will work until 3D and the Slave are communicating. Stop is always appropriate no matter what state the EtherLink Plus adapter is in; however, booting will destroy the state of the EtherLink Plus adapter.

**Step** forces the EtherLink Plus adapter to execute the next instruction; values in the tile area are refreshed. The TEST jumper must be in the "1" position in order to single step.

**Break** sets a breakpoint at the location indicated by the typein. 3D allows eight breakpoints. All of these breakpoints are sticky, unlike debuggers which forget all breakpoints after hitting a breakpoint. After hitting a breakpoint, 3D removes all breakpoints instructions from memory. 3D restores all breakpoints when it Continues the EtherLink Plus adapter. The TEST jumper must be in the "1" position to use breakpoints.

**Unbreak** removes a breakpoint. Breakpoints are numbered from zero to seven; 3D evaluates the typein and removes the breakpoint with the specified number. Blank typein clears breakpoint zero.

**UnbreakAll** removes all breakpoints.

**Probe** refreshes the values in the tile area. Probe is useful when the EtherLink Plus adapter has entered the Slave as a result of a software trap condition and no 3D was present to monitor the Command Register. At a later time, 3D can be executed on the PC with the EtherLink Plus adapter; a Probe then can retrieve the state of the EtherLink Plus adapter when it entered the Slave.

# Typein Area

Most tile operations take the typein as an argument. 3D evaluates arithmetic expressions in the typein using customary operator precedence; parentheses and square brackets can be used freely to group subexpressions. All operations use long (32 bit) arithmetic. Legal operators are addition (+), subtraction (-), multiplication (*), division (/), remainder (%), bitwise and (&), bitwise exclusive or (^), bit wise inclusive or (|), bit wise and with complement (\). The colon (:) operator specifies a memory address. The expression to left of the colon is the segment; the value to the right is the offset. When 3D encounters a register name in a colon expression, the contents of the referenced register are used for evaluating the expression; for example SS:SP refers to the top of the stack, SS:SP+2 refers to the word just below the top of the stack.

All constants are hexadecimal; constants beginning with an alphabetic digit must have a leading zero. 3D predefines all of the 80186 processor registers: AX, BX, CX, DX, SP, BP, ES, DS, CS, IP, and FLAGS. The symbol .IO references registers on the I/O bus; for example, .IO+6 references I/O bus address six. Any expression which evaluates to a constant represents an unsegmented absolute memory location; hence 400 references the same locations as 40:0 and 30:100.

# Using the Mouse to Control the Display

To display a register, move the cursor to the left field of a tile and click the left mouse button. 3D evaluates the expression in the typein as the name of a register or memory location and displays the name contents in the selected tile. To alter the contents of a register move the cursor over the right field of the tile and click the left mouse button. 3D takes the typein, evaluates it, and puts the result in the register.

The other two mouse buttons are also useful. Clicking the right button picks up the text underneath the cursor and appends it to the typein; clicking over an empty tile clears the typein. Clicking the middle button appends a colon to the typein.

Columns of tiles can be cleared or filled by holding down the shift key and clicking the left mouse button. If the cursor is over an empty tile, the tiles below are cleared. If the cursor is over the left field of a filled tile, 3D fills the column below with successive words. If the cursor is over the right field of a filled tile, 3D copies the value under the cursor down the column into the right fields. In all cases, 3D evaluates the typein for the number of tiles to touch; zero or illegal values touch all tiles to the bottom of the column.

# Function Keys

At times, 3D overlays the bottom sixteen lines of the tile area with other information. An inverted square indicates the cursor location; clicking the right mouse button picks up the text under the cursor and places it in the typein. The function keys control what information and the format of data displayed in this region of the tile area.

**F1**

F1 evaluates the typein and disassembles at the specified memory location. If the typein is blank or can't be evaluated, 3D displays the last instructions that were disassembled. 3D caches the last eight screenfuls of disassembled instructions; PgUp can therefore step through previously displayed disassembly. PgDn disassembles the next screenful.

### F2 through F4

F2 through F4 work in a manner similar to F1 except that they display memory as longs (32 bits), shorts (16 bits), or bytes respectively. An additional column to the right displays storage as ASCII text; unprintable characters are printed as a period, ".".

