

**8500
Modular MDL
Series**

**80186/80188
Emulator
Including
Prototype Control Probes**

This manual supports the
following TEKTRONIX products:


8300E38
8300P45
8300P46
8300P45 Option 01
8300P46 Option 01

**Users and Installation
Manual**

*Please check for change information
at the rear of this manual*

Copyright © 1984 by Tektronix, Inc. All rights reserved.
Contents of this publication may not be reproduced in any
form without the permission of Tektronix, Inc.

Products of Tektronix, Inc. and its subsidiaries are covered
by U.S. and foreign patents and/or pending patents.

TEKTRONIX, TEK, SCOPE-MOBILE, and  are regis-
tered trademarks of Tektronix, Inc. TELEQUIPMENT is a
registered trademark of Tektronix U.K. Limited.

There is no implied warranty of fitness for a particular
purpose. Tektronix, Inc. is not liable for consequential
damages.

Specification and price change privileges are reserved.

Printed in U.S.A.

PREFACE

ABOUT THIS MANUAL

This manual tells how to operate and install the 80186/80188 Emulator in an 8500 Series Tektronix development system. In addition, this manual explains the features of the 8550 and 8540 development systems unique to the 80186 Emulator and 80188 Emulator.

DEFINITION OF TERMS

Throughout this manual, the following terms are used to enhance readability:

- All references to a "development system" refer to:
 - 8550 Microcomputer Development Lab which consists of the 8301 Microprocessor Development Unit, and the 8501 Data Management Unit
 - 8540 Integration Unit
- The term "emulator boards" refers to the 80186/80188 Emulator boards (Board I, Board II, and Board III).
- The term "probe" or "prototype control probe" refers to either the 80186 Prototype Control Probe or 80188 Prototype Control Probe.
- The term "probe plug" refers to the 80186/80188 Prototype Control Probe Plug. The probe plug is a 68-pin chip carrier plug that inserts into the prototype's microprocessor socket.
- The term "emulator" or "emulator processor" refers to the 80186/80188 Emulator boards with either the 80186 Prototype Control Probe or the 80188 Prototype Control Probe.

MANUAL ORGANIZATION

This manual is divided into seven sections:

- Section 1** contains general information about the emulator boards and the prototype control probe.
- Section 2** contains user information that is specifically for the 80186 and 80188 Emulators.
- Section 3** contains an emulator demonstration program.
- Section 4** contains technical information related to the emulator.
- Section 5** defines all jumper and strap locations and describes their function and proper position.
- Section 6** provides installation procedures for the emulator boards and prototype control probe. In addition, provides software installation procedures for development systems.
- Section 7** describes procedures used to verify the functional performance of the 80186/80188 Emulator.

GENERAL INFORMATION

Revision History

As this manual is revised and reprinted, revision history information is included on the text and diagram pages. Existing pages of manuals that have been revised are indicated by REV and date (REV FEB 1984) at the bottom inside corner of the page. New pages added to an existing section, whether they contain old, new, or revised information, contain the word "ADD" and the revision date (ADD FEB 1984).

Change Information

Change notices are issued by Tektronix, Inc., to document changes to the manual after it has been published. Change information is located at the back of this manual, following the yellow tab marked "CHANGE INFORMATION". When you receive this manual, you should enter any change information into the body of the manual, according to instructions on the change notice.

Notational Conventions

In this manual there are instructions to enter development system commands. The following example illustrates one of these commands:

```
$ a1 0 3fff -s
  4 BLOCK(S) ALLOCATED    12 BLOCK(S) FREE
```

These command lines use the following notational conventions:

- The \$ sign represents the development system's prompt.
- All command lines must be entered from your terminal exactly as shown. Be sure to enter any spaces that are shown.
- You must press the RETURN key at the end of each command line.
- The system response to a command is displayed on the next line. System responses are not preceded with the prompt sign.

Hexadecimal Notation

All addresses are in hexadecimal notation, unless otherwise noted. Where necessary for clarity, hexadecimal numbers are defined by an H following the number, such as 43ACH. Other hexadecimal identifiers, such as the prefix or suffix "hex" or subscripts denoting base 16, are not used.

DOCUMENTATION OVERVIEW

To benefit the most from this manual, you should be familiar with your Tektronix development system, and the manuals described in the following paragraphs.

Service Manuals

Service manuals tell how to perform system testing, how to isolate hardware problems, and how to repair system components. Service manuals may be purchased from Tektronix as optional accessories.

The following manuals provide service information for the 80186/80188 Emulator:

- 8301 Microprocessor Development Unit Service Manual
- 8540 Integration Unit Service Manual
- 80186/80188 Emulator Including Prototype Control Probes Service Manual

Installation Manuals

Installation manuals or guides tell how to unpack the equipment, how to install it, and how to verify its proper operation. Installation manuals may be separate manuals, or may be provided as supplements to existing publications. Installation manuals are provided with the equipment as a standard accessory.

The following manuals provide installation information for the 80186/80188 Emulator:

- 8550 Microcomputer Development Lab Installation Guide
- 8540 Integration Unit Installation Guide

Users Manuals

Users manuals describe procedures required to operate the development system and its peripheral devices. Users manuals are provided as a standard accessory to the product.

The following manuals provide user information for the 80186/80188 Emulator:

- 8550 Microcomputer Development Lab System Users Manual
- 8540 Integration Unit System Users Manual
- 8560 Multi-User Software Development Unit System Users Manual

Users and Installation Manuals

These instruction manuals contain both emulator-specific user information and installation information. These manuals tell how to install the emulator hardware and software, and how to verify its proper operation. Users and installation manuals are provided with the equipment as a standard accessory.

The following instruction manuals are available for the 80186/80188 Emulator:

- 80186/80188 Emulator Including Prototype Control Probes Users and Installation Manual

CONTENTS

	Page
Operators Safety Summary	xiii
Servicing Safety Summary	xv

Section 1 GENERAL INFORMATION

Introduction	1-1
Modes Of Operation	1-1
Emulator Overview	1-3
Microprocessors Supported	1-3
Emulator Hardware Configuration	1-3
Clock Rate	1-3
Emulator Boards	1-4
Prototype Control Probe	1-4
Prototype Control Probe LEDs	1-6

Section 2 EMULATOR-SPECIFICS OPERATOR INFORMATION

Introduction	2-1
Symbolic Debug	2-1
Byte/Word Parameter	2-1
Register Set	2-1
Status Word Register	2-3
Operating System Commands	2-6
AL--Allocate Physical Memory	2-6
BK--Set or Display Breakpoint Conditions	2-7
D, F, LO, MOV, P, and SAV--Memory Manipulation Commands	2-8
DEAL--Deallocate Physical Memory	2-8
DI--Disassemble Object Code into Mnemonics	2-9
DS--Display Status of Emulator Registers	2-9
G--Begin Program Execution	2-15
LPCB--Locate Peripheral Control Block	2-16
MAP--Set or Display Memory Map Assignments	2-17
MEM--Specify Available User Memory	2-17
MEMSP--Defines Memory Space	2-17
NOMEM--Specify Unavailable User Memory	2-17
RD--Read From Emulator I/O Port	2-18
RESET--Reinitialize Emulator	2-19
S--Assign Value to Register or Symbol	2-19
SEL--Select the Emulator	2-20
Selecting an Assembler	2-20
SPCB--Set Peripheral Control Block	2-21
TRA--Control Display of Executed Instructions	2-22
WRT--Write to Emulator I/O Port	2-24
X--Load and Execute Program	2-24

Section 2 EMULATOR-SPECIFICS OPERATOR INFORMATION (Cont.)

	Page
Real-Time Prototype Analyzer (RTPA)	2-24
Trigger Trace Analyzer (TTA) Commands	2-25
BUS and EVE--TTA Bus Operation Designators	2-25
CONS--Set Consecutive Events	2-25
DISP--Display Contents of TTA's Acquisition Memory	2-25
Special Considerations	2-28
The HALT Instruction	2-28
Losing Clock Causes Emulator Fault Error	2-28
Interrupt Status Register Constraints	2-28
PROM Programmer Considerations	2-29
Emulating the Execution Bus	2-29
Prototype Interrupts During Trace All Mode	2-29
DMA Activity During Trace All Mode	2-29
TTA Display Changes During DMA Activity	2-29
Segment Registers	2-30
CSX, DSX, SSX and ESX Symbols	2-30
Service Calls	2-31
SVCs in Modes 1 and 2	2-32
SRB Format	2-32
SVC Demonstration	2-32
Error Messages	2-37

Section 3 EMULATOR DEMONSTRATION RUN

Introduction	3-1
Demonstration Program	3-1
Examine the Demonstration Program	3-1
Set Table Pointer	3-2
Set Pass Counter	3-3
Clear Accumulator	3-3
Add Byte from Table	3-3
Point to Next Byte	3-3
Decrement CX and Loop Until CX = Zero	3-3
Call Exit SVC	3-3
Development System Configurations	3-3
Assemble and Load the Demonstration Program	3-5
Case 1: Assemble and Load on the 8550	3-6
Start Up and Log In	3-6
Copy the Demonstration Run Program from the Installation Disk ...	3-7
Examine the Demonstration Program	3-7
Assemble the Source Code	3-8
Link the Object Code	3-9
Select the 80186/80188 Emulator	3-10
Load the Program into Memory	3-10
Allocate Memory	3-10
Zero Out Memory	3-10
Check that Memory is Filled with Zeros	3-11
Load the Object Code into Memory	3-11
Load the Program Symbols	3-11

Section 3 EMULATOR DEMONSTRATION RUN (Cont.)

	Page
Case 2: Assemble on the 8560; Download to the 8540 or 8550	3-12
Start Up and Log In	3-12
Create the Demonstration Program	3-13
Enter the Text	3-13
Check for Errors	3-14
Assemble the Source Code	3-16
Link the Object Code	3-17
Download the Program to the 8540	3-17
Allocate Memory	3-17
Zero Out Memory	3-17
Check that Memory is Filled with Zeros	3-18
Download the Object Code	3-18
Download the Program Symbols	3-18
Case 3: Download from Your Host to the 8540 or 8550	3-19
Create the Extended Tekhex Load Module	3-20
Start Up and Log In	3-21
Allocate Memory	3-21
Zero Out Memory	3-21
Check that Memory is Filled with Zeros	3-23
Download the Load Module	3-23
Case 4: Patch the Program into Memory	3-24
Start Up and Log In	3-24
Allocate Memory	3-24
Zero Out Memory	3-24
Check that Memory is Filled with Zeros	3-25
Patch the Object Code into Memory	3-25
Put Symbols into the Symbol Table	3-26
Run the Demonstration Program	3-26
Check Memory Contents Again	3-27
Turn On Symbolic Display	3-27
Disassemble the Object Code	3-27
Put Values into the Table in Memory	3-29
Check the Contents of the Table	3-29
Check the Code Segment Register	3-30
Start Program Execution	3-31
Monitor Program Execution	3-31
Trace All Instructions	3-31
Trace to the Line Printer	3-34
Trace Jump Instructions Only	3-34
Set a Breakpoint after a Specific Instruction	3-37
Set Breakpoints to Stop Program Execution	3-37
Set New Values in Pass Counter and Table Pointer;	
Check Results	3-38
Resume Program Execution	3-39
Summary of 80186 Emulator Demonstration Run	3-40
Delete the Demonstration Run Files	3-40
Delete 8550 Files	3-41
Delete 8560 Files	3-41
Turn Off Your System	3-41

WARNING

The following servicing instructions are for use by qualified personnel only. To avoid personal injury do not perform any servicing other than contained in operating instructions unless you are qualified to do so.

Section 4 TECHNICAL INFORMATION

	Page
Introduction	4-1
Emulator Timing	4-1
Probe/Prototype Interface Diagram	4-10
Probe Power Supply Calibration	4-10
Equipment Required	4-10
Procedure	4-10
Specifications	4-13

Section 5 JUMPERS

Introduction	5-1
Emulator Boards Jumpers and Straps	5-1
Board I Jumpers and Strap	5-2
P1086	5-2
W4037	5-2
Board II Jumpers	5-3
P1011	5-3
P2089	5-3
P6102	5-3
P7105	5-3
Prototype Control Probe Jumpers and Straps	5-5
Control Board Jumpers	5-5
P3021	5-5
P4021	5-5
P4023	5-5
Buffer Board Jumpers	5-6
P4023 and P6023	5-7
P5023	5-7
P7011	5-7
P7023	5-7
P8027	5-8
Power Supply Board Jumper	5-9
P5061	5-10
CPU Board Jumpers	5-11
P2067	5-11
P2068	5-11
Dummy Address Jumpers	5-11
P6042	5-12
P6043 and P6044	5-12

Section 5 JUMPERS (Cont.)

	Page
Development System Jumpers and Straps	5-13
32K Program Memory Board Jumper Adjustments	5-14
64K/128K Program Memory Board Jumper Adjustments	5-14
80186/80188 and Development System Jumper Summary	5-15

Section 6 INSTALLATION

Introduction	6-1
Hardware Installation	6-1
Installing the Emulator Boards and Prototype Control Probe	6-1
Connecting to the Prototype	6-9
Probe Plug Insertion Procedure	6-9
Prototype Control Probe Disassembly Procedure	6-11
Interface Assembly and Disassembly Procedure	6-12
Top Cover Disassembly Procedure	6-12
Bottom Cover Disassembly Procedure	6-13
Connecting to the Trigger Trace Analyzer (Optional)	6-14
Grounding	6-15
Software Installation	6-16
Setting up the 8560	6-16
Setting Your Shell Variable	6-16
Installing the 8540 Firmware	6-16
Installing the 8550 Software	6-19
Start Up and Set the Date	6-20
Installation Procedure	6-21
Installing the Diagnostic Software	6-22
Using Your 8086/8088 Assembler on the 8550	6-22

Section 7 VERIFICATION PROCEDURES

Introduction	7-1
Equipment Required	7-1
Functional Test Procedures	7-2
Equipment Setup	7-2
Test Procedure	7-2
No HELLO Display	7-3
Processor Test Procedure	7-4
Emulator Timing Verification	7-5
Measurement Considerations	7-5
Equipment Required	7-5
Controlling the Signal Lines Under Test	7-6
System Functional Verification	7-6
8550 Microcomputer Development Lab Verification	7-6
8540 Integration Unit Verification	7-8

CHANGE INFORMATION

ILLUSTRATIONS

Fig. No.		Page
1-1	Emulation modes 0, 1, and 2	1-2
1-2	80186/80188 Emulator interconnections	1-5
2-1	80816/80188 register set	2-2
2-2	80186/80188 Status Word (FLAGS) Register	2-3
2-3	Relocation Register content	2-12
2-4	80186/80188 SVC demonstration program listing	2-34
3-1	80186 demonstration run program	3-2
3-2	Development system configurations	3-4
3-3	Host computer commands for preparing demonstration program ...	3-20
3-4	80186 demonstration run program: extended Tekhex format	3-22
4-1	80186 RMX Mode timing diagram	4-5
4-2	Clock timing diagram	4-7
4-3	HOLD-HOLDA timing diagram	4-8
4-4	80186 Timer timing diagram	4-9
4-5	Probe Power Supply Board	4-12
5-1	Board I jumper and strap locations	5-2
5-2	Board II jumper and indicator locations	5-4
5-3	Control Board jumper locations	5-6
5-4	Buffer Board jumper and fuse locations	5-9
5-5	Power Supply Board jumper and indicator locations	5-10
5-6	CPU Board jumper locations	5-13
6-1	Removal/installation of the top cover	6-2
6-2	80186/80188 Emulator board interconnections	6-3
6-3	Recommended board arrangement for the 8301	6-4
6-4	Recommended board arrangement for the 8540	6-5
6-5	Strain relief plate installation/removal	6-6
6-6	Ribbon cable installation	6-7
6-7	Emulator boards with prototype control probe	6-8
6-8	Pin identification and proper plug insertion	6-11
6-9	80186/80188 Emulator Board I connections to the TTA	6-15
6-10	System ROM Board socket locations	6-17
6-11	Type-2764 and type-27128 ROM strapping	6-19

TABLES

Table

No.		Page
2-1	80186/80188 Status Word (FLAGS) Bit Functions	2-4
2-2	80186/80188 Registers and Flags	2-5
2-3	Peripheral Control Block--DMA, Chip-Select, and Timer Registers	2-14
2-4	Peripheral Control Block--Interrupt Control Registers	2-15
2-5	80186/80188 TTA Bus Operation Designators	2-25
2-6	80186/80188 Service Calls	2-33
3-1	Basic 8560 Editing Commands	3-15
4-1	80186 Emulator Timing Differences (RMX Mode)	4-2
4-2	80186/80188 Emulator Electrical Characteristics	4-13
4-3	80186/80188 Emulator Environmental Characteristics	4-13
4-4	80186/80188 Emulator Physical Characteristics	4-14
5-1	Ready Fault/Timeout Jumpers	5-8
5-2	Dummy Address Jumpers	5-12
5-3	80186/80188 Jumper Default Position Summary	5-15
5-4	System Jumper Default Position Summary	5-16

OPERATORS SAFETY SUMMARY

The general safety information in this part of the summary is for both operating and servicing personnel. Specific warnings and cautions will be found throughout the manual where they apply, but may not appear in this summary.

TERMS

In This Manual

CAUTION statements identify conditions or practices that could result in damage to the equipment or other property.

WARNING statements identify conditions or practices that could result in personal injury or loss of life.

As Marked on Equipment

CAUTION indicates a personal injury hazard not immediately accessible as one reads the marking, or a hazard to property including the equipment itself.