### F5

F5 makes 3D monitor the Command Register and keyboard.

```
*AX         0000   BP      300C   CS          03F0   *IP       6A76
*BX         6892  *SP      3000   SS          3C6D   *FLAGS    F046
 CX         0000   SI      04C4   DS          03F0
 DX         0000  *DI      6806  *ES          03F0

03F0:6A76  9D              popf
03F0:6A77  0B DB           er bx,bx
03F0:6A79  C3              ret
03F0:6A7A  8B 3E 00 68     mov di, 6800
03F0:6A7E  OB FF           or di di
03F0:6A80  78 06           bai 6A88
03F0:6A82  E8 BF FF        call 6A44
03F0:6A85  E8 96 FF        call 6A1E
03F0:6A88  04 00           mov cx, #4
03F0:6A8B  BF 0E 68        mov di, #680E
03F0:6A8E  8B 1D           mov bx, (di)
03F0:6A90  0B DB           or bx, bx
03F0:6A92  75 07           bne 6A9B
03F0:6A94  83 EF 04        sub di, #4
03F0:6A97  E2 F5           loop 6A8E
Boot!  Load!  Go!  Continue!  Stop!  Step!  Break!  Unbreak!  UnbreakAll!
Probe!
cs:ip
popf >> Non maskable interrupt at 3F0:6A76
```

**Figure D-2. Disassembly**

### F6

F6 evaluates the typein as the load address for the Load command in the menu bar. Be sure to select a memory area that is not used by the EtherLink Plus adapter ROM.

### F9

F9 toggles the bottom sixteen lines of the tile area between the serial line display and tiles.

### F10

F10 causes the tiles to appear in the bottom sixteen rows of the tile area.

### [Alt]q

[Alt]q quits.

# Appendix E: Software Diskette

The EtherLink Developer's Software Diskette contains the 3C505 diagnostic program, the 3D Debugger, a collection of utility subroutines and definition files, and an example host program that uses the utilities. The diagnostic is discussed in Appendix C and the 3D Debugger in Appendix D. The utility subroutines are all written in assembly language and implement a range of functions that can be incorporated in resident device drivers or standalone programs.

All assembly language subroutines are coded using the Microsoft Macro Assembler (Version 4.00) conventions. In addition, many subroutines adhere to the segmentation, object code, linkage, and function call conventions required by the Microsoft C Compiler (Version 4.00). If some other compiler is to be used for development, the assembly routines will probably need to be changed accordingly.

The 3C505 EtherLink Developer's Software Diskette contains the following files:

**IO.ASM**
This file contains subroutines that perform programmed I/O to the EtherLink Plus adapter. The subroutine IO_INIT should be called first with the base I/O address of the EtherLink Plus adapter. The subroutines OUTPCB and INPCB implement the Primary Command Block (PCB) protocol described in Chapter 4 of the Software Developer's Manual. There are also subroutines that manage the state of the Host Control Register.

**DMA.ASM**
This file contains all DMA related subroutines and global variables for an IBM PC or IBM AT host. The subroutine DMA_INIT should be called first with the DMA channel number to use (0-3 on the IBM PC plus 5-7 on the IBM AT). DMA transfers and DMA completion interrupts are handled in this module.

**INT.ASM**
This file contains subroutines to handle EtherLink Plus adapter interrupts in an IBM PC or IBM AT host. The subroutine INT_INIT should be called first with the interrupt vector number to use (channel 3-7 are allowed on the IBM PC plus 9-15 on the IBM AT). There are also subroutines to enable, disable, and acknowledge interrupts in this file.

The first level interrupt handler in this file, determines the source of an interrupt as either command or DMA and then passes control to the appropriate interrupt handler in CMD.ASM or DMA.ASM.

**CMD.ASM**
Command Register interrupts are serviced in this module. The subroutine CMD_INIT must be called first to initialize buffer pointers. When a Command Register interrupt is received by the main interrupt handler, CMD_INT is called to read and locally buffer the PCB. CMD_GET can be called to determine if an interrupt has occurred and retrieve the PCB. Although the EtherLink Plus adapter is able to buffer and process multiple commands, the routines presented in CMD.ASM are synchronous, and not designed for this type of operation.

**UTIL.ASM**
This contains general purpose utility subroutines callable from Microsoft C programs, and utilities used by the other ASM files.

**HOSTIO.DEF**
This is the definition file that equates names to the EtherLink Plus adapter registers and to each bit in each register. Equates are also included for the DMA and interrupt controller registers in the IBM PC and IBM AT.

**HOSTIO.H**
C-language definition file, equivalent to HOSTIO.DEF.

**IDP.H**
Definition file that contains the structures for XNS IDP packets.

**ADAPTER.DEF**
This definition file equates names to the EtherLink Plus adapter soft interrupt vectors 80H-88H. Subfunctions for each vector are also defined.

**PCB.DEF**
This is the definition file that contains the assembly language structures for each PCB from Chapter 3.

**PCB.H**
Definition file for C-language programs that contains the structures for each PCB.

**PCBDEM.H**
A special version of PCB.H that has additional PCB's for use by the download program example in DEMO.C.

**DEMO.C**
C-language source file to demonstrate many of the features and programming techniques for the EtherLink Plus adapter. This program was written using Microsoft C 4.0. The program will not function correctly if other EtherLink Plus adapter drivers (i.e., 3Com 3Plus driver eth505.sys) are loaded on the PC. The program must be linked with object files from the ASM files on the diskette.

The program expects a EtherLink Plus adapter in a 16 bit AT slot and configured with I/O address 310, interrupt channel 7 and DMA channel 5. If other values are desired, they can be specified on the command line in the format:

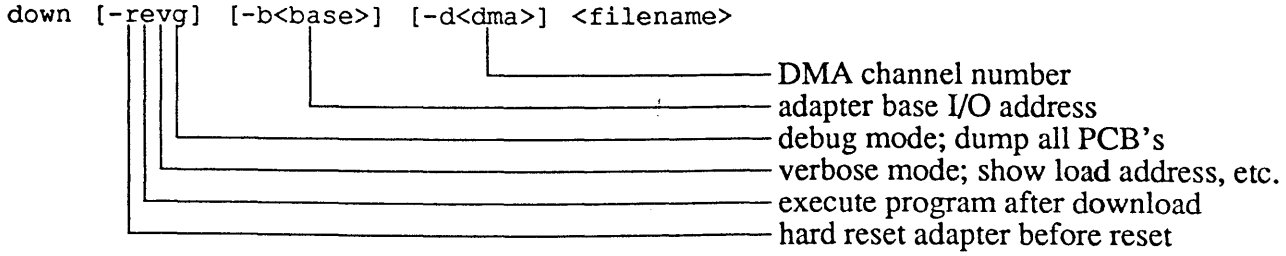
```
DEMO [Int #] [I/O Addr] [DMA #]
```

For example, 'DEMO 3 300 1' would start the program with Interrupt channel 3, I/O address 300 hex, and DMA channel 1.

The program demonstrates initialization, configuration, transmit, receive, and contains a program that can be downloaded onto the adapter which implements a packet filter, allowing the board to receive packets for up to 8 Ethernet addresses. Demo is intended for demonstration only and does not necessarily use efficient techniques, nor has it been extensively tested.

**DLD.ASM**
This module is linked as part of the DEMO program. This is the packet filtering program that can be downloaded and executed on the EtherLink Plus adapter. When the appropriate menu selections are made from the DEMO program, this module is downloaded and executed on the EtherLink Plus adapter. It implements 3 new PCB's that allow the user to select up to 8 Ethernet addresses to be received by the adapter.

**DOWN.C**
This program downloads a DOS COM format file to the EtherLink Plus adapter card. The syntax of usage is:




**DOWN.EXE**
The executable EtherLink Plus adapter download program. See DOWN.C.

**AHTOI.ASM**
Source for a conversion routine that is linked as part of DOWN.C.

**DLD_LED.ASM**
Source for a module that can be downloaded and executed on the EtherLink Plus adapter using the program DOWN. The program causes alternate flashing of LED1 and LED2 on the adapter board.

**DLD_LED.COM**
A demonstration module that can be downloaded to the EtherLink Plus adapter. See DLD_LED.ASM.

**DEMO**
A Microsoft 4.0 makefile that can be used to generate DEMO.EXE from DEMO.C and the other files on the diskette.

**DOWN**
A Microsoft 4.0 makefile that can be used to generate DOWN.EXE and DLD_LED.COM from the files on the diskette.

**DEMO.EXE**
The executable EtherLink Plus adapter demonstration program. See DEMO.C.

**3D.EXE**
This is the 3D Debugger program that controls a special debug mode of the EtherLink Plus adapter to assist in debugging programs running on the card. The 3D Debugger requires a Mouse Systems PC mouse (or equivalent) connected to the host PC's COM2 port. It also assumes that the base I/O address of the EtherLink Plus adapter is set to 310 hex. Refer to Appendix D for information on how to use 3D.