DANGER indicates a personal injury hazard immediately accessible as one reads the marking.

SYMBOLS

As Marked on Equipment



DANGER high voltage.



Protective ground (earth) terminal.



ATTENTION - Refer to manual.

SAFETY PRECAUTIONS**Grounding the Product**

The product is grounded through the grounding conductors in the interconnecting cables. To avoid electrical shock, plug the supporting system's power cord into a properly wired receptacle. A protective ground connection by way of the grounding conductor in the system power cord is essential for safe operation.

Use the Proper Fuse

To avoid fire hazard, use only the fuse specified in the parts list for your product. Be sure the fuse is identical in type, voltage rating, and current rating.

Refer fuse replacement to qualified service personnel.

Do Not Operate in Explosive Atmospheres

To avoid explosion, do not operate this product in an atmosphere of explosive gases.

Do Not Remove Covers or Panels

To avoid personal injury, do not remove covers or panels from this product. Do not operate the product without the covers and panels properly installed.

SERVICING SAFETY SUMMARY

FOR QUALIFIED SERVICE PERSONNEL ONLY

(Refer also to the preceding Operators Safety Summary)

DO NOT SERVICE ALONE

Do not perform internal service or adjustment on this product unless another person capable of rendering first aid and resuscitation is present.

USE CARE WHEN SERVICING WITH POWER ON

Dangerous voltages exist at several points in this product. To avoid personal injury, do not touch exposed connections and components while power is on.

Disconnect power before removing protective panels, soldering, or replacing components.

POWER SOURCE

The system that supports this product is intended to operate from a power source that will not apply more than 250 volts rms between the supply conductors or between either supply conductor and ground. A protective ground connection by way of the grounding conductor in the supporting system's power cord is essential for safe operation of this product.

Section 1

GENERAL INFORMATION

INTRODUCTION

The 80186/80188 Emulator is an option designed to function as part of the 8500 Series development systems. The 80186/80188 Emulator provides in-circuit emulation for an 80186 or an 80188 microprocessor.

This instruction manual is divided into two parts. The first part contains specific user and operating information for the 80186/80188 Emulator. The second part contains procedures for emulator hardware and control software installation. In addition, the second part contains procedures for performing product verification for the emulator in either an 8540 or an 8550 development system. Installation and verification procedures are contained in Sections 6 and 7 of this manual. For information about the development system, refer to your System Users Manual. Servicing information is contained in the 80186/80188 Emulator Including Prototype Control Probes Service Manual.

This section explains the emulation modes and gives an overview of the emulator.

CAUTION

The 80186 or 80188 microprocessor contained within the prototype control probe is extremely static-sensitive. A static discharge can interrupt the normal operation of the emulator. Avoid touching the prototype control probe cables and 68-pin probe plug, while the emulation system is operating. You may need an anti-static wrist strap while operating this equipment.

MODES OF OPERATION

The 80186/80188 Emulator operates in one of three emulation modes:

Mode 0 System mode. Use mode 0 to develop software for a prototype circuit based on an 80186 or 80188 microprocessor. In mode 0, the development system acts as a stand-alone, 80186- or 80188-based microcomputer. Thus, the prototype software is stored in your development system's program memory, and the 80186/80188 Emulator provides the I/O facilities and clock. Figure 1-1A illustrates mode 0 operation.

Mode 1 Partial emulation mode. Use mode 1 to develop some of the prototype circuit's hardware functions. In mode 1, the prototype control probe is connected between the 80186/80188 Emulator boards and the microprocessor socket in the prototype circuit. Mode 1 exercises the prototype's memory, I/O, and clock while the development system retains full control over the prototype system.

In mode 1, the memory mapping feature allows portions of the prototype software to be stored in prototype memory. The rest of the prototype software remains in the development system's program memory. Figure 1-1B illustrates mode 1 operation.

Mode 2 Full emulation mode. Use mode 2 in the final software/hardware integration phases of prototype system design. You may also use this mode when you are troubleshooting hardware with known-good software. The only difference between mode 1 and mode 2 is that in mode 2 the entire prototype program is stored in the prototype's memory. Figure 1-1C illustrates mode 2 operation.

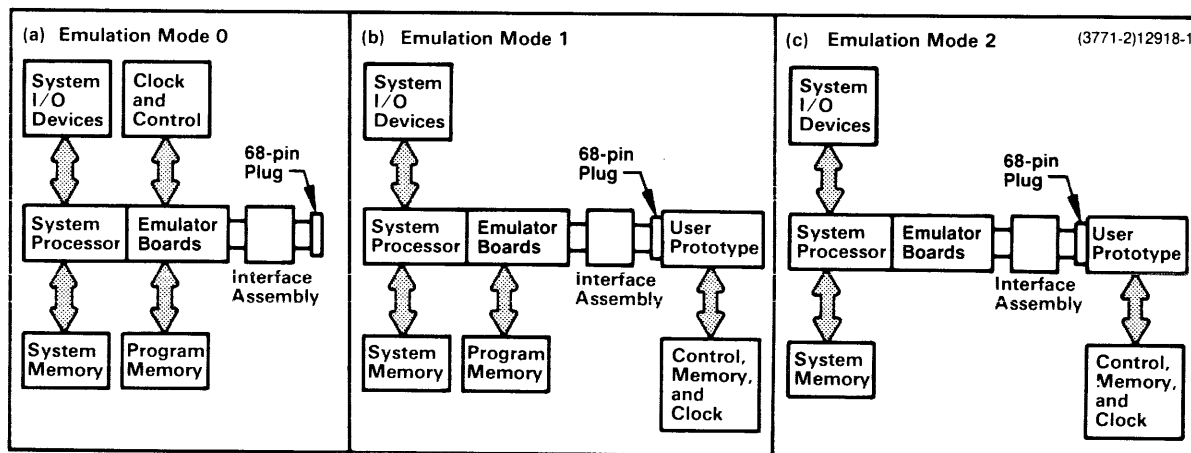


Fig. 1-1. Emulation modes 0, 1, and 2.

EMULATOR OVERVIEW

The 80186/80188 Emulator consists of three emulator boards (Board I, Board II, and Board III) and either an 80186 Prototype Control Probe or an 80188 Prototype Control Probe. The three emulator boards plug into the development system's mainframe, and the prototype control probe attaches to Emulator Board II.

The 80186/80188 Emulator serves two purposes in the microcomputer development system. First, the emulator can run a program written specifically for the target microprocessor. With the help of other boards in the system, the emulator can check the program for run-time errors or program-logic errors. Second, with the prototype control probe connected between the emulator boards and the prototype circuit, the prototype circuit can be debugged and stepped through the final stages to design completion.

The 80186/80188 Emulator emulates the operation of a target microprocessor device for the final version of a prototype system. The emulator responds to software the same way the target microprocessor responds and also allows software debugging.

MICROPROCESSORS SUPPORTED

The 80186 Emulator emulates the Intel 80186 microprocessor.

The 80188 Emulator emulates the Intel 80188 microprocessor.

The 80186 and 80188 microprocessors differ only in their data bus width. The 80186 uses either a 16-bit or an 8-bit data bus width, and the 80188 uses an 8-bit data bus width only.

EMULATOR HARDWARE CONFIGURATION

In emulation mode 0, the prototype control probe is connected to the emulator boards. Your program runs in 8550 or 8540 program memory. In emulation modes 1 and 2, the prototype control probe must be connected to both the emulator boards and your prototype. For instructions on installing your emulator and probe, refer to Section 6 of this manual.

CLOCK RATE

The input frequency to the 80186/80188 Emulator Clock (used in emulation mode 0) is 8 MHz or 16 MHz. The maximum input frequency from the prototype's clock (used in emulation modes 1 and 2) is 16 MHz.

EMULATOR BOARDS

The three 80186/80188 Emulator boards plug into the Main Interconnect Board of the development system's mainframe. These boards are designated as Board I, Board II, and Board III. The functions of each board are as follows:

- Board I contains breakpoint circuitry, clocks, address and data strobes, and the TTA connectors.
- Board II contains the address latch and buffer, mainframe interface timing, mapping RAM, relocation RAM, and the prototype control probe interface.
- Board III contains the data bus interface, PROM, function RAM, command port, status port, queue, and master register.

PROTOTYPE CONTROL PROBE

The prototype control probe allows an emulator to functionally replace a prototype system's microprocessor. The probe is the interface between the emulator boards and the prototype circuit's microprocessor socket.

The prototype control probe consists of:

1. Two 6-foot, 40-conductor ribbon cables that attach to Emulator Board II.
2. An interface assembly, which contains the Buffer Board, Control Board, CPU Board, and Power Supply Board.
3. A 68-pin chip carrier plug that inserts into the prototype circuit board's microprocessor socket. The plug is connected to the interface assembly by four 20-conductor, 1-foot ribbon cables.

Figure 1-2 shows the 80186/80188 Emulator interconnections.

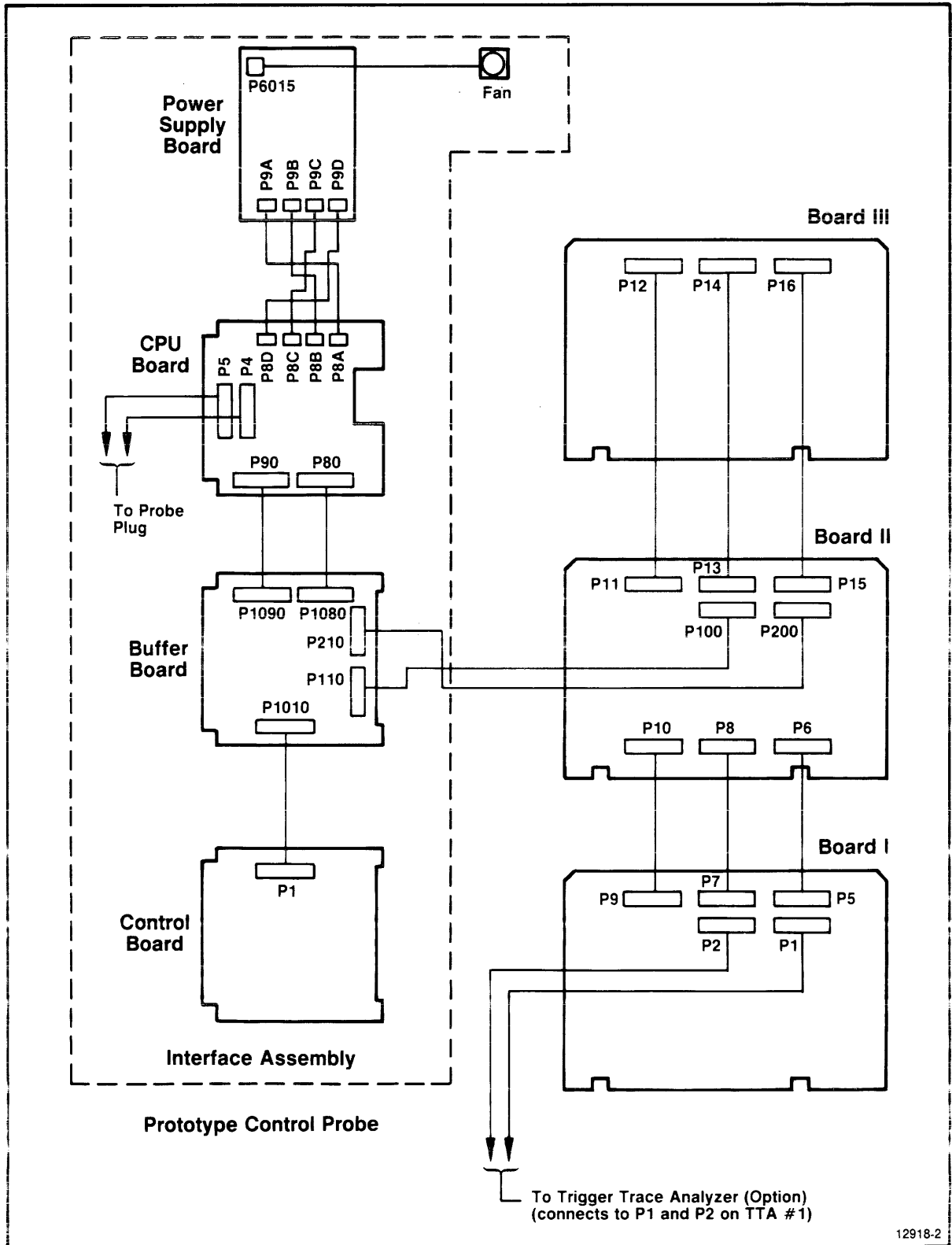


Fig. 1-2. 80186/80188 Emulator interconnections.

Prototype Control Probe LEDs

Six indicator LEDs are visible on the prototype control probe's interface assembly. When lit, these LEDs indicate the following conditions:

RESET	The prototype circuit has asserted the RESET line and caused the processor to stop current activity.
TEST	The prototype circuit has asserted the TEST line. (This LED is always lit in mode 0.)
WAIT	The processor is in a wait state.
HOLD ACK	The processor is acknowledging a bus request.
SELECT	The emulator has been selected.
PROTOTYPE POWER	The prototype circuit's +5 V supply is present.

Section 2

EMULATOR-SPECIFICS OPERATOR INFORMATION

INTRODUCTION

This section explains the features of the 8550 and 8540 systems unique to the 80186 Emulator and the 80188 Emulator. Throughout this section, the phrase "your System Users Manual" refers to your 8550 System Users Manual or 8540 System Users Manual.

SYMBOLIC DEBUG

The 80186/80188 Emulator supports the use of symbolic debug. Many displays in this section include symbolic debug information. Refer to the emulation section of your System Users Manual for information on symbolic debug.

BYTE/WORD PARAMETER

Several commands offer you the choice of operating in memory on a byte-oriented or word-oriented basis. For those commands that permit a byte/word parameter, the parameter is represented by a **-b** (byte) or **-w** (word). For the 80186/80188 Emulator, the default value is **-b**.

REGISTER SET

The basic architecture for the 80186 or 80188 microprocessor device contains fourteen 16-bit special registers that are divided into three functional categories. Figure 2-1 shows the register name, category, and special function of each register.

General	Eight 16-bit general purpose registers store arithmetic and logical operands. Four of these registers (AX, BX, CX, and DX) are configured as 16-bit registers, or each register can be divided into two separate 8-bit registers.
Segment	Four 16-bit segment registers address four segments of memory with each segment containing 64K bytes of addressable memory. The memory segments are addressed by adding a 16-bit offset to the 16-bit address in one of the segment registers. This permits a physical address size up to 1 MByte.

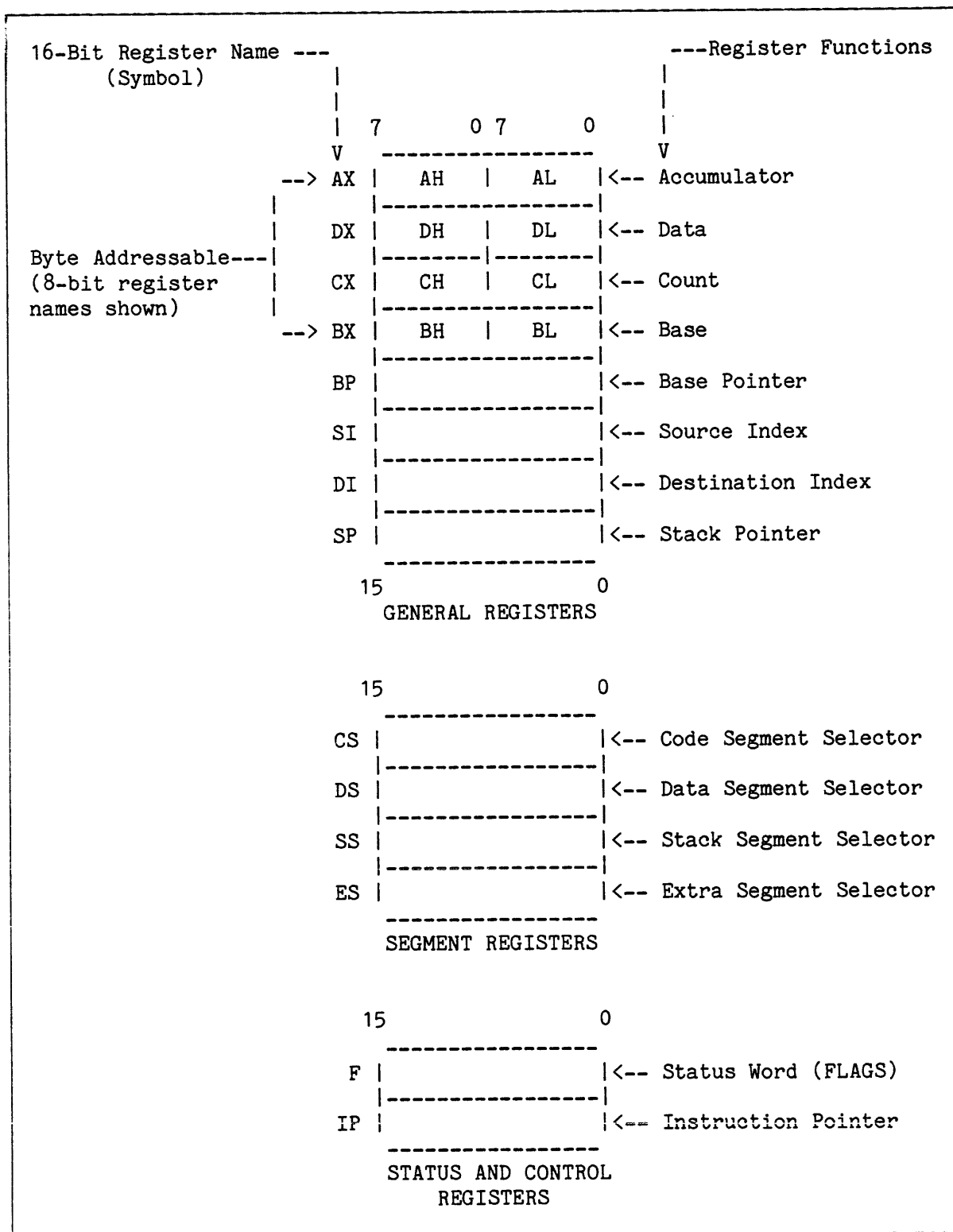


Fig. 2-1. 80816/80188 register set.

Status and Control Two 16-bit special purpose registers provide status and control. The Status Word (FLAGS) Register contains the status and control flag bits. (See Fig. 2-2.) The Instruction Pointer Register contains the offset address of the next sequential instruction to be executed.

STATUS WORD REGISTER

Figure 2-2 shows the format of the 80186/80188 Status Word (FLAGS) Register. Table 2-1 describes the Status Word bit functions.

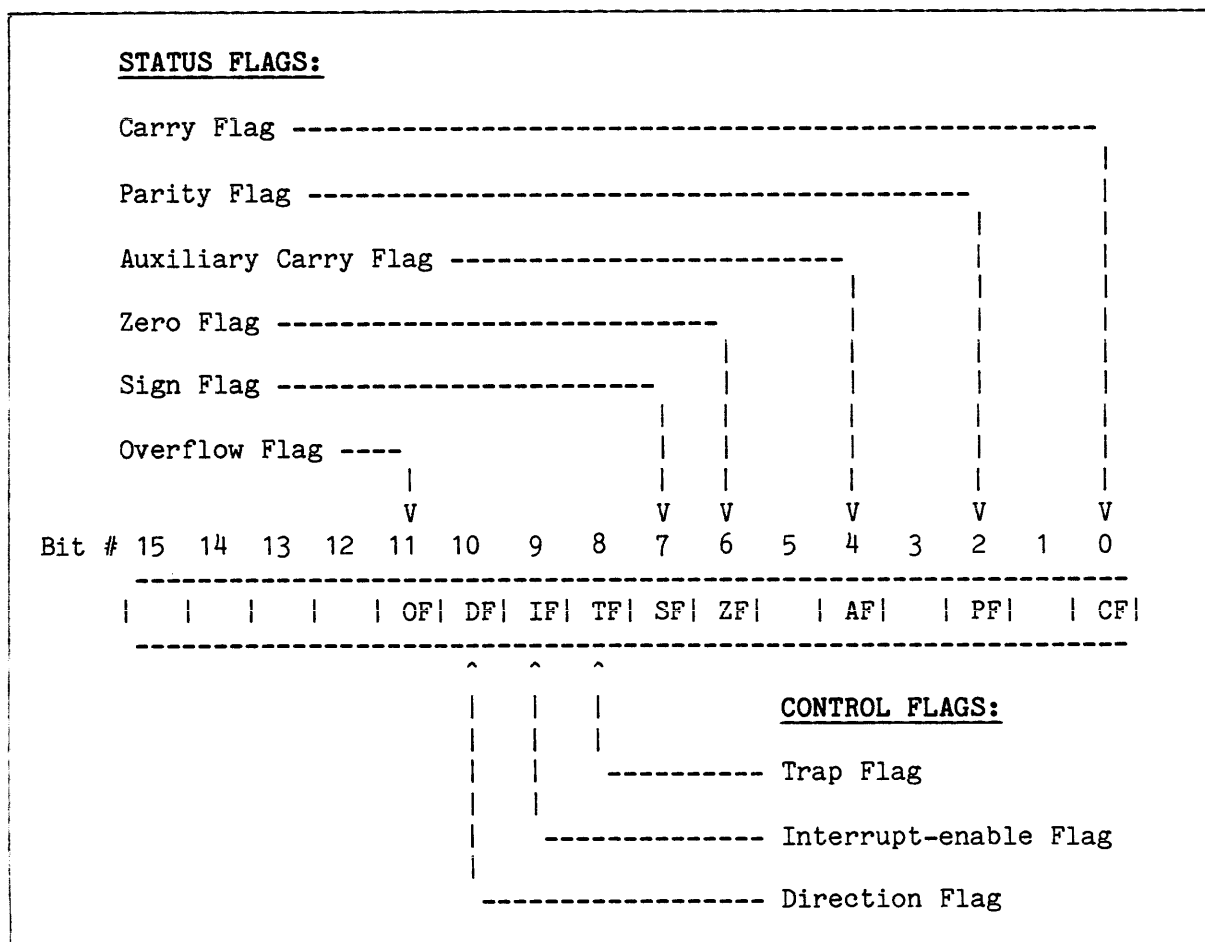


Fig. 2-2. 80186/80188 Status Word (FLAGS) Register.

Table 2-1
80186/80188 Status Word (FLAGS) Bit Functions

Status Bit	Symbol	Function
0	CF	Carry Flag--Set to 1 for a high-order bit carry or borrow. Cleared (zero) at other times.
2	PF	Parity Flag--Set to 1 if the low-order result (8 bits) contains an even number of 1-bits. Cleared (zero) at other times.
4	AF	Auxiliary Carry Flag--Set to 1 on carry from or borrow to the low-order four bits of AL. Cleared (zero) at other times.
6	ZF	Zero Flag--Set to 1 if the result is zero. Cleared (zero) at other times.
7	SF	Sign Flag--Set equal to the result's high-order bit. Set to 0 if positive and to 1 if negative.
8	TF	Trap (Single Step) Flag-- When this flag is set to 1, a single step interrupt occurs after the next instruction executes. Cleared by the single step interrupt.
9	IF	Interrupt-enable Flag--When this flag is set to 1, maskable interrupts cause the processor to transfer control to a specified interrupt vector location.
10	DF	Direction Flag--When set to 1, this flag causes string instructions to automatically decrement the appropriate Index Register. When this flag is clear (zero), the Index Register is automatically incremented.
11	OF	Overflow Flag--Set to 1 if the signed result cannot be expressed within the number of bits in the destination operand. Cleared (zero) at other times.

The register and flag symbols shown in Figs. 2-1, 2-2, and Table 2-1 are used by DOS/50 and OS/40 to designate a specific register or flag used with the 80186/80188 Emulator. Table 2-2 alphabetically lists these register and flag symbols and provides a brief description of each symbol.

Table 2-2
80186/80188 Registers and Flags

Symbol	Description	Size in Bits	Value After RESET (a)	Altered by s Command?
AF	Auxiliary Carry Flag (b)	1	0	yes
AH	High-order byte of Register A	8	NC	yes
AL	Low-order byte of Register A	8	NC	yes
AX	Register A	16	NC	yes
BH	High-order byte of Register B	8	NC	yes
BL	Low-order byte of Register B	8	NC	yes
BP	Base Pointer Register	16	NC	yes
BX	Register B	16	NC	yes
CF	Carry Flag (b)	1	0	yes
CH	High-order byte of Register C	8	NC	yes
CL	Low-order byte of Register C	8	NC	yes
CS	Code Segment Register	16	FFFF	yes
CX	Register C	16	NC	yes
DF	Direction Flag (b)	1	0	yes
DH	High-order byte of Register D	8	NC	yes
DI	Destination Index Register	16	NC	yes
DL	Low-order byte of Register D	8	NC	yes
DS	Data Segment Register	16	0000	yes
DX	Register D	16	NC	yes
ES	Extra Segment Register	16	0000	yes
FLAGS	Flags Register (b)	16	0000	yes
IF	Interrupt-Enable Flag (b)	1	0	yes
INTR (c)	Interrupt Request Input	1	NC	no
IP	Instruction Pointer	16	0000	no
OF	Overflow Flag (b)	1	0	yes
NMI (c)	Non-Maskable Interrupt Input	1	NC	no
PF	Parity Flag (b)	1	0	yes
SF	Sign Flag (b)	1	0	yes
SI	Source Index Register	16	NC	yes
SP	Stack Pointer Register	16	NC	yes
SS	Stack Segment Register	16	0000	yes
TEST (c)	Test control for WAIT instruction	1	NC	no
TF	Trap Flag (b)	1	0	yes
ZF	Zero Flag (b)	1	0	yes

a NC = not changed by **reset** command.

b The FLAGS Register is illustrated in Fig. 2-2.

c INTR, NMI, and TEST are hardware inputs to the microprocessor.

OPERATING SYSTEM COMMANDS

Several 8550 and 8540 system operation commands have unique features or operate differently when used with the 80186/80188 Emulator. The **memsp** (memory space) system command is not implemented for the 80186/80188 Emulator. In addition, two new emulator specific commands, **spcb** (set Peripheral Control Block) and **lpcb** (locate Peripheral Control Block), are added to the system commands for the 80186/80188 Emulator. The system commands that the 80186/80188 Emulator supports are described in the following pages. For additional information on the 8540/8550 system commands refer to your System Users Manual.

AL--ALLOCATE PHYSICAL MEMORY

The 80186/80188 Emulator supports the **al** (allocate) command. However, memory segment information is not valid. Use the **s** (set) command to assign appropriate values to the segment registers.

The **al** (allocate) command allocates 4K-byte blocks of program memory for your program's logical addresses. The 80186/80188 Emulator supports the **-f** and **-s** dash modifiers, which specify fast or slow memory, respectively. The default condition is fast memory (no wait states are inserted during program memory accesses). The **-s** modifier is never required for emulator operation but is available for prototype simulation purposes. The **-s** modifier is used when jumpers P6102 and P7105 (on Emulator Board II) have been set to support it. These jumpers are discussed in Section 5 of this manual. Deallocating memory does **not** remove the "slow" designation: the **-s** modifier remains in effect until changed by a **-f** modifier.

Example:

The following commands allocate memory and display the allocations:

```
> al 0 3fff
  4 BLOCK(S) ALLOCATED   12 BLOCK(S) FREE

> al 8000 -s
  1 BLOCK(S) ALLOCATED   11 BLOCK(S) FREE

> al
  00000 - 03FFF
  08000 - 08FFF S

  5 BLOCK(S) ALLOCATED   11 BLOCK(S) FREE
```

BK--SET OR DISPLAY BREAKPOINT CONDITIONS

The 80186/80188 Emulator allows you to set up to three breakpoints. These emulators support "arming" breakpoints with the `-a` modifier. When breakpoints are armed, the breakpoint conditions must occur in sequence. The actual break and its trace line occur only when the final breakpoint condition is met.

You can program any of three conditions: BK1 to arm BK2, BK2 to arm BK3, or BK1 to arm BK2 to arm BK3. To use this arming feature for the condition BK1 to arm BK2, first program BK1, then program BK2 and include the `-a` parameter. To clear the arming feature, you must clear the breakpoint that contains the `-a` parameter.

NOTE

With a breakpoint set and using "trace all" to monitor the execution of the program, the display sometimes shows the displayed breakpoint to be several instructions past the actual breakpoint parameters. This discrepancy occurs because the instructions are prefetched before they are executed. The difference in the displayed breakpoint and the actual breakpoint depends on the amount of instructions in the queue.

The 80186/80188 Emulator also allows you to specify read (`rd`), write (`wt`) memory (`m`) and I/O (`i`) operations with the `bk` command.

Example:

The following breakpoints set the emulator to break on any I/O read that follows a memory write to address 1C0F0:

```
> bk 1 1c0f0 m wt
> bk 2 ,,i rd -a
```

NOTE

Addresses 8, 9, A, and B are the vectors that point to a Non-Maskable Interrupt (NMI). These addresses are used by the operating system whenever a break is generated. If you set a breakpoint at one of these addresses (for example, to test your program's NMI routine), any system or user interrupt that occurs will generate a break, not just an NMI.

D, F, LO, MOV, P, AND SAV--MEMORY MANIPULATION COMMANDS

With the **d** (dump) command, and other commands that manipulate memory (**f**, **lo**, **mov**, **p**, **sav**), the **loadr** parameter may be either an absolute address or an address relative to the current Code Segment (CS). For example, if CS=0100, the following commands are equivalent:

```
> d 1020
      0 1 2 3 4 5 6 7 8 9 A B C D E F
001020 12 34 56 78 90 00 00 00 00 00 00 00 00 00 00 00 .4Vx.....

> d CS:20
      0 1 2 3 4 5 6 7 8 9 A B C D E F
000020 12 34 56 78 90 00 00 00 00 00 00 00 00 00 00 00 .4Vx.....
```

NOTE

When you specify an address relative to the current CS, the display includes only the offset value and does not include the CS information.

DEAL--DEALLOCATE PHYSICAL MEMORY

The **deal** (deallocate) command removes the previous allocation of program memory. This command operates on 4K-byte blocks of program memory. The **-a** command modifier deallocates all of program memory. Memory segment information is not valid with the **deal** command. The **deal** command does **not** affect the conditions set with the **al** command modifiers **-f** and **-s**.

Example:

The following are examples of **al** and **deal** commands:

```
> al 0 3fff -s
4 BLOCK(S) ALLOCATED 12 BLOCK(S) FREE

> deal -a
8 BLOCK(S) ALLOCATED 8 BLOCK(S) FREE

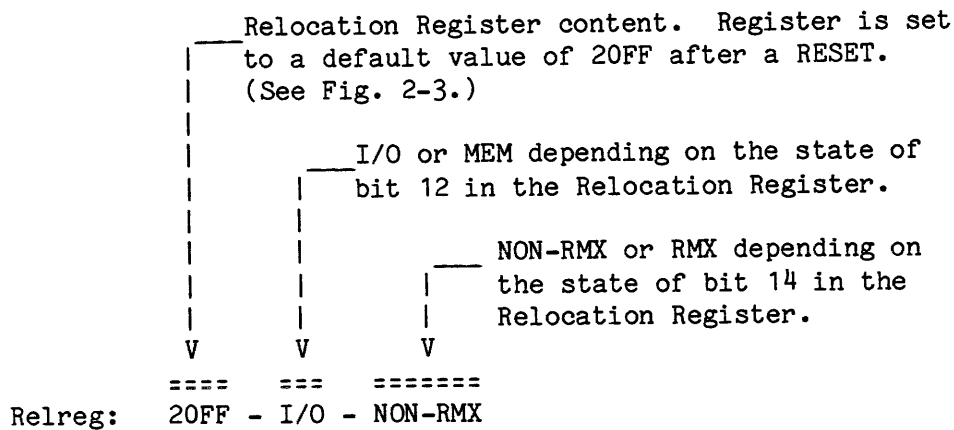
> al
00000 - 03FFF S
04000 - 07FFF

8 BLOCK(S) ALLOCATED 8 BLOCK(S) FREE
```


Table 2-2 earlier in this section explains the symbols displayed by the short form of the ds command.

The -p modifier is a new modifier to the ds command that is unique for the 80186/80188 Emulator. The ds -p command displays the contents of all Peripheral Control Block Registers that are readable. (Some registers are write only and cannot be read.) The register contents are displayed in two modes: RMX or NON-RMX. The mode of operation depends on the setting of bit 14 in the Relocation Register. This will be discussed later in greater detail.

The NON-RMX mode of operation is the default mode after a RESET or when bit 14 in the Relocation Register is set to zero. This is an example of a ds -p command display when in NON-RMX mode:



DMA

```

DMAOCW: XXXX   DMA1CW: XXXX
DMAOTC: XXXX   DMA1TC: XXXX
DMAODP: XXXXX  DMA1DP: XXXXX
DMAOSP: XXXXX  DMA1SP: XXXXX
  
```

Chip-Selects

```

MPCS: XXXX   MMCS: XXXX   PACS: XXXX
LMCS: XXXX   UMCS: XXXX
  
```

Timers

```

TOMCW: XXXX   T1MCW: XXXX   T2MCW: XXXX
TOMXB: XXXX   T1MXB: XXXX
TOMXA: XXXX   T1MXA: XXXX   T2MXA: XXXX
TOCNR: XXXX   T1CNR: XXXX   T2CNR: XXXX
  
```

Interrupts

```

ICRIO: XXXX   ICRI1: XXXX   ICRI2: XXXX   ICRI3: XXXX
ICRDO: XXXX   ICRD1: XXXX   ICRTC: XXXX   ICRIS: XXXX
ICRIR: XXXX   ICRIN: XXXX   ICRPM: XXXX   ICRMR: XXXX
ICRPS: XXXX   ICRPR: XXXX   ICRER: XXXX
  
```

| See Note

NOTE

The Interrupt Control Register ICRER is a write only register. Therefore, the displayed register contents is always "XXXX".

All Peripheral Control Block Registers are 16-bit registers, except for four 20-bit DMA registers (two words). Each 20-bit register is configured from two 16-bit registers (12 bits of each register pair are not used). These four DMA registers are:

- DMAODP
- DMA1DP
- DMAOSP
- DMA1SP

The first line in the display shows the Relocation Register's contents. In the preceding display, the "20FF" is the default value of the Relocation Register after a RESET. This causes the Peripheral Control Block to be located at an address of 0FF00 in I/O space. Figure 2-3 shows the functions of the bits in the Relocation Register.

NOTE

For an 80188 Emulator, if the user's program modifies the value of the Relocation Register during a **g** (go) command, the address of the Peripheral Control Block is lost until the next RESET. When this happens, the error message "Can't find the Peripheral Control Block" is displayed. If you know the modified value of the Relocation Register, you can use the **lpcb** command to let the emulating processor know where the Peripheral Control Block was moved.