**3C505.EXE**
This is the 3C505 diagnostic utility program file. Appendix C describes in detail how to use it and what it tests.

# Appendix F: Revision 2.0 ROM

## Configure Adapter Memory

In Revision 1.0, certain minimums must be used in the Configure Adapter Memory command as detailed below. In Revision 2.0, the minimums are enforced and the default values are changed.

| | Size | Rev 1.0 Default | Rev 2.0 Default | Min | Notes |
|---|---|---|---|---|---|
| pcb_queue | 64 | 5 | 10 | 2 | e |
| rcv_pcb_queue | 14 | 32 | 20 | 2 | e |
| mc_list | 6 | 0 | 0 | 0 | |
| frame_desc | 22 | 40 | 20 | 2 | a,d |
| rcv_buffer | 1536 | 40 | 20 | 2 | d |
| prog_table | 8 | 10 | 10 | 1 | |
| rcv_buf_desc | 10 | 40 | 20 | 2 | b |

a. The number of Frame Descriptors should normally equal the number of receive buffers.

b. The number of Receive Buffer Descriptors allocated always equals the number of Receive Buffers specified.

c. Revision 2.0 has an additional Receive Buffer Queue (12- bytes per entry). The number of entries in the queue always equals the number of Receive Buffers specified plus 1.

d. The value is the minimum that the firmware will accept but is really not sufficient for network operation. At least 4 Frame Descriptors and Receive buffers should be allocated.

e. One entry in this circular queue is wasted to simplify queue-empty and queue-full checking.

## Timeout Values

The maximum timeout value that can be specified in PCB commands and system ROM service routines is increased in Revision 2.0 to 32767 ticks (10ms per tick) from 127 in Revision 1.0.

# Timestamp and Timer Resolution

Revision 2.0 corrects a problem where the high word of the doubleword timestamp in the receive packet response PCB is wrong. Also, the timer resolution is changed to 15μs per tick instead of 25μs.

# DMA Downloading Programs

The adapter downloads programs using DMA utilities in ROM; i.e., int 80H function 5. This function only initiates the DMA transfer and does not wait for it to complete. In Revision 1.0, a successful download response is sent immediately after DMA initiation without waiting for DMA completion status. Revision 2.0 waits for download complete before sending the response PCB.

# Zero Offset Problem for Downloaded Programs

In Revision 1.0, the first downloaded program is guaranteed to be executed with IP set to zero but subsequent downloaded programs will not. If the IP plus the size of the program's code segment is greater than 0FFFFH, then the program will not work. Revision 2.0 solves this problem.

# Receive/Return Packet functions

In Revision 1.0, successive Receive Packet (Int 81, function 2) calls without a Return Buffer (Int 81, function 3) will always yield the same receive buffer. Also, the Return Buffer call will always attempt to return the first receive buffer in the queue. Revision 2.0 allows any combination of Receive and Return packets.

# PCB Formats

The following PCB's are changed in Revision 2.0:
(Refer to Chapter 3 for more details.)

a. Download Program (PCB 0DH). The offset:segment words have been removed.

b. Download Program Response (PCB 3DH). The offset:segment of the downloaded program is returned.

c. Adapter Info (PCB 11H). This is a new PCB command that retrieves general status information from the EtherLink Plus adapter.

d. Adapter Info Response (PCB 41H). This is a new response PCB that contains information such as revision ID, memory size, and ROM checksum value.

e. Network Statistics Response (3AH). The counters for the number of received and transmitted packets are now double word values.

f. Self-Test Response (3FH). The self-test command now performs ROM checksum, RAM test, and 82586 loopback tests. The response returns error information when any test fails.

# Interrupt Vector Services

The following service routines are changed in Revision 2.0:
(Refer to Chapter 3 for details.)

a. Int 80H, function 0BH: Check DMA Done. This new function allows a program to poll for DMA completion of a DMA initiated with Int 80H function 5 or 7.

b. Int 82H, function 01H: Configure Adapter Memory. New defaults and minimums noted in item #1 of this Appendix are in effect.

c. Int 82H, function 05H: Set LEDs. New function giving downloaded programs ability to turn off/on LEDs without conflicting with ROM usage.

d. Int 82H, function 06H: Get Adapter Info. New function which returns general adapter information such as revision level, amount of memory, free memory pointers, and ROM checksum value.