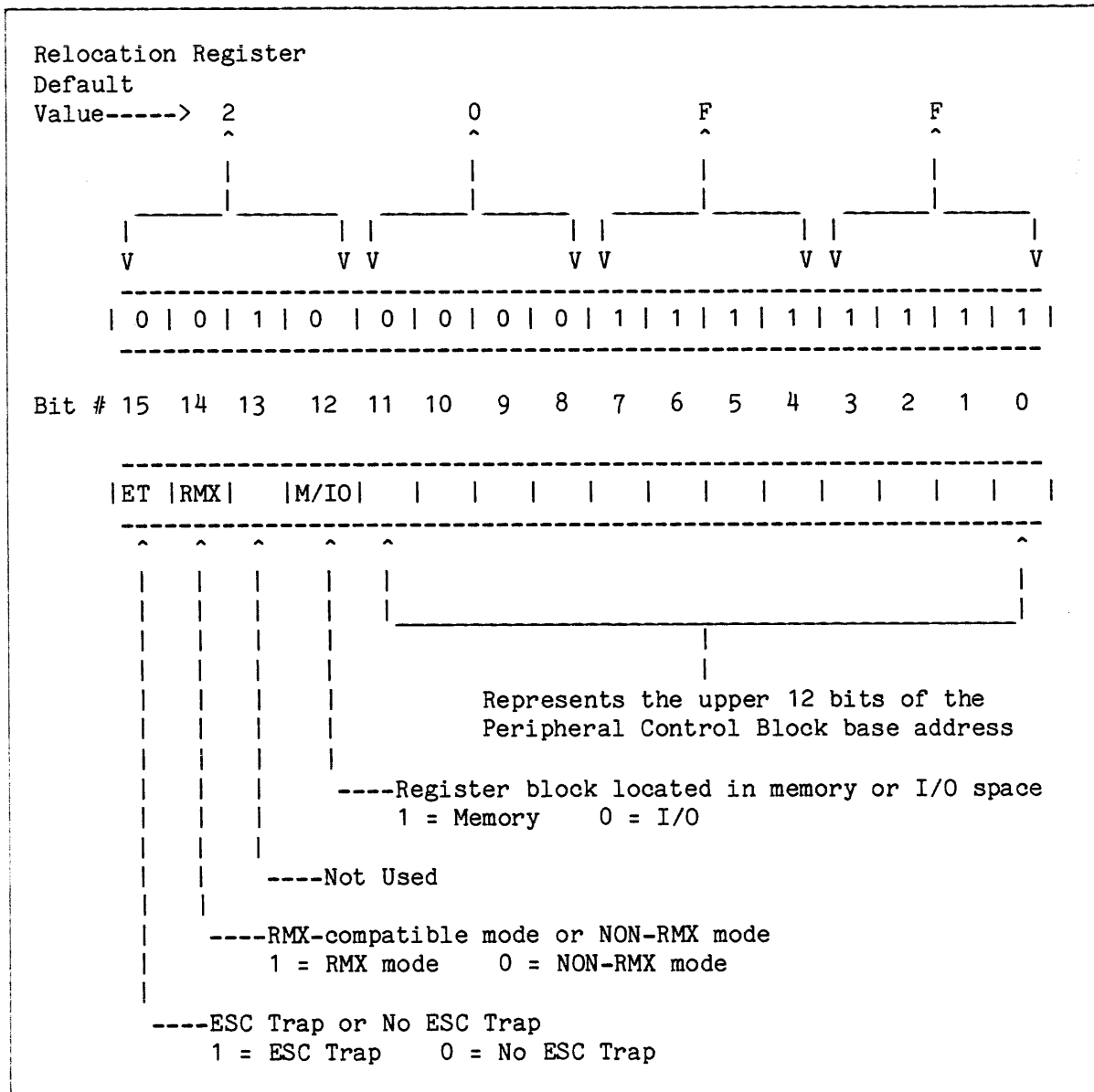


Fig. 2-3. Relocation Register content.

The RMX mode of operation (the 8086/8088 compatibility mode) is set when bit 14 of the Relocation Register is set to 1 (RMX mode). This is an example of a `ds -p` command display when in RMX mode:

Relreg: 50FF - MEM - RMX

DMA

DMAOCW: XXXX	DMA1CW: XXXX
DMAOTC: XXXX	DMA1TC: XXXX
DMAODP: XXXXX	DMA1DP: XXXXX
DMAOSP: XXXXX	DMA1SP: XXXXX

Chip-Selects

MPCS: XXXX	MMCS: XXXX	PACS: XXXX
LMCS: XXXX	UMCS: XXXX	

Timers

TOMCW: XXXX	T1MCW: XXXX	T2MCW: XXXX
TOMXB: XXXX	T1MXB: XXXX	
TOMXA: XXXX	T1MXA: XXXX	T2MXA: XXXX
TOCNR: XXXX	T1CNR: XXXX	T2CNR: XXXX

Interrupts

ICRLO: XXXX	ICRL2: XXXX	ICRL3: XXXX
ICRL4: XXXX	ICRL5: XXXX	ICRIS: XXXX
ICRIR: XXXX	ICRIN: XXXX	ICRPM: XXXX
ICRMR: XXXX	ICRSE: XXXX	ICRIV: XXXX

|__ See Note

NOTE

The Interrupt Control Register ICRSE is a write only register. Therefore, the displayed register contents is always "XXXX".

The register symbols used in the preceding RMX and NON-RMX displays are defined in Tables 2-3 and 2-4. The DMA, Chip-Select, and Timer registers have the same symbols for both RMX and NON-RMX modes. The register symbols for these registers are shown in Table 2-3. Some of the Interrupt Controller register symbols are not the same for RMX and NON-RMX modes. The register symbols for the Interrupt Controller registers are shown in Table 2-4.

Table 2-3
Peripheral Control Block--DMA, Chip-Select, and Timer Registers

Symbol	Description
RELREG	Relocation Register
	DMA Descriptors
DMAOCW	DMA Channel 0 Control Word Register
DMAODP	DMA Channel 0 Destination Pointer Register
DMAOSP	DMA Channel 0 Source Pointer Register
DMAOTC	DMA Channel 0 Transfer Count Register
DMA1CW	DMA Channel 1 Control Word Register
DMA1DP	DMA Channel 1 Destination Pointer Register
DMA1SP	DMA Channel 1 Source Pointer Register
DMA1TC	DMA Channel 1 Transfer Count Register
	Chip-Select Registers
LMCS	Lower Memory Chip Select Register
MMCS	Mid-Range Memory Chip Select Register
MPCS	Memory Programming Chip Select Register
PACS	Peripheral Address Chip Select Register
UMCS	Upper Memory Chip Select Register
	Timer Mode/Control Registers
TOCNR	Timer 0 Count Register
TOMCW	Timer 0 Mode/Control Word Register
TOMXA	Timer 0 Max Count A Register
TOMXB	Timer 0 Max Count B Register
T1CNR	Timer 1 Count Register
T1MCW	Timer 1 Mode/Control Word Register
T1MXA	Timer 1 Max Count A Register
T1MXB	Timer 1 Max Count B Register
T2CNR	Timer 2 Count Register
T2MCW	Timer 2 Mode/Control Word Register
T2MXA	Timer 2 Max Count A Register

Table 2-4
 Peripheral Control Block--Interrupt Control Registers

Symbol	Description

	Interrupt Control Registers - NON-RMX Mode
ICRDO	ICR DMA Channel 0 Register
ICRD1	ICR DMA Channel 1 Register
ICRER (b)	ICR EOI Register
ICRIN (a)	ICR In-Service Register
ICRIR (a)	ICR Interrupt-Request Register
ICRIS (a)	ICR Interrupt Status Register
ICRIO	ICR INTO Control Register
ICRI1	ICR INT1 Control Register
ICRI2	ICR INT2 Control Register
ICRI3	ICR INT3 Control Register
ICRMR (a)	ICR Mask Register
ICRPM (a)	ICR Priority Mask Register
ICRPR	ICR Poll Register
ICRPS	ICR Poll Status Register
ICRTC	ICR Timer Control Register

	Interrupt Control Registers - RMX Mode
ICRIN (a)	ICR In-Service Register
ICRIR (a)	ICR Interrupt-Request Register
ICRIS (a)	ICR Interrupt Status Register
ICRIV	ICR Interrupt Vector Register
ICRL0	ICR Level 0 Control Register (Timer 0)
ICRL2	ICR Level 2 Control Register (DMA 0)
ICRL3	ICR Level 3 Control Register (DMA 1)
ICRL4	ICR Level 4 Control Register (Timer 1)
ICRL5	ICR Level 5 Control Register (Timer 2)
ICRMR (a)	ICR Mask Register
ICRPM (a)	ICR Priority-Level Mask Register
ICRSE (b)	ICR Specific EOI Register

- a Register symbol used for both RMX and NON-RMX modes.
 b Write only registers.

G--BEGIN PROGRAM EXECUTION

The **g** (go) command begins program execution. With the 80186/80188 Emulator, the **g** command operates differently than with other emulators. Usually, if you enter a command like **g 1000**, the program counter is set to 1000, and execution begins from there. With the 80186/80188 Emulator, the **g** command uses information from the Code Segment (CS) Register and Instruction Pointer (IP) to calculate the effective starting address (EA). The EA is calculated as follows:

$$EA = IP + (CS * 16)$$

IP is the value of the **address** parameter you enter with the **g** command. Because the value of the CS Register may be changed during program execution, you should set the CS Register with the **s** command before running the program. Setting this register ensures that the program starts at the desired address.

NOTE

The value of the **g** command's **address** parameter must be less than 10000H.

LPCB--LOCATE PERIPHERAL CONTROL BLOCK

The **lpcb** (locate Peripheral Control Block) command is a new system command unique to the 80188 Emulator only. When using the 80188 Emulator if the user's program modifies the value of the Relocation Register during a **g** (go) command, the emulating processor loses the Peripheral Control Block's address. When this happens, the next time either the **ds -p** or **spcb** command is used, the error message "Can't find the Peripheral Control Block" is displayed. The **lpcb** command permits you to tell the emulating processor where the Peripheral Control Block was moved. The major difference between the **lpcb** command and the **spcb** command is that the **lpcb** command is used only by the emulating processor. No registers in the Peripheral Control Block are changed in value.

NOTE

If this command is used with the 80186 Emulator, the error message "Command only usable with 80188 Emulator" is displayed.

Example:

The following is an example of the **lpcb** command for either an 8540 or 8550:

```
> lpcb 60ff
```

```
====
```

```
^
```

```
|
```

```
----- This 16-bit expression is the value you believe is
           in the Relocation Register, including memory or
           I/O space selection, the interrupt mode, and the
           Peripheral Control Block base address. The content
           of the Relocation Register is defined in Fig. 2-3.
```

The following is an example of the `lpcb` command for an 8560 or other host in term mode:

```
> 8540 lpcb 60ff
```

MAP--SET OR DISPLAY MEMORY MAP ASSIGNMENTS

The 80186/80188 Emulator supports the `map` command described in your System Users Manual, with the following exceptions:

- The 80186/80188 Emulator maps memory in 4K-byte blocks.
- The `-m` and `-a` modifiers are not supported.
- You may not include memory segment information in the `map` command.

MEM--SPECIFY AVAILABLE USER MEMORY

The `mem` (memory) command informs the emulator that the prototype contains memory at a given block of addresses. This command operates on 4K-byte blocks. The `mem` command can also be used to reverse a previous `nomem` (no memory) command. The default condition is that all prototype memory is available to the program. Memory segment information is not valid with the `mem` command. For an example of `mem` command usage, see the discussion of the `nomem` command.

MEMSP--DEFINES MEMORY SPACE

The `memsp` (memory space) command is not implemented for the 80186/80188 Emulator.

NOMEM--SPECIFY UNAVAILABLE USER MEMORY

The `nomem` (no memory) command informs the emulator that the prototype does not contain memory at a given block of addresses. The default condition is that all prototype memory is available to the program. The `nomem` command operates on 4K-byte blocks. Memory segment information is not valid with the `nomem` command.

Example:

The following are examples of **mem** and **nomem** commands:

```
> nomem 0 7ffff

> nomem
INVALID USER MEMORY
  00000 - 7FFFF

> mem 0

> mem
VALID USER MEMORY
  00000 - 00FFF
  80000 - FFFFF

> nomem
INVALID USER MEMORY
  01000 - 7FFFF
```

RD--READ FROM EMULATOR I/O PORT

The **rd** (read) command reads data from a prototype I/O port. This command supports the **-b** (byte) and **-w** (word) command modifiers. The default is **-b**. You may include symbolic names as input to the **rd** command.

NOTE

Do not use the **rd** command to read from the area where the Peripheral Control Block is located. The **ds -p** command should be used to display the contents of the Peripheral Control Block registers.

The 80186/80188 Emulator supports both fixed-port and memory-mapped I/O. The default is fixed-port I/O. The I/O ports must be in the range 0000--FFFF. Memory-mapped I/O is selected with the **-m** command modifier. The 80186/80188 Emulator does **not** support the **-s** modifier of the **rd** command.

NOTE

The **rd** command **always** reads from the prototype. In mode 0, the command internally changes to mode 1 to execute the command and then returns to mode 0. An error message occurs if the probe is not connected to the prototype when this command is executed.

After the program has assembled and before the program is loaded into program memory, enter the `sel` command again with either the "80186" or "80188" parameter to select either the 80186 or 80188 Emulator.

SPCB--SET PERIPHERAL CONTROL BLOCK

The `spcb` (set Peripheral Control Block) command is a new system command unique to the 80186/80188 Emulator. The `spcb` command writes to the registers within the Peripheral Control Block. The `ds -p` (display status) command displays the contents of all the Peripheral Control Block Registers. Refer to Tables 2-3 and 2-4 for a list of the registers and register symbol names. Symbol names may be entered in either lowercase or uppercase. When a value is entered for a particular peripheral register, the value is not checked before sending it to the peripheral register. The relocation address for the Peripheral Control Block is contained in the lower 12 bits of the Peripheral Control Block's Relocation Register (refer to Fig. 2-3). These bits should not be set in the lowest 1000 of I/O address space or in the same I/O area that contains the SVC ports. Neither of these conditions are reported as an error.

To change the value of the Relocation Register for either an 8540 or 8550,, enter:

```
> spcb RELREG=30FF
```

To change the value of the Relocation Register for an 8560 or other host in term mode, enter:

```
> 8540 spcb relreg=30ff
```

You cannot write to two of the ICRs (Interrupt Control Registers) within the Peripheral Control Block. These registers are the ICRPR (ICR Poll Register) and the ICRPS (ICR Poll Status Register). If you attempt to write to these registers, an error message "Register not writable:" is displayed.

NOTE

You should use caution when writing to the registers in the Peripheral Control Block. The register symbols and values (entered with the `spcb` command) are not checked before writing to the designated register. A check is made to see if the designated register is available and that the Peripheral Control Block address can be read. The accuracy of the register values is not checked. Due to various types of registers (especially the timer registers), the values written to the registers with the `spcb` command may not equal the register values displayed with the `ds -p` command.

NOTE

For an 80188 Emulator, if the user's program modifies the value of the Relocation Register during a `g` (go) command, the address of the Peripheral Control Block is lost until the next RESET. When this happens, an error message "Can't find the Peripheral Control Block" is displayed the next time either the `ds -p` or `spcb` command is used. If you know the modified value of the Relocation Register, the `lpcb` command can be used to let the emulating processor know where the Peripheral Control Block was moved.

TRA--CONTROL DISPLAY OF EXECUTED INSTRUCTIONS

The `tra` (trace) command selects the range and type of instructions the system displays as your program executes. The short form of the `tra` command displays register values for the 80186/80188 microprocessors.

NOTE

Segment registers are not allowed in the trace address parameters because these registers may be changed during program execution. If you enter a segment register as part of a trace parameter, an error will occur when you enter the `g` (go) command.

WRT--WRITE TO EMULATOR I/O PORT

The **wrt** (write) command writes data to an I/O port. The **wrt** command supports the **-b** (byte) and **-w** (word) command modifiers. The default value is **-b**. You may include symbolic names as input to the **wrt** command.

NOTE

Do not use the **wrt** command to write to the area where the Peripheral Control Block is located. Only the **spcb** command should be used to change the contents of the Peripheral Control Block registers.

The 80186/80188 Emulator supports both fixed-port and memory-mapped I/O. The I/O ports must be in the range 0000--FFFF. You can select memory-mapped I/O with the **-m** command modifier. The 80186/80188 Emulator does not support the **-s** modifier of the **wrt** command. The default condition is fixed-port I/O.

NOTE

The **wrt** command **always** writes to the prototype. In mode 0, the command internally changes to mode 1 to execute the command and then returns to mode 0. An error message occurs if the probe is not connected to the prototype when you enter this command.

X--LOAD AND EXECUTE PROGRAM

The **x** (execute) command loads a file and executes a **g** (go) command at the transfer address. On the 80186/80188 Emulator, the **x** command is supported **only** under the following conditions:

- The CS Register is set to zero.
- The transfer address is within the range 0000--FFFF.

Since the transfer address is an absolute address containing both CS (Code Segment) and IP (Instruction Pointer) information, the **x** command can rarely be used with the 80186/80188 Emulator.

Refer to the "Special Considerations" paragraph later in this section for more information on program execution.

REAL-TIME PROTOTYPE ANALYZER (RTPA)

The 80186/80188 Emulator supports the Trigger Trace Analyzer (TTA) rather than the Real-Time Prototype Analyzer (RTPA).

TRIGGER TRACE ANALYZER (TTA) COMMANDS

The following text discusses the emulator-specific commands of the Trigger Trace Analyzer (TTA) option. For more information about the TTA, refer to the Trigger Trace Analyzer Users Manual, or to the TTA discussion in the emulation section of your System Users Manual.

BUS AND EVE--TTA BUS OPERATION DESIGNATORS

Table 2-5 lists the 80186/80188 bus operation designators recognized by the bus command and by the -b parameter of the eve command for the TTA.

Table 2-5
80186/80188 TTA Bus Operation Designators

Symbol	Bus Operation Type
BHE	Bus high enable (80186)
CLR	All types
F	First fetch
FS	Subsequent fetches
IA	Interrupt acknowledge
INT	Interrupt request
IR	I/O read
IW	I/O write
LCK	Bus priority lock control
RD	Memory reads
TST	Test control for WAIT instruction
WT	Memory writes

CONS--SET CONSECUTIVE EVENTS

The 80186/80188 Emulator supports the cons command's CYC mode of operation. The 80186/80188 Emulator does not support either the cons command's FET mode or EMU mode of operation.

DISP--DISPLAY CONTENTS OF TTA'S ACQUISITION MEMORY

The disp command displays the contents of the TTA's Acquisition Memory.

NOTE

When breakpoints are set, the display sometimes shows the displayed breakpoint is several instructions past the actual breakpoint parameters. This discrepancy occurs because the instructions are prefetched before they are executed. The difference in the displayed breakpoint and the actual breakpoint depends on the amount of instructions in the queue.

Here is an example of the `disp` command:

```
> disp
ADDR  DATA  MNEM  OPER          7-PROBE-0  BUS
000100 BB  MOVW  BX,#0500     1111 1111  F BHE TST LCK
000101 00                                1111 1111  FS BHE TST
000102 05                                1111 1111  FS BHE TST
000103 B9  MOVW  CX,#0005     1111 1111  F BHE TST
000104 05                                1111 1111  FS BHE TST
000105 00                                1111 1111  FS BHE TST
000106 32  XORB  AL,AL       1111 1111  F BHE TST
000107 C0                                1111 1111  FS BHE TST
000108 02  ADDB  AL,[BX]     1111 1111  F BHE TST
000109 07                                1111 1111  FS BHE TST
000500 00                                1111 1111  RD   TST
00010A 43  INC   BX          1111 1111  F BHE TST
00010B E2  LOOP  $-03       1111 1111  F BHE TST
00010C FB                                1111 1111  FS BHE TST
000108 02  ADDB  AL,[BX]     1111 1111  F BHE TST
000109 07                                1111 1111  FS BHE TST
000501 00                                1111 1111  RD BHE TST
00010A 43  INC   BX          1111 1111  F BHE TST
00010B E2  LOOP  $-03       1111 1111  F BHE TST
00010C FB                                1111 1111  FS BHE TST
000108 02  ADDB  AL,[BX]     1111 1111  F BHE TST
000109 07                                1111 1111  FS BHE TST
```

NOTE

A change exists in the format of the displayed TTA's Acquisition Memory with an 80186/80188 Emulator. The "Q" column that showed the number of bytes remaining in the prefetch queue and the Segment Registers column (CS, DS, SS, and ES) are deleted from the display.

The following display contains an example of the `disp` command's output with both `tra all` and symbolic debug on. The information at addresses 8, 9, and A appears whenever the operating system generates a break. (The system

breaks at every instruction during tra all.) The LCK bus transaction occasionally appears when the system resumes execution and does not signify an error.

```

> disp
ADDR  DATA MNEM  OPER          7-PROBE-0  BUS

START
000100 BB  MOVW  BX,#0500    1111 1111    F BHE TST LCK
000101 00                                1111 1111    FS BHE TST
000102 05                                1111 1111    FS BHE TST
000008 7F                                1111 1111    RD BHE TST
000009 EF                                1111 1111    RD BHE TST
00000A 7F                                1111 1111    RD BHE TST

DEMO+000103
000103 B9  MOVW  CX,#0005    1111 1111    F BHE TST
000104 05                                1111 1111    FS BHE TST
000105 00                                1111 1111    FS BHE TST
000008 7F                                1111 1111    RD BHE TST
000009 77                                1111 1111    RD BHE TST
00000A 7F                                1111 1111    RD BHE TST

DEMO+000106
000106 32  XORB  AL,AL      1111 1111    F BHE TST LCK
000107 C0                                1111 1111    FS BHE TST
000008 7F                                1111 1111    RD BHE TST
000009 E7                                1111 1111    RD BHE TST
  
```

SPECIAL CONSIDERATIONS

The following paragraphs describe those special considerations that are unique to the 80186/80188 Emulator.

THE HALT INSTRUCTION

In all three modes, a HALT instruction causes the emulator to halt. Control does **not** automatically return to the operating system. Press CTRL-C to return control to the operating system.

LOSING CLOCK CAUSES EMULATOR FAULT ERROR

When you select the 80186/80188 Emulator, a section of system code is loaded in the emulator. This code is checked each time you start the emulator. If this code is corrupted in some way, the development system's error "45 - Emulator faulted" is displayed, and you must reselect the emulator.

Losing the clock in emulation modes 1 or 2 causes this error to occur. You will lose the clock if you turn off your prototype while the development system is in emulation mode 1 or 2. You can avoid this error by changing to emulation mode 0 before turning off your prototype.

INTERRUPT STATUS REGISTER CONSTRAINTS

During code execution, DMA activity can be prevented by setting bit 15 in the Interrupt Status Register of the Peripheral Control Block. Setting bit 15 (DHLT--DMA halt transfer bit) high disables all DMA activity. Bit 15 is automatically set high whenever an NMI occurs. Bit 15 is cleared by an interrupt return (IRET) instruction or a system RESET.

When breaking or escaping from user code execution, the 80186/80188 Emulator uses an NMI. The NMI sets bit 15 in the Interrupt Status Register high. When the emulator returns to user code execution (**go** command), the emulator uses the IRET instruction which clears bit 15. The interrupt (NMI) and interrupt return (IRET) are transparent to the user except for the setting and clearing of bit 15.

The status of bit 15 is displayed with the **ds -p** command. The value of register symbol ICRIS is 0XXX when bit 15 is cleared, and 8XXX when bit 15 is set high.

PROM PROGRAMMER CONSIDERATIONS

Because of power supply limitations, the 80186/80188 Emulator must **not** be selected while you are programming a PROM.

EMULATING THE EXECUTION BUS

Internally, the 80186 and 80188 are each dual processors. Each contains an Execution Unit (EU) and a Bus Interface Unit (BIU). The Bus Interface Unit prefetches instructions for the Execution Unit, and interfaces the EU to the outside world. The BIU communicates with the EU via an "Execution Bus".

The 80186/80188 Emulator emulates the information on the Execution Bus: this means that instructions are disassembled and traced accurately. Breakpoints are generated by information on the Execution Bus.

PROTOTYPE INTERRUPTS DURING TRACE ALL MODE

If a prototype interrupt occurs while operating in trace all mode, the instruction immediately preceding the interrupt may be displayed twice. The first display is the execution of the instruction and the second display is caused by the prototype interrupt. The mnemonics of the second display are the same as the first display. However, the register contents of the second display reflect those register changes resulting from the interrupt. Even though this instruction is displayed twice, it is only executed once. If an optional TTA is available, the TTA `disp` command shows the correct sequence of events.

DMA ACTIVITY DURING TRACE ALL MODE

Whenever you are executing code in the trace all mode, the last instruction executed prior to a DMA interrupt is repeatedly displayed until the DMA activity is completed. The last instruction is displayed at the rate of one trace for each DMA transfer. The repeated display of the last instruction can be prevented by turning off the trace during the address range of the DMA activity.

TTA DISPLAY CHANGES DURING DMA ACTIVITY

Since the 80186/80188 device is a pipe-lined processor, during DMA activity the 80186/80188 processor continues to execute the instructions that are in the processor's queue, providing the execution of the instruction does not intervene with the DMA activity. An example of a non-intervening instruction is one that requires a move from one register to another. However, if the first instruction fetched requires a move from a register to memory, the execution of the instruction is suspended until the DMA activity is completed.

Therefore, during DMA activity, the TTA display may be affected as follows:

1. If the executing instruction causes no intervention of DMA activities, the first and second instruction fetches may be displayed separately in the TTA's display, with the read/write DMA activities shown between the two instruction fetches. This displayed separation of the instruction fetches is necessary due to the amount of time required to store the DMA activities.

NOTE

The displayed separation depends on several variables: queue depth, type of instruction, and the length of the DMA transfer.

2. If the executing instruction causes an intervention of DMA activities, the first instruction fetch is displayed. However, the second instruction fetch is not displayed until it is executed at the completion of the DMA activity. This prevents the TTA from correctly disassembling the instructions due to the time difference between the two fetches.

SEGMENT REGISTERS

The 80186/80188 microprocessor contains four segment registers: Code Segment (CS), Data Segment (DS), Stack Segment (SS), and Extra Segment (ES). Each segment register contains the 16 most significant bits of an 80186/80188 20-bit address. Addresses are given as a 16-bit offset to the segment base address.

NOTE

When executing a program with trace all in effect, all instructions that change the segment registers are not displayed. The **trace all** command skips over these instructions.

CSX, DSX, SSX AND ESX SYMBOLS

For convenience, a special symbol has been created for each of the four memory segment registers that is 16 times the value in that segment's associated segment register. For example, the symbol DSX represents a value 16 times the value of the DS Register.

The following example shows a use for the CSX symbol:

Example

The 80186/80188 microprocessor creates the effective address (EA) for a symbolic label by adding the current Code Segment (CS) times 16 to the contents of the Instruction Pointer (IP).

The 80186/80188 Emulator only knows the effective address (EA) of a symbol. However, the **address** parameter of the **g** command presumably contains only IP information. Under certain circumstances, (such as when the EA of a symbol is greater than FFFF, or the CS is not zero) the 80186/80188 Emulator calculates an IP larger than 16 bits, and a truncation error occurs.

However, you can use the EA and the CS to calculate the IP.

$$IP = EA - (16*CS)$$

Because the value of CSX is 16*CS, you can avoid truncation errors by subtracting the CSX symbol in the expression representing the desired address.

> g START-CSX

The -CSX modifier subtracts the CS value from the EA, thus ensuring that the IP is no more than 16 bits.

SERVICE CALLS

Service calls (SVCs) enable your program to use many 8450, 8550, or 8560 system capabilities.

An SVC is invoked with an 80186/80188 OUT instruction followed by two NOPs. The OUT instruction's operand is indirectly referenced through Register DX whenever the SVC port range is larger than 00FF. (For example, the default SVC port range of 01000--01007 requires that the DX Register be used.) This instruction sequence directs the system to a specified memory address called the SRB (service request block) pointer, which points to the address of the service request block (SRB). The SRB pointer tells the system where to find the data stored in the SRB. The SRB tells the development system what service to perform. Refer to the service calls section of your System Users Manual for an explanation of service calls, service request blocks, and SRB pointers.

Table 2-6 shows the default addresses for the eight SRB pointers. These addresses and their associated port values can be altered with the SVC command to suit your program requirements. See the Command Dictionary of your System Users Manual for syntax and use of the **svc** command.

SVCS IN MODES 1 AND 2

The 80186/80188 Emulator supports SVCs for all three emulation modes. In all three modes, use **two** NOPs following the I/O instruction. This allows time for the SVC to occur.

NOTE

In mode 1, only the instruction sequence that calls the SVC may reside in prototype memory. The SRB pointers, the SRB, and the optional I/O buffer must all reside in program memory.

SRB FORMAT

The 80186/80188 Emulator uses the LAS (Large Address Space) format for SRBs and SRB pointers. This format is described in the service calls section of your System Users Manual.

SVC DEMONSTRATION

Figure 2-4 lists an 80186/80188 program that uses four SVC functions: Assign Channel, Read ASCII, Write ASCII, and Abort. The program's algorithm is explained in the service calls section of your System Users Manual, which demonstrates a version of the program written in 8085A assembly language. Using the program in Fig. 2-4, you can perform a parallel demonstration with the 80186/80188 B Series Assembler and 80186/80188 Emulator.

The program accepts a line of ASCII characters from the system terminal. Then, when it receives a RETURN character, the program writes the line to the line printer and accepts another line. On the 8550, output to the line printer is buffered. No text is printed until the 8550's line printer buffer becomes full or the program ends.

Table 2-6
 80186/80188 Service Calls

SVC Number	Service Call (a)		Default
	Mnemonic	Hexadecimal	Location of SRB Pointer
1	MOVW DX, #01007H	BA0710	40--43
	OUT DX, AL	EE	
	NOP	90	
	NOP	90	
2	MOVW DX, #01006H	BA0610	44--47
	OUT DX, AL	EE	
	NOP	90	
	NOP	90	
3	MOVW DX, #01005H	BA0510	48--4B
	OUT DX, AL	EE	
	NOP	90	
	NOP	90	
4	MOVW DX, #01004H	BA0410	4C--4F
	OUT DX, AL	EE	
	NOP	90	
	NOP	90	
5	MOVW DX, #01003H	BA0310	50--53
	OUT DX, AL	EE	
	NOP	90	
	NOP	90	
6	MOVW DX, #01002H	BA0210	54--57
	OUT DX, AL	EE	
	NOP	90	
	NOP	90	
7	MOVW DX, #01001H	BA0110	58--5B
	OUT DX, AL	EE	
	NOP	90	
	NOP	90	
8	MOVW DX, #01000H	BA0010	5C--5F
	OUT DX, AL	EE	
	NOP	90	
	NOP	90	

a = The default SVC port range, 01000--01007, is assumed.

NOTE

The program shown in Fig. 2-4 is written for a B Series assembler. To make this program acceptable for an A Series assembler used on some 8550s, change each single quote (') to a double quote ("). To terminate the program, enter a CTRL-Z while the program is waiting for input.


```

; SSSSS V V CCCCC
; S V V C
; SSSSS V V C DEMONSTRATION. 80186/80188 EMULATOR
; S V V C
; SSSSS V CCCCC
    ORG 40H ; Beginning of SRB vector
    BYTE 0,BITS(SRB1FN,16,4),HI(SRB1FN),LO(SRB1FN)
    BYTE 0,BITS(SRB2FN,16,4),HI(SRB2FN),LO(SRB2FN)
    BYTE 0,BITS(SRB3FN,16,4),HI(SRB3FN),LO(SRB3FN)
    BYTE 0,BITS(SRB4FN,16,4),HI(SRB4FN),LO(SRB4FN)
    BYTE 0,BITS(SRB5FN,16,4),HI(SRB5FN),LO(SRB5FN)
; End of SRB vector
    ORG 60H ; Set up SRB areas
; SRB1 = Assign 'CONI' to Channel 0
SRB1FN BYTE 10H ; Assign
        BYTE 00H ; to Channel 0
SRB1ST BLOCK 01H ; Status returned here
        BLOCK 03H ; Bytes 4 through 6 not used
        BYTE 00H,05H ; Length of 'CONI' +
        BYTE 00H ; Pointer
        BYTE BITS(CONI,16,4) ; to
        BYTE HI(CONI) ; 'CONI'
        BYTE LO(CONI) ; +
; End of SRB1
; SRB2 = Assign 'LPT' to Channel 1
SRB2FN BYTE 10H ; Assign
        BYTE 01H ; to Channel 1
SRB2ST BLOCK 01H ; Status returned here
        BLOCK 03H ; Bytes 4 through 6 not used
        BYTE 00H,04H ; Length of 'LPT' +
        BYTE 00H ; Pointer
        BYTE BITS(LPT,16,4) ; to
        BYTE HI(LPT) ; 'LPT'
        BYTE LO(LPT) ; +
; End of SRB2
; SRB3 = Read ASCII line from CONI (Channel 0)
SRB3FN BYTE 01H ; Read ASCII
        BYTE 00H ; from Channel 0
SRB3ST BLOCK 01H ; Status returned here
        BLOCK 01H ; Byte 4 not used
        BLOCK 02H ; Byte count returned here
        BYTE 01H,00H ; 256 Bytes in our buffer
        BYTE 00H ; Pointer
        BYTE BITS(BUFFER,16,4); to
        BYTE HI(BUFFER) ; BUFFER
        BYTE LO(BUFFER) ; +
; End of SRB3

```

Fig. 2-4. 80186/80188 SVC demonstration program listing (Part 1 of 3).

```

;      SRB4 = Write ASCII line to LPT (Channel 1)
SRB4FN BYTE    02H          ; Write ASCII
        BYTE    01H          ;   to Channel 1
SRB4ST BLOCK   01H          ; Status returned here
        BLOCK   01H          ; Byte 4 not used
        BLOCK   02H          ; Byte count returned here
        BYTE    01H,00H     ; 256 bytes in our buffer
        BYTE    00H          ; Pointer
        BYTE    BITS(BUFFER,16,4); to
        BYTE    HI(BUFFER)   ;   BUFFER
        BYTE    LO(BUFFER)   ;   +
;      End of SRB4
;      SRB5 = Abort (Close all channels and terminate)
SRB5FN BYTE    1FH          ; Abort
        BLOCK   0BH          ; Bytes 2 through 12 not used
;      End of SRB5
;
;      BUFFER BLOCK   100H     ; Our I/O area
CONI    ASCII   'CONI'       ; ASCII of 'CONI'
        BYTE    0DH          ;   +
LPT     ASCII   'LPT'        ; ASCII of 'LPT'
        BYTE    0DH          ;   +
;      End of data definitions
;
;      Beginning of executable code
START  ORG     1000H         ; Entry point into program
        MOVW   DX,#01007H   ; Call SVC1
        OUT   DX,AL         ; to assign 'CONI'
        NOP                   ; to Channel 0
        NOP                   ;
        MOV    AL,SRB1ST    ; Check the status
        CMP   AL,#00H       ; to see if all went well
        JNZ  ABORT          ; No? Stop everything
        MOVW  DX,#01006H   ; Call SVC2
        OUT   DX,AL         ; to assign 'LPT'
        NOP                   ; to Channel 1
        NOP                   ;
        MOV   AL,SRB2ST    ; Check the status
        CMP  AL,#00H       ; to see if all went well
        JNZ  ABORT          ; No? Stop everything

```

Fig. 2-4. 80186/80188 SVC demonstration program listing (Part 2 of 3).

```
ALOOP  MOVW  DX,#01005H      ; Call SVC3
        OUT   DX,AL        ; to read a line
        NOP                    ; from 'CONI'
        NOP                    ; into the buffer
        MOV   AL,SRB3ST     ; Check the status
        CMP   AL,#00H      ; to see if all went well
        JNZ  ABORT         ; No? Stop everything
        MOVW  DX,#01004H   ; Call SVC4
        OUT   DX,AL        ; to write the line
        NOP                    ; from the buffer
        NOP                    ; to 'LPT'
        MOV   AL,SRB4ST     ; Check the status
        CMP   AL,00H       ; to see if all went well
        JZ   ALOOP         ; Yes? Back to read another line
                          ; No? Fall through to termination
ABORT  MOVW  DX,#01003H   ; Call SVC5
        OUT   DX,AL        ; to exit
        NOP                    ; the program
        NOP                    ;
        END   START       ; End of program
```

Fig. 2-4. 80186/80188 SVC demonstration program listing (Part 3 of 3).