# TEST Jumper Usage

The state of the TEST jumper on the adapter is represented by the SWTC flag in the Adapter Status Register. When the TEST jumper is set to one, the Revision 2.0 ROM code will:

a. Ignore powerup memory test error. Memory errors detected during powerup normally prevent the adapter from entering the main ROM idle loop. Ignoring errors is useful when using ICE systems that need to modify the NMI vector location in order to operate.

b. Ignore ROM checksum error. During ROM development, it is convenient not to checksum since the code is changing frequently.

c. Install 3D interrupt vectors. The interrupt vectors known as "exceptions" (basically INT 0 to 7) and all unused interrupt vectors are made to point to the 3D slave in the Revision 2.0 ROM. When an exception occurs, 3D will become active and attempt to communicate with the 3D Debugger program.

# Configure 82586 Receive Mode

The Configure 82586 Receive Mode function 2 of the INT 82H service routines is not correct. In the Revision 1.0 implementation, an ES:BX pointer to a Configure 82586 PCB command is expected instead of the receive mode in register BX. In Revision 2.0, this will be fixed to match the documentation.

## Receive Packet PCB Timeout

In the Receive Packet PCB there is a timeout quantity which represents the maximum time to wait for a packet before sending a response PCB. In Revision 1.0, the timeout may never occur depending on the timeout value and the adapter system time at the PCB was processed. To flush the PCB from the receive packet queue, use the soft reset function. This problem will be corrected in Revision 2.0.

## Loopback Mode

The LPBK bit in the Adapter Control Register (described in Chapter 3) controls loopback mode at the 8023 Manchester Code Converter. LPBK is active low. This means that if LPBK is clear, then loopback mode is enabled. Be sure to set LPBK for normal network operation.

# Appendix G: Revision 3.0 ROM

## Adapter Selftest Command

In Revision 2.0, the Adapter Selftest command (PCB 0FH) hangs when testing an adapter with 128KB of memory. A work-around is to load the adapter with 256KB of memory. The Revision 3.0 firmware eliminates this problem and can test the latest revision EtherLink Plus adapter, containing up to 512KB of memory.

## Transmit Packet Command

The Transmit Packet command (PCB 09H) in Revision 2.0, after signaling acceptance of the command, waits up to 30ms for the host to download transmit packet data. Then, the packet is transmitted whether the data download is completed or not. The Revision 3.0 firmware will now allow 50ms instead of 30ms and will not transmit the packet if the download has not completed. If the download is not complete, response PCB 39H is returned to the host with error code -2.

## Get Adapter Information Command

When an adapter-resident program uses INT 82 function 6 to get adapter status information, the data is incorrectly returned in the buffer pointed to by register pair DS:BX. The Revision 3.0 firmware returns the data in the buffer pointed to by register pair ES:BX as documented.

## Packet Processor

One of the functions of the firmware Packet Processor INT 86 is to upload receive packets to the host if there is an outstanding Receive Packet command (PCB 08H). The Revision 2.0 firmware, however, might not upload packets in the same order in which they were received. In Revision 3.0, the order of packets is preserved when being uploaded to the host.

## Adapter LEDs

In the latest revision EtherLink Plus adapter, the bits in the Adapter Control Register (ACR) that control the LEDs are inverted from earlier revision adapters. Downloaded programs that modify the ACR should keep this in mind. The Revision 3.0 firmware detects whether the LED control bits should be inverted and will do so accordingly.

# Power On Self Test (POST)

The first 6KB of RAM is "system" memory needed by the firmware and the rest of RAM is "buffer" memory. In Revision 3.0 if buffer memory errors are found during POST, the firmware will loop infinitely flashing both LEDs at 800 hz. If an 82586 initialization or loopback test error occurs during POST, the firmware will loop flashing the LEDs at 700 hz.