ERROR MESSAGES

There are two error messages associated with the **spcb** command:

Register not compatible with RMX mode: This error message is displayed when an Interrupt Register name assignment is made and the register name is not valid in the current interrupt mode.

Register not writable: This error message is displayed whenever you attempt to write to either the ICRPR (Interrupt Control Registers - Poll Register) or the ICRPS (Interrupt Control Registers - Poll Status Register). Refer to Table 2-4.

There are two error messages associated with the **lpcb** command:

Can't find the Peripheral Control Block: This error message is displayed for an 80188 Emulator only. If the user's program modifies the value of the Relocation Register during a **g** (go) command, the address of the Peripheral Control Block is lost until the next RESET. When this happens, the next time either the **ds -p** or **spcb** command is used this error message is displayed. If you know the modified value of the Relocation Register, the **lpcb** command can be used to let the emulating processor know where the Peripheral Control Block was moved.

Command only usable with 80188 Emulator: This error message is displayed when the **lpcb** command is used with an 80186 Emulator.

Section 3

EMULATOR DEMONSTRATION RUN

INTRODUCTION

This section contains a demonstration run that shows you how to load, execute, and monitor a simple 80186 program on your 8540 or 8550. This demonstration run is designed to be used with the Learning Guide in Section 1 of your System Users Manual. Before you perform this demonstration, your 80186/80188 Emulator hardware and control software must be installed in your development system. Refer to Section 6 of this manual for hardware and software installation procedures.

NOTE

There is no 80186/80188 Assembler for the 8550 development system. If you have an 8086/8088 Assembler for the 8550, you can use the demonstration program and control software provided with your 8086/8088 Emulator to assemble and load the demonstration program for this demonstration run. The demonstration program does not contain the unique instructions for the 80186/80188 microprocessor.

DEMONSTRATION PROGRAM

Figure 3-1 shows the source and object code for the demonstration program. Refer to this figure as you examine the demonstration program.

EXAMINE THE DEMONSTRATION PROGRAM

The demonstration program adds five numbers from a table stored in locations 500--504 in program memory and leaves the sum in Register AL. You will place values in the table later in this demonstration. The 8085A Emulator demonstration run in your System Users Manual contains a flowchart that illustrates the steps of the program.

The source code contains two kinds of statements: assembler directives (like `ORG` and `BYTE`) and 80186 assembly language instructions. The assembler directives are microprocessor independent and are explained in the 8085A Emulator demonstration run. The 80186 assembly language instructions are discussed in the following paragraphs.

Set Pass Counter

Register CX is used as the pass counter. The MOVW CX,#TSIZE instruction moves the value of TSIZE into CX. This step sets the number of passes to 5.

Clear Accumulator

The XORB AL,AL instruction zeros Register AL so that you can use AL as the accumulator and can start adding numbers from the table into AL.

Add Byte from Table

The ADDB AL,[BX] instruction adds the byte addressed by BX into the accumulator AL. The label ALOOP represents this instruction's address. The LOOP instruction uses this label.

Point to Next Byte

The INCW BX instruction increments Register BX, which then points to the next byte in the table. For example, BX is initialized to contain 500H. After the INCW BX instruction is first executed, BX contains 501, the address of the second element of the table.

Decrement CX and Loop Until CX = Zero

The LOOP ALOOP instruction decrements Register CX, the pass counter. If CX is not yet zero, the program jumps back to the label ALOOP. If CX is equal to zero, the program calls the Exit SVC routine.

Call Exit SVC

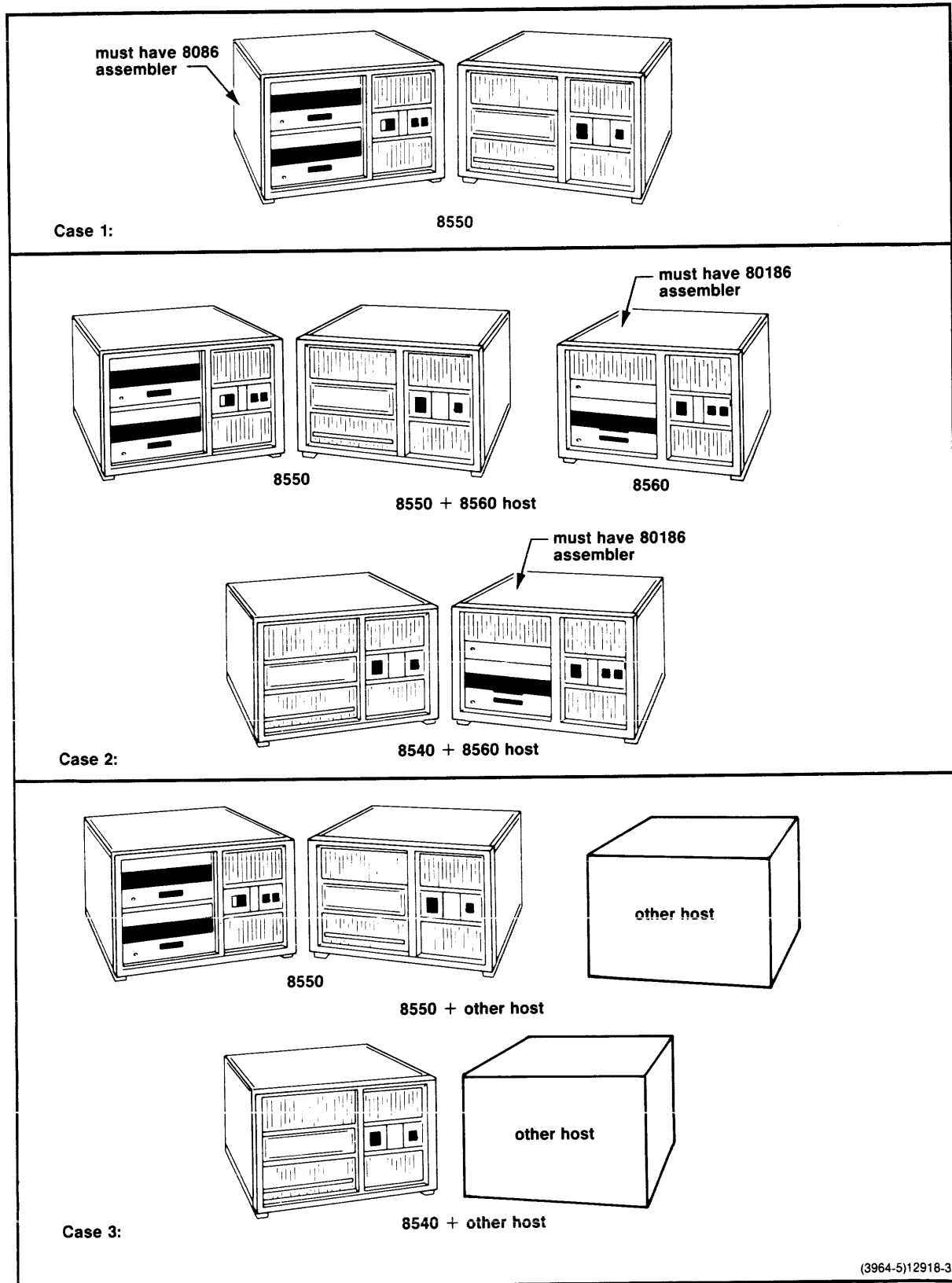
The last four instructions of the program are:

```
MOVW DX,#01007H
OUT DX,AL
NOP
NOP
```

These instructions constitute a service call (SVC) that causes an exit from the program. For more information on SVCs, refer to the Service Calls discussion in Section 2 of this manual.

DEVELOPMENT SYSTEM CONFIGURATIONS

Figure 3-2 shows the various hardware configurations that apply to this demonstration run. These configurations are briefly described here with detailed descriptions following later in this section. Find the case that is appropriate for your hardware configuration.



(3964-5)12918-3

Fig. 3-2. Development system configurations.

1. **Case 1 -- For 8550 users:** This demonstration shows you how to assemble the 80186/80188 program on your 8550. If your system disk does not contain an 8086/8088 Assembler, you cannot complete this part of the demonstration. Skip ahead to "Case 4: Patch the Program Memory" for instructions.

NOTE

There is no 80186/80188 Assembler for the 8550 development system. If you have an 8086/8088 Assembler for your 8550, the demonstration program and control software provided with your 8086/8088 Emulator can be used to assemble and load the demonstration program for this demonstration run. The demonstration program does not contain the unique instructions for the 80186/80188 microprocessor.

2. **Case 2 -- For 8540/8560 or 8550/8560 users:** If you have an 8540/8560 or 8550/8560 system and your 8560 has an 80186/80188 Assembler installed, you can create and assemble the program on the 8560 and then download it to the 8540 or 8550. This demonstration shows how.
3. **Case 3 -- For 8540 or 8550 users with host computers other than the 8560:** If you have an 8540 or 8550 that is connected to a host computer other than an 8560, this manual doesn't give you a specific list of commands for creating and assembling the program on your host (since we don't know what host you're using). However, you can create the Tekhex file using your host's assembler or text editor, then download the file to the 8540 or 8550 through the system's COM interface.
4. **Case 4 -- For other hardware configurations:** If none of these cases apply to you, you can patch the program into memory using the p command. This demonstration shows how.

ASSEMBLE AND LOAD THE DEMONSTRATION PROGRAM

Now it's time to create the program so you can run it on your emulator. Detailed discussions for each of the preceding cases are contained in the following paragraphs. Go ahead and work through the discussion that's appropriate for you. Once the program is loaded or patched into memory, you can then execute the program on your emulator. Turn to the heading "Run the Demonstration Program," later in this section for instructions.

CASE 1: ASSEMBLE AND LOAD ON THE 8550NOTE

There is no 80186/80188 Assembler for the 8550 development system. If you have an 8086/8088 Assembler for the 8550, the demonstration program and control software provided with your 8086/8088 Emulator can be used to assemble and load the demonstration program for this demonstration run. The demonstration program does not contain the unique instructions for the 80186/80188 microprocessor.

This discussion shows you how to copy the demonstration program from your 80186/80188 Emulator software installation disk, assemble the program, and load it into 8550 program memory. If your system disk does not contain an 8086/8088 Assembler, you cannot complete this part of the demonstration. Skip ahead to "Case 4: Patch the Program Memory" for instructions.

Start Up and Log In

Turn on your 8550 system. For start-up instructions, refer to the paragraph "Start Up the 8550 and Its Peripherals" in the Learning Guide of your System Users Manual. Place your system disk in drive 0 and shut the drive 0 door. When your system displays the > prompt, place your 80186/80188 Emulator software installation disk in drive 1 and shut the drive 1 door.

Use the **dat** command to set the current date and time. For example, if it is 2:30 p.m. on March 30, 1984, enter the following command line:

```
> dat 30-mar-84/2:30 pm
```

Use the **sel** command to tell DOS/50 to use the assembler and emulator software designed for the 8086/8088 microprocessors.

```
> sel 8086
```

The system responds with the current version number:

```
8086 Emulator V n.nn mm/dd/yy
```

The **sel** command automatically sets the emulation mode to 0.

Copy the Demonstration Run Program from the Installation Disk

Enter the following command lines to create an empty directory called DEMO on your system disk and make DEMO the current directory. The **br** command creates a brief name, ROOT, to mark the old current directory. At the end of this demonstration, you will return to this ROOT directory and delete the DEMO directory and its contents.

```
> br root /usr  
  
> create DEMO  
  
> user DEMO
```

Now use the **cop** command to copy all the files in the DEMO2 directory on the installation disk to the DEMO directory you just created:

```
> cop /vol/emu.8086/DEMO2/* *
```

Remove your installation disk from drive 1 and put the disk away.

Now list the files you have just copied to the current directory:

```
> l  
FILENAME  
  
ASM  
LOAD  
  
Files used      124  
Free files      132  
Free blocks     821  
Bad blocks      0
```

The file named ASM contains the assembly language source code for this demonstration program, and the file named LOAD contains the executable object code. This copy of LOAD will be used in the demonstration only if you do not have an 8086/8088 Assembler and cannot create your own object file and load file from the source file.

Examine the Demonstration Program

Enter the following command line to display the source file ASM on the system terminal. Your display may differ slightly from this example because of your terminal's tab settings.

```

> con ASM
;80186 DEMONSTRATION RUN PROGRAM
SECTION DEMO
ORG 100H ; START PROGRAM CODE AT ADDRESS 100 HEX
START MOVW BX,#TABLE ; SET TABLE POINTER
MOVW CX,#TSIZE ; SET PASS COUNTER
XORB AL,AL ; CLEAR ACCUMULATOR
ALOOP ADDB AL,[BX] ; ADD BYTE FROM TABLE
INCW BX ; POINT TO NEXT BYTE
LOOP ALOOP ; DECREASES CX AND LOOPS UNTIL CX=0
MOVW DX,#01007H ; I/O ADDRESS FOR EXIT SVC
OUT DX,AL ; CALL EXIT SVC
NOP ; TO END PROGRAM
NOP ; EXECUTION
;SRB POINTER
ORG 40H ; STORE SRB POINTER AT ADDRESS 40 HEX
BYTE 0H,0H,0H,44H ;POINT TO SRB FOR EXIT SVC
;SRB FOR EXIT SVC
BYTE 1AH ; 1AH = FUNCTION CODE FOR EXIT SVC
;TABLE OF NUMBERS TO BE ADDED
TSIZE EQU 5 ; TABLE SIZE = 5
ORG 500H ; SET UP TABLE AT ADDRESS 500 HEX
TABLE BLOCK TSIZE
LIST DBG
END START

```

Assemble the Source Code

If you do not have an 8086/8088 Assembler on your system disk, you cannot perform this step. Skip the next four commands: **asm**, **cop**, **link**, and **l**.

The **asm** (assemble) command translates assembly language (source code) into binary machine language (object code). The **asm** command also creates an assembler listing that correlates the object code with the source code. Enter the following command line to assemble the source code in the file ASM and create the listing and object files ASML and OBJ:

```
> asm OBJ ASML ASM
    ^    ^    ^
    |    |    |
    |    |    |  -- source file
    |    |    |
    |    |    |  ----- assembler listing file
    |    |    |
    |    |    |  ----- object file
```

```
Tektronix    8086 Vxx.xx-xx Copyright (c) 1981 Tektronix
**** Pass 2
  24 Source Lines  24 Assembled Lines  xxxxx Bytes Available
    >>> No assembly errors detected <<<
```

Make sure the printer is properly connected and is turned on. Then enter the following command to copy the assembler listing to the line printer:

```
> cop ASML lpt
```

The different fields of your source listing are shown in Fig. 3-1, earlier in this demonstration. For a detailed explanation of assembler listings, consult your Assembler Core Users Manual.

Link the Object Code

The linker creates an executable load file from one or more object files. Enter the following linker command to create a load file called LOAD from your object file, OBJ:

```
> link -O OBJ -o LOAD -d
```

The linker command options **-O** and **-o** specify the object file and load file, respectively. The **-d** command option causes the linker to pass the program symbols from the object file to the load file for use in program debugging.

The files generated by the **asm** and **link** commands should now be on your disk. Enter the following command to list the files in your current directory:

```
> l
FILENAME

ASM
LOAD
OBJ
ASML

Files used      126
Free files      130
Free blocks     811
Bad blocks      0
```

Notice that there are now four files listed in your directory. OBJ and ASML were created by the assembler, and LOAD was created by the linker.