# Appendix H: Firmware Idle Loop

```
;-------------------------------------------------------------------------
;     This procedure is the main loop of the 3C505 ROM routines.
;     It is provided as a reference for the normal sequence of
;     processing and functions used by the 3C505 ROM resident code.
;     Developers writing code that executes on the 3C505 may want
;     to insert their routines in interrupt chains called by this
;     procedure.

;-------------------------------------------------------------------------
;
cmd_processor  proc  near
;-------------------------------------
;this is main pcb processor
;-------------------------------------
      sti                               ;just make sure rupts enabled
      xor         ax,ax
      mov         dx,pic_service
      out         dx,ax                 ;clear all service bits
      mov         bx,offset pcbQparm    ;get top of q pcb
      ;-----------------------------------------------------------------
      ;See if PCB needs processing
      ;-----------------------------------------------------------------
      call        dequeue               ;check for PCB on Q,
                                        ;  if none CY=1
                                        ;  if PCB ok,
                                        ;  ds:bx is pointer, CY=0
      jc          cmd05                 ;branch if no entry in pcb q
      push        es
      mov         ax,ds
      mov         es,ax                 ;setup es
      ;-----------------------------------------------------------------
      ;Process PCB from queue
      ;-----------------------------------------------------------------
      int         85H                   ;call PCB processor
      mov         ax,seg data_seg
      mov         ds,ax                 ;restore ds
      pop         es

cmd05:
      ;-----------------------------------------------------------------
      ;Host commands are normally received via H/W Int 0,
      ;if not, this code will simulate H/W Int 0
      ;(call hcmdf_proc interrupt handler)
      ;-----------------------------------------------------------------
```

```
      mov    dx,cntrl_status
      in     ax,dx                    ;is cmd byte stuck in cmd reg?
      test   ax,ACRF
      jz     cmd00

      mov    cx,100
spin: nop                             ;spin a while
      loop        spin
      mov         dx,cntrl_status
      in          ax,dx               ;is it still there?
      test        ax,ACRF
      jz          cmd00

      ;Next 4 instructions simulate H/W Int 0
      pushf
      push        cs
      push        offset word ptr cmd00
      jmp         far ptr hcmdf_proc  ;Cmd rupt handler
;-------------------------------------------------------------------
;  Notes on hcmdf_proc functionality:
;  first, PCBs are loaded via Int 80 - func 3 into a temp buffer.
;     - a PCB with cmd code 00 is discarded
;     - a PCB with cmd code 01 - 11 is processed as follows:
;           01 - enqueue PCB, Int 88
;           02 - enqueue PCB, Int 88
;           03 - enqueue PCB, Int 88
;           04 - process, Int 80 - 5
;           05 - process, Int 80 - 7
;           06 - process, Int 80 - 6
;           07 - process, Int 80 - 8
;           08 - enqueue recv, Int 86 - 1
;           09 - get pkt data, Int 80 - 5
;                enqueue xmit PCB, Int 88
;           0a - enqueue PCB, Int 88
;           0b - enqueue PCB, Int 88
;           0c - enqueue PCB, Int 88
;           0d - get program, Int 80 - 5
;                enqueue exec PCB, Int 88
;           0e - enqueue PCB, Int 88
;           0f - enqueue PCB, Int 88
;           10 - enqueue PCB, Int 88
;           11 - enqueue PCB, Int 88
;     - a PCB with cmd code > 11 is enqueued, Int 88
;
;-------------------------------------------------------------------
cmd00:
      ;-----------------------------------------------------
      ;See if any packets have been received
      ;-----------------------------------------------------
      mov         ax,2                ;rcv packet function
      mov         dx,0                ;don't wait for packet
      int         81H                 ;call network IO
                                      ;  get addr of nxt pkt
      mov         ax,seg data_seg
      mov         ds,ax               ;restore ds
      jc          cmd10               ;branch if no packet arrived
```

```
        ;-------------------------------------------------------------
        ;if so,
        ;     - and pending rcv, send response PCB and oldest packet
        ;     - queue pkt if no rcv pending or this not oldest
        ;-------------------------------------------------------------
        mov         ax,2                ;Enqueue or return packet func
        int         86H                 ;call packet processor

        mov         ax,seg data_seg
        mov         ds,ax               ;restore ds

cmd10:
        ;-------------------------------------------------------------
        ;Chk pending rcv Q, if packet send PCB and pkt to host;
        ;if not check for timeouts - send PCB if timeout
        ;-------------------------------------------------------------
        mov         ax,3                ;timeout or end for
                                        ;  rcv PCB func
        int         86H                 ;call check rcv pcb

        mov         ax,seg data_seg
        mov         ds,ax               ;restore ds

        cmp         enable_int87, flag_yes ;see if 50ms timer elapsed
        jne         cmd_processor

        ;-------------------------------------------------------------
        ;allow other processes to kickstart
        ;-------------------------------------------------------------
        int         87H                 ;reset timer

        mov         ax,seg data_seg
        mov         ds,ax               ;restore ds
        jmp         cmd_processor

cmd_processor endp
```