Select the 80186/80188 Emulator

Before loading the demonstration program into program memory, use the **sel** command to turn off the 8086 Emulator/Assembler and select the 80186 Emulator. Enter the **sel** command as follows:

```
> sel 80186
```

Load the Program into Memory

Now it's time to load the object code from the load file LOAD into program memory.

Allocate Memory. Enter the following command to allocate a 4K-byte block of program memory to logical addresses 00000--00FFFH.

```
> al 0
```

Zero Out Memory. Before you load any code, use the **f** (fill) command to fill program memory with zeros. Later, when you examine memory, the zeros make it easy to identify the beginning and end of your code. Zeroing out memory has no effect on how the program is loaded. Enter the following command line to fill memory addresses 40--11F with zeros:

```
> f 40 11f 00
```

Check that Memory is Filled with Zeros. Check the contents of memory with the d (display) command. The display shows the data in hexadecimal format, and also shows the corresponding ASCII characters. Display the contents of memory addresses 40--11F with the following command line:

```
> d 40 11f
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
000040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
```

Load the Object Code into Memory. Enter the following command line to load the object code for the demonstration program into program memory:

```
> lo <LOAD
      ====
      ^
      |
      load file
```

Load the Program Symbols. The source code for the demonstration program contains the directive LIST DBG. (Refer to Fig. 3-1.) Because of this directive, the object file contains a list of the symbols that appeared in the source code, and the values associated with those symbols. Because you included the -d command option when you invoked the linker, those symbols were passed to the load file. Use the **symlo** command to load those symbols into the symbol table in 8550 system memory.

```
> symlo -s <LOAD
```

The -s option means that both addresses and scalars are loaded. If you omit the -s option, only addresses are loaded. A scalar is a number that is not an address. In Fig. 3-1, the scalar TSIZE represents the length of a table.

Later in this demonstration, whenever you use a symbol in a command line, DOS/50 refers to the symbol table you just created to find the value that the symbol represents.

You've assembled and linked the demonstration program and loaded it into memory. Now skip ahead to "Run the Demonstration Program."

CASE 2: ASSEMBLE ON THE 8560; DOWNLOAD TO THE 8540 OR 8550

This discussion shows you how to create the demonstration program source code and assemble it on an 8560, then download it into 8540 or 8550 program memory. If your 8560 does not have an 80186/80188 Assembler, you cannot complete this part of the demonstration. Skip ahead to "Case 4: Patch the Program into Memory" for instructions.

NOTE

This demonstration shows the 8540 commands that can also be used for an 8550 connected to an 8560. As a result, you can substitute 8550 for 8540 throughout the demonstration unless otherwise noted.

Start Up and Log In

Start up your 8540, make sure it's in TERM mode, and log in to the 8560 TNIX operating system. See your 8560 System Users Manual for details.

While you are logged in to TNIX, your system prompt is "\$". Later in the demonstration the system prompt ">" appears for people using 8540s and 8550s in LOCAL mode. Every command you enter is processed by TNIX. If you enter an OS/40 command, TNIX passes it to the 8540.

Enter the following commands to select the 80186/80188 Assembler on the 8560 and the 80186/80188 Emulator in the 8540:

```
$ uP=80186; export uP
$ sel 80186
```

NOTE

If you are using an 8550, enter either **sel 80186** or **sel 80188** depending on which Prototype Control Probe you have installed.

The **sel** command automatically sets the emulation mode to 0.

Create the Demonstration Program

Enter the following TNIX command lines to create an empty working directory called demo. You'll create your source file and related files in this demo directory.

```
$ mkdir demo
$ cd demo
```

Now use the TNIX editor, **ed**, to create the demonstration program source file. The following command line invokes the editor and specifies that you want to create a file called asm:

```
$ ed asm
?asm
```

The editor responds with "?asm" to remind you that asm does not exist. Notice that the editor does **not** give a prompt to let you know it's ready for input.

Enter the Text. Now enter the editor command **a** (append text) and type in the program. Use the BACKSPACE key to erase typing mistakes.

a <CR>

```

-----Program starts here
|           |           |           | | |
|           |           |           |           |
|           |           |           |           |           |
|           |           |           |           |           |
V           V           V           V
;80186 DEMONSTRATION RUN PROGRAM
SECTION DEMO
ORG 100H ;START PROGRAM CODE AT ADDRESS 100
START MOVW BX,#TABLE ;SET TABLE POINTER
MOVW CX,#TSIZE ;SET PASS COUNTER
XORB AL,AL ;CLEAR ACCUMULATOR
ALOOP ADDB AL,[BX] ;ADD BYTE FROM TABLE AND
INCW BX ; POINT TO NEXT BYTE
LOOP ALOOP ;DECREMENTS CX AND LOOPS UNTIL CX=0
MOVW DX,#01007H ;I/O ADDRESS FOR EXIT SVC
OUT DX,AL ;CALL EXIT SVC
NOP ; TO END PROGRAM
NOP ; EXECUTION
;SRB POINTER
ORG 40H ;STORE SRB POINTER AT ADDRESS 40
BYTE 0H,0H,0H,44H ;POINT TO SRB FOR EXIT SVC
;SRB FOR EXIT SVC
BYTE 1AH ;1AH = FUNCTION CODE FOR EXIT SVC
;TABLE OF NUMBERS TO BE ADDED
TSIZE EQU 5 ;TABLE SIZE = 5
ORG 500H ;SET UP TABLE AT ADDRESS 500
TABLE BLOCK TSIZE
LIST DBG
END START
. <CR>

```

At the end of your text, enter a period on a line by itself. The editor will return to command mode.

Check for Errors. The following editor command displays the text you have entered. Check for typing mistakes.

```
1,$p <CR>
^  ^^
|  |
|  | -- represents the print command, which
|  | displays the lines in the designated range
|  |
|  | --- designates the last line in file
|  |
|  | ----- designates the first line in the file
```

If you made any mistakes, go ahead and fix them. If you're not familiar with **ed**, Table 3-1 lists the commands you need to add, delete, or replace any line. For more information on **ed**, refer to your 8560 System Users Manual.

Table 3-1
Basic 8560 Editing Commands

Command	Function
mm,nnp <CR>	Displays lines mm through nn.
nn <CR>	Makes line nn the current line.
d <CR>	Deletes the current line.
a <CR>	Adds text after the current line.
<line(s) of text>	
. <CR>	(Enter period on line by itself.)
c <CR>	Replaces the current line with the
<line(s) of text>	text you type in.
. <CR>	(Enter period on line by itself.)

Once your text is correct, enter the **w** command to write the text to the source file, asm:

```
w <CR>
902
```

The editor responds with the number of characters it wrote to the file.

3. Enter the command asm obj asml asm to reassemble your source code.

Link the Object Code

The linker creates an executable load file from one or more object files. Enter the following command to create a load file called load from your object file, obj. Be sure to capitalize all parameters as shown.

```
$ link -d -O obj -o load
```

The -d option causes the linker to pass the program symbols from the object file to the load file for programming debugging.

The files generated by the asm and link commands should now be in your working directory, demo. Enter the following command to list the files in your working directory:

```
$ ls  
asm  
asml  
load  
obj
```

Notice that there are now four files listed in your directory: obj and asml were created by the assembler, and load was created by the linker.

Download the Program to the 8540

Now it's time to download the object code produced by the 8560's linker into 8540 program memory.

Allocate Memory. Enter the following command to allocate a 4K-byte block of program memory to logical addresses 00000--00FFFH.

```
$ al 0
```

Zero Out Memory. Before you download any code, use the OS/40 f (fill) command to fill 8540 program memory with zeros. Later, when you examine memory, the zeros make it easy to identify the beginning and end of your code. Zeroing out memory has no effect on how the program is loaded. Enter the following command line to fill memory addresses 40--11F with zeros:

```
$ f 40 11f 00
```

Check that Memory is Filled with Zeros. Check the contents of memory with the OS/40 **d** (display) command. The display shows the data in hexadecimal format and also shows the corresponding ASCII characters. Display the contents of memory addresses 40--11F with the following command line:

```
$ d 40 11f
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
000040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
```

Download the Object Code. Enter the following command line to download the object code from the 8560 file load to 8540 program memory:

```
$ lo <load
      ^
      |
      load file
```

Download the Program Symbols. The source code for the demonstration program contains the directive LIST DBG. (Refer to Fig. 3-1.) Because of this directive, the object file contains a list of the symbols that appear in the source code, and the values associated with those symbols. Because you included the **-d** option in the link command line, those symbols were passed to the load file. Use the OS/40 **symlo** command to download those symbols into the symbol table in 8540 system memory.

```
$ symlo -s <load
```

The **-s** option means that both addresses and scalars are downloaded. If you omit the **-s** option, only addresses are downloaded. A scalar is a number that is not an address. In Fig. 3-1, the scalar TSIZE represents the length of a table.

Later in this demonstration, whenever you use a symbol in an OS/40 command line, OS/40 refers to the symbol table you just created to find the value that the symbol stands for.

You've assembled and linked the demonstration program and downloaded it into memory. Now skip ahead to the heading "Run the Demonstration Program."

CASE 3: DOWNLOAD FROM YOUR HOST TO THE 8540 OR 8550

This discussion gives some general instructions for downloading the demonstration program from an unspecified host computer to 8540 or 8550 program memory. If your 8540 is not equipped with the optional COM interface package, you cannot complete this part of the demonstration. Skip ahead to "Case 4: Patch the Program into Memory" for instructions. COM interface software is standard on the 8550.

Since we don't know what host computer you're using, this manual only provides a general outline for creating the demonstration program and downloading it to the 8540 or 8550. Once you have determined the command sequence appropriate for your host, record this information in the space provided in Fig. 3-3.

NOTE

This demonstration shows the 8540 commands that can also be used for an 8550 connected to an unspecified host computer. As a result, you can substitute 8550 for 8540 throughout the demonstration unless otherwise noted.

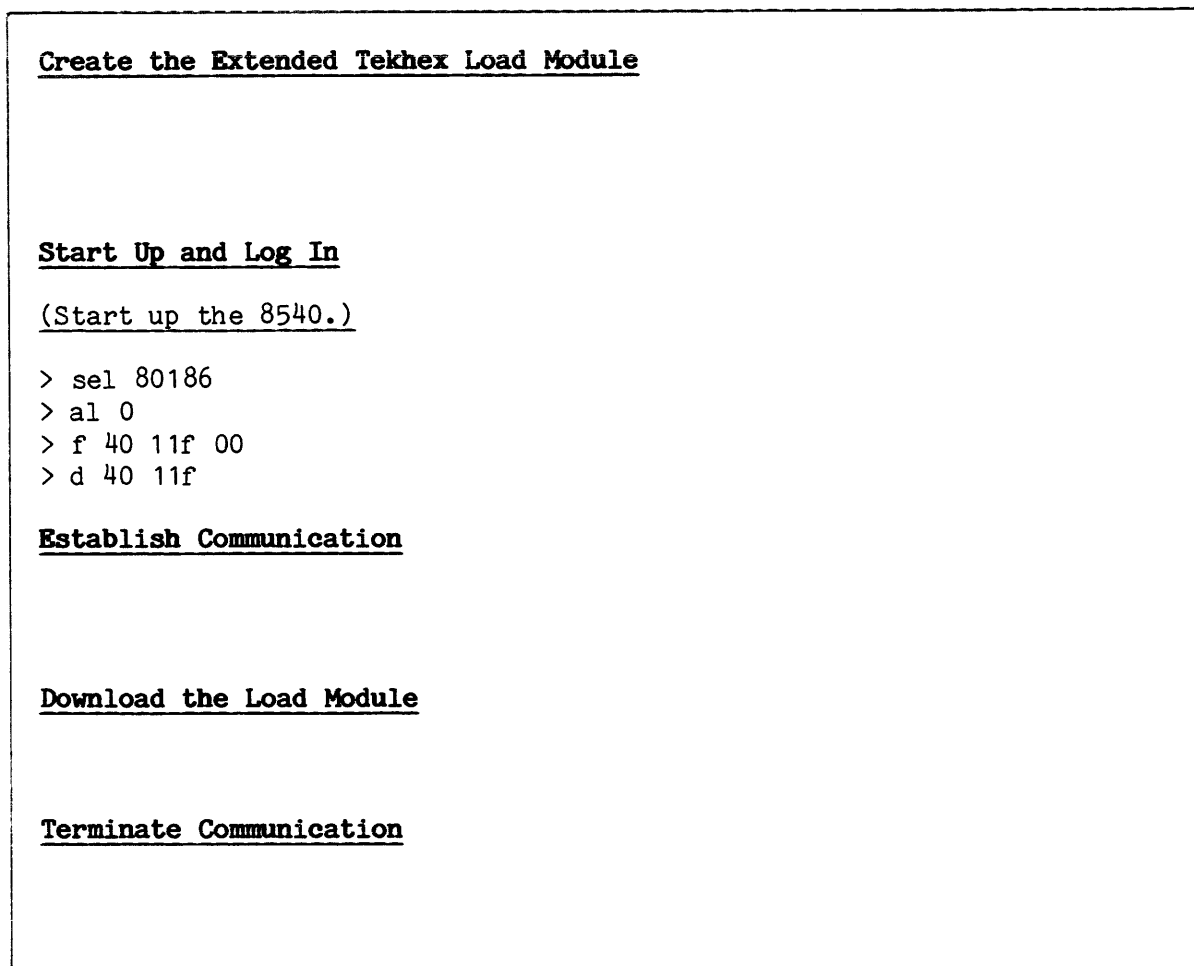


Fig. 3-3. Host computer commands for preparing demonstration program.

Create the Extended Tekhex Load Module

To download the object code to the 8540, the code must be in Extended Tekhex format, as shown in Fig. 3-4. You can create the load module in one of two ways:

1. Use your host computer's text editor and type in the load module.
2. Use your host computer's 80186/80188 Assembler:
 - Translate the demonstration program into the language of your host's 80186/80188 Assembler.

- Create and assemble the source file.
- Link the object code, if necessary.
- Translate the object code produced by the assembler or linker into Extended Tekhex format. The Intersystem Communication section of your System Users Manual provides a general algorithm for conversion to Extended Tekhex format.

Figure 3-4A shows an Extended Tekhex load module that contains the object code and program symbols for the demonstration program. Figure 3-4B gives the meanings of the different fields in the message blocks. If you have a host computer other than an 8560, you can create this load module and download it to your 8540 or 8550.

Start Up and Log In

Start up your 8540 and enter the following command to select the 80186 Emulator:

```
> sel 80186
```

NOTE

If you are using an 8550, enter either sel 80186 or sel 80188 depending on which Prototype Control Probe you have installed.

The **sel** command automatically sets the emulation mode to 0 and displays the current version number.

Allocate Memory. Enter the following command to allocate a 4K-byte block of program memory to logical addresses 00000--00FFFH.

```
> al 0
```

Zero Out Memory. Before you download any code, use the OS/40 **f** (fill) command to fill 8540 program memory with zeros. Later, when you examine memory, the zeros make it easy to identify the beginning and end of your code. Zeroing out memory has no effect on how the program is loaded. Enter the following command line to fill memory addresses 40--11F with zeros:

```
> f 40 11f 00
```

```

(A)

%2F6113 100BB0005B9050032C00207 43E2FBBA07 10EE9090
%1 262 324 0000000441A
%3B357 4DEM00 1035 05 15AL00P3 10815START3 100 15TABLE350025TSIZE15
%098 153100

(B)

FIRST DATA BLOCK: (Object Code for Addresses 100--112)

Header
|   Load Address   Object Code
|   |               |
|   V               V
=====
%2F6113 100BB0005B9050032C00207 43E2FBBA07 10EE9090

SECOND DATA BLOCK: (Object Code for Addresses 40--44)

Header
|   Load   Object
|   Address Code
|   |       |
|   V       V
=====
%1 262 324 0000000441A

SYMBOL BLOCK:

Header          Section
|   Section    Definition
|   Name       Field      Symbol Definition Fields
|   |         |           |
|   V         V           V
=====
%3B357 4DEM00 1035 05 15AL00P3 10815START3 100 15TABLE350025TSIZE15

TERMINATION BLOCK:

Header
|   Transfer
|   Address
|   |
|   V   V
=====
%098 153100

```

Fig. 3-4. 80186 demonstration run program: extended Tekhex format.

Check that Memory is Filled with Zeros. Check the contents of memory with the OS/40 **d** (display) command. The display shows the data in hexadecimal format and also shows the corresponding ASCII characters. Display the contents of memory addresses 40--11F with the following command line:

```
> d 40 11f
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
000040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0000F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
000110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
```

Download the Load Module. Be sure that your 8540 and your host computer are connected through an RS-232-C-compatible communications link. Then perform the following steps to download the Tekhex load module to 8540 program memory. Refer to the Intersystem Communication section of your System Users Manual to determine the commands and parameters that are appropriate for your host computer.

- a. Enter the 8540 **com** command to establish communication. The parameters of the **com** command are host-specific. Log on to your host and execute any necessary host initialization commands.
- b. Enter the command line that downloads the Tekhex load module to the 8540. This command line consists of a host computer command that performs the download, followed by a null character (CTRL-@ on most terminals) and a carriage return. The **com** command places the object code in 8540 program memory and puts the program symbols into the symbol table in 8540 system memory.
- c. Log off from your host and terminate the **com** command execution by entering the null character and then pressing the ESC key.

Once you've downloaded the program to the 8540, skip ahead to "Run the Demonstration Program."

CASE 4: PATCH THE PROGRAM INTO MEMORY

This discussion shows you how to patch the demonstration program into 8540 or 8550 program memory using the **p** command, and how to add the program symbols into the symbol table using the **adds** command.

Ordinarily, you would load the object code and symbols from a binary or hexadecimal load file, as illustrated for Cases 1, 2, and 3. The procedure presented here is **not** normally used for preparing a program for execution. Use this procedure only if you have no standard means for preparing the program but would still like to try out the demonstration program.

NOTE

This demonstration shows the 8540 commands that can also be used for an 8550. As a result, you can substitute 8550 for 8540 throughout the demonstration unless otherwise noted.

Start Up and Log In

Start up your 8540 and enter the following command to select the 80186/80188 Emulator:

```
> sel 80186
```

NOTE

For 8550 users, enter either **sel 80186** or **sel 80188** depending on which Prototype Control Probe you have installed.

The **sel** command automatically sets the emulation mode to 0 and displays the current version number.

Allocate Memory. Enter the following command to allocate a 4K-byte block of program memory to logical addresses 00000--00FFFH.

```
> al 0
```

Zero Out Memory. Before you patch in any code, use the OS/40 **f** (fill) command to fill 8540 program memory with zeros. Later, when you examine memory, the zeros make it easy to identify the beginning and end of your code. Enter the following command line to fill memory addresses 40--11F with zeros:

```
> f 40 11f 00
```

Now patch in the instructions that call the SVC (MOVW DX,#01007H; OUT DX,AL; and two NOPs):

```
> p 10D BA0710EE9090
```

Finally, patch in the Exit SVC information at address 40:

```
> p 40 000000441A
```

You'll check the contents of memory later in this demonstration.

Put Symbols into the Symbol Table. Later in this demonstration, you will use symbols from the demonstration program (START, LOOP, TSIZE, and TABLE) when communicating with OS/40. Whenever you use a symbol in a command line, OS/40 consults a symbol table in 8540 system memory to find the values that the symbol represent. Enter the following command line to add the program symbols and their values to the symbol table:

```
> adds START=100 ALOOP=108 -s TSIZE=5 TABLE=500
```

The **adds** command cannot provide all the symbol-related information that is provided by the **symlo** command in Cases 1 and 2 or the **com** command in Case 3. Because this information is missing, some of the displays you produce later in this demonstration will not match the symbolic displays shown in this manual. For more information on the **adds** command, refer to the Command Dictionary of your System Users Manual.

You've patched the demonstration program into program memory and placed the program symbols in the symbol table. Now it's time to run the program.

RUN THE DEMONSTRATION PROGRAM

For the rest of this demonstration, the commands you enter are shown in lowercase. If you are not logged in to an 8560, you may enter commands in either lowercase or uppercase. If you **are** using an 8560, you **must** enter the name of every command in lowercase. In addition, the 8560's system prompt is "\$", not ">".

Now that you've loaded the program into memory, you need to:

- Verify that the program was loaded correctly
- Put values into the table in memory for the program to add

CHECK MEMORY CONTENTS AGAIN

Before you loaded the program, you filled memory locations 40--11F with zeros. Look at the same memory area again with the following command line:

```
> d 40 11f
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
000040 00 00 00 44 1A 00 00 00 00 00 00 00 00 00 00  ...D.....
000050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
000060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
000070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
000080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
000090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
0000A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
0000B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
0000C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
0000D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
0000E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
0000F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
000100 BB 00 05 B9 05 00 32 C0 02 07 43 E2 FB BA 07 10  ....2...C....
000110 EE 90 90 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
```

The object code is loaded in two different blocks:

- The 80186 machine instructions are loaded at address 100 (specified by the first ORG directive in the source code).
- The information for the Exit SVC is loaded at address 40 (specified by the second ORG directive).

The contents of the table at address 500 are still undefined, but you'll put some values into the table in just a few minutes.

TURN ON SYMBOLIC DISPLAY

Enter the following command to tell the system to replace the hexadecimal display numbers with symbols from your program, where appropriate:

```
> symd on
```

DISASSEMBLE THE OBJECT CODE

The **di** (disassemble) command displays memory contents both in hexadecimal notation and in assembly language mnemonics. You can use the **di** command to verify that the object code in memory corresponds to your source code. Enter the following command to disassemble the memory area occupied by the executable part of your program:

```
> di 100 112
LOC      INST          MNEM   OPER

START
000100  BB0005        MOVW   BX,#0500

DEMO+000103
000103  B90500        MOVW   CX,#0005

DEMO+000106
000106  32C0             XORB   AL,AL

ALOOP
000108  0207             ADDB  AL,[BX]

DEMO+00010A
00010A  43               INC   BX

DEMO+00010B
00010B  E2FB            LOOP  $-03

DEMO+00010D
00010D  BA0710         MOVW  DX,#1007

DEMO+000110
000110  EE             OUT  DX,AL

LOC      INST          MNEM   OPER

DEMO+000111
000111  90             NOP

DEMO+000112
000112  90             NOP
```

Compare the `di` command display with the assembler listing you generated earlier or refer to Fig. 3-1.

The entire memory area your program uses (location 0 through 504, the end of the table) belongs to section DEMO. DEMO was declared by the SECTION directive in the source code. (If you used the `adds` command to create your symbols, as in Case 4, the `di` command display shows NO.SECTION as the section name.)

The LOC (location) column of the `di` command display contains information that lets you correlate the display with your assembler listing. The symbols START and ALOOP in the `di` command display correspond to the labels START and ALOOP in the source code. When a location in the display does not correspond to a label in the symbol table, the `di` command substitutes the

address of the instruction **relative to the beginning of the section**, as shown in the address field of your assembler listing.

If you haven't loaded the pertinent symbols and related information into the symbol table using a command such as **symlo**, the **di** command supplies only the absolute (actual) addresses in the LOC column. Since section DEMO begins at address 0, the relative address, or **offset**, is the same as the absolute address in this display. This offset feature is much more useful for sections that **don't** start at address 0.

You've seen your system use the symbol table to translate numbers into symbols making a display easier to read. Your system can also translate a symbol in a command line into an address. For example, since your system knows that the symbol START is equivalent to the address 100, you could have entered the **di** command in any of the following ways:

```
di 100 112
di START 112
di start start+12
di 100 START+12
di start,,10
```

Notice that a symbol can be entered in either lowercase or uppercase.

The feature that enables DOS/50 and OS/40 to correlate symbols from your program with the numbers they represent is **symbolic debug**.

PUT VALUES INTO THE TABLE IN MEMORY

The demonstration program sums five numbers from a table in memory. Use the **p** (patch) command to store the numbers 1, 2, 3, 4, and 5 in the table. Remember that the symbol TABLE represents the table's address.

```
> p table 0102030405
      ^      ^
      |      |
address of  string of bytes to be stored
table: 500  at addresses 500--504
```

CHECK THE CONTENTS OF THE TABLE

Use the **d** command to display the contents of the table. When you don't specify an upper boundary for the area to be displayed, the **d** command displays 16 bytes.

```

----- lower address: 500
|
|   -- upper address: omitted
|   (defaults to lower address + 0F)
|   |
V   V
=====
> d table
    0 1 2 3 4 5 6 7 8 9 A B C D E F
000500 01 02 03 04 05 27 EB 8F C3 3C EB B6 9D 2B AB DB .....'<...<.....

```

Notice that bytes 500--504 (the table) contain the values you patched in. Bytes 505--50F contain random data left over from previous system operations.

The following command displays only the contents of the table:

```

> d table table+tsize-1
    0 1 2 3 4 5 6 7 8 9 A B C D E F
000500 01 02 03 04 05 .....

```

CHECK THE CODE SEGMENT REGISTER

The CS Register must be zero for the demonstration run program to execute properly. Enter the following command to check the status of the CS Register:

```

> ds

PC/CS:IP      HD STATUS/CLOCK      AX/BX CX/DX SI/DI SP/BP SS/DS ES/F
000106        NMI=0 INTR=0 TEST=1      000F 0002 0000 0000 0000 0000
0000:0106     8MHZ                    0502 1007 0000 0000 0000 F006

      FLAGS . . . . OF DF IF TF  SF ZF . AF . PF . CF
      F006 X X X X  0 0 0 0   0 0 X 0   X 1 X 0

```

The CS Register is set to zero when you select the 80186/80188 Emulator. If CS is not zero (for example, if you have entered a RESET command), enter the following command to set CS to zero:

```

> s cs=0

```

START PROGRAM EXECUTION

Enter the **g** (go) command to start program execution at location START, the address specified by the END directive in the source code.

```
> g start
```

PC/CS:IP	MNEMONIC/DATA	AX/BX	CX/DX	SI/DI	SP/BP	SS/DS	ES/F
DEMO+000112							
000112	NOP	000F	0000	0000	0000	0000	0000
0000:0113	90	0505	1007	0000	0000	0000	F006
000112	<BREAK	>					

The program executes. When the Exit SVC occurs, the program breaks (stops), and the contents of the emulator registers are displayed. Symbol AL (lower-order byte of Register A) contains the sum of the numbers in the memory table: 1+2+3+4+5=0F.

NOTE

Register A (Symbol AX in the preceding display) is "000F". Symbol AL (lower-order byte of Register A) is "0F".

MONITOR PROGRAM EXECUTION

You have assembled, loaded, and executed the demonstration program. The rest of this demonstration shows you some commands for monitoring program execution. These commands let you watch the changes in the emulator's registers and observe each instruction's effect as the program proceeds.

TRACE ALL INSTRUCTIONS

The **tra** (trace) command lets you observe the changes in the 80186 Emulator registers as the program proceeds. When you enter a **tra** command and then start execution with the **g** command, display lines are sent to the system terminal. As each instruction executes, the display line shows the instruction (as in the **di** command display) and the contents of the registers after that instruction has executed. Enter the following command to trace all of the program's instructions:

```
> tra all
```

Enter the command **g start** (or **g 100**) to resume program execution at the beginning of the program:

> g start

The following display shows the trace of each instruction as the program executes. Remember that you can type CTRL-S to suspend the display and CTRL-Q to resume the display.

PC/CS:IP	MNEMONIC/DATA	AX/BX	CX/DX	SI/DI	SP/BP	SS/DS	ES/F
START							
000100	MOVW BX,#0500	0063	0000	0000	0000	0000	0000
0000:0103	BB0005	0500	1007	0000	0000	0000	F006
DEMO+000103							
000103	MOVW CX,#0005	0063	0005	0000	0000	0000	0000
0000:0106	B90500	0500	1007	0000	0000	0000	F006
DEMO+000106							
000106	XORB AL,AL	0000	0005	0000	0000	0000	0000
0000:0108	32C0	0500	1007	0000	0000	0000	F046
ALOOP							
000108	ADDB AL,[BX]	0001	0005	0000	0000	0000	0000
0000:010A	0207	0500	1007	0000	0000	0000	F046
DEMO+00010A							
00010A	INC BX	0001	0005	0000	0000	0000	0000
0000:010B	43	0501	1007	0000	0000	0000	F002
DEMO+00010B							
00010B	LOOP \$-03	0001	0004	0000	0000	0000	0000
0000:0108	E2FB	0501	1007	0000	0000	0000	F002
ALOOP							
000108	ADDB AL,[BX]	0003	0004	0000	0000	0000	0000
0000:010A	0207	0501	1007	0000	0000	0000	F046
DEMO+00010A							
00010A	INC BX	0003	0004	0000	0000	0000	0000
0000:010B	43	0502	1007	0000	0000	0000	F002
DEMO+00010B							
00010B	LOOP \$-03	0003	0003	0000	0000	0000	0000
0000:0108	E2FB	0502	1007	0000	0000	0000	F002
ALOOP							
000108	ADDB AL,[BX]	0006	0003	0000	0000	0000	0000
0000:010A	0207	0502	1007	0000	0000	0000	F006

PC/CS:IP	MNEMONIC/DATA	AX/BX	CX/DX	SI/DI	SP/BP	SS/DS	ES/F
DEMO+00010A							
00010A	INC BX	0006	0003	0000	0000	0000	0000
0000:010B	43	0503	1007	0000	0000	0000	F006
DEMO+00010B							
00010B	LOOP \$-03	0006	0002	0000	0000	0000	0000
0000:0108	E2FB	0503	1007	0000	0000	0000	F006
ALoop							
000108	ADDB AL,[BX]	000A	0002	0000	0000	0000	0000
0000:010A	0207	0503	1007	0000	0000	0000	F002
DEMO+00010A							
00010A	INC BX	000A	0002	0000	0000	0000	0000
0000:010B	43	0504	1007	0000	0000	0000	F002
DEMO+00010B							
00010B	LOOP \$-03	000A	0001	0000	0000	0000	0000
0000:0108	E2FB	0504	1007	0000	0000	0000	F002
PC/CS:IP MNEMONIC/DATA AX/BX CX/DX SI/DI SP/BP SS/DS ES/F							
ALoop							
000108	ADDB AL,[BX]	000F	0001	0000	0000	0000	0000
0000:010A	0207	0504	1007	0000	0000	0000	F016
DEMO+00010A							
00010A	INC BX	000F	0001	0000	0000	0000	0000
0000:010B	43	0505	1007	0000	0000	0000	F006
DEMO+00010B							
00010B	LOOP \$-03	000F	0000	0000	0000	0000	0000
0000:010D	E2FB	0505	1007	0000	0000	0000	F006
DEMO+00010D							
00010D	MOVW DX, 1007	000F	0000	0000	0000	0000	0000
0000:0110	BA0710	0505	1007	0000	0000	0000	F006
DEMO+000110							
000110	OUT DX,AL	0000F	0000	0000	0000	0000	0000
0000:0111	EE	0505	1007	0000	0000	0000	F006
000110	<BREAK TRACE>						

After the accumulator is cleared, it begins to store the sum of the numbers being added. The ADDB AL,[BX] instruction adds a number from the table into the accumulator. At the end of the program, the accumulator contains the sum of the numbers you put into the table.

Register CX, the pass counter, is set to contain 5 (TSIZE) at the beginning of the program. It decreases by one because of the LOOP instruction each

effect at the same time, it is useful to see which selections are active. Check your trace status with the following command line:

```
> tra
TRACE  ALL,000000,00FFFF
TRACE  JMP,ALoop,DEMO+000112
```

The preceding display shows that the **tra** command is set to trace all instructions for addresses 0--0108, to trace only jump instructions for addresses 0108--0112, and to trace all instructions again for addresses 0113--FFFF.

Start your program again with the **g** command. The following trace is displayed:

> g start

PC/CS:IP	MNEMONIC/DATA	AX/BX	CX/DX	SI/DI	SP/BP	SS/DS	ES/F
START							
000100	MOVW BX, #0500	000F	0000	0000	0000	0000	0000
0000:0103	BB0005	0500	1007	0000	0000	0000	F006
DEMO+000103							
000103	MOVW CX, #0005	000F	0005	0000	0000	0000	0000
0000:0106	B90500	0500	1007	0000	0000	0000	F006
DEMO+000106							
000106	XORB AL, AL	0000	0005	0000	0000	0000	0000
0000:0108	32C0	0500	1007	0000	0000	0000	F046
DEMO+00010B							
00010B	LOOP \$-03	0001	0004	0000	0000	0000	0000
0000:0108	E2FB	0501	1007	0000	0000	0000	F002
DEMO+00010B							
00010B	LOOP \$-03	0003	0003	0000	0000	0000	0000
0000:0108	E2FB	0502	1007	0000	0000	0000	F002
PC/CS:IP	MNEMONIC/DATA	AX/BX	CX/DX	SI/DI	SP/BP	SS/DS	ES/F
DEMO+00010B							
00010B	LOOP \$-03	0006	0002	0000	0000	0000	0000
0000:0108	E2FB	0503	1007	0000	0000	0000	F006
DEMO+00010B							
00010B	LOOP \$-03	000A	0001	0000	0000	0000	0000
0000:0108	E2FB	0504	1007	0000	0000	0000	F002
DEMO+000110							
000110	OUT DX, AL	000F	0000	0000	0000	0000	0000
0000:0111	EE	0505	1007	0000	0000	0000	F006
000110	<BREAK TRACE>						

In the preceding display, observe that Register CX, the pass counter, is decremented; Register BX, the table pointer, is incremented; and AL, the accumulator, stores the sum of the numbers from the table. The contents of these registers are the same as the previous trace all display. With the trace jump instruction selection in effect, the instructions within the loop are not displayed.

SET A BREAKPOINT AFTER A SPECIFIC INSTRUCTION

Now that you've seen how the program adds the numbers together, let's add only the third and fourth numbers from the table. To perform this task, you want the pass counter to contain 2, and the table pointer to contain 502, the address of the third number in the table. You can accomplish these changes without altering the object code in memory. First, stop program execution after the pass counter and the table pointer have been set. Next, while the program is stopped, enter new values for the pass counter and the table pointer. When execution resumes, the program will treat the new values as if they were the original programmed values.

Enter the following command line to trace all of the instructions as the program executes:

```
> tra all
```

Check the status of the trace with the following command line:

```
> tra  
TRACE    ALL,DEMO+000000,OFFF
```

The last `tra all` command makes earlier trace selections obsolete.

Set Breakpoints to Stop Program Execution

Now set a breakpoint to stop the program after the table pointer and the pass counter have been set. The following command causes the program to stop after it executes the `MOVW BX,#TABLE` instruction at address 103:

```
> bk 1 103  
  = ==  
  ^ ^  
  | |  
  | -- Breakpoint Address  
  |  
  ----- Breakpoint Number  
           (can be 1, 2, or 3)
```

Use the `g` command to start program execution:

> g start

```

PC/CS:IP  MNEMONIC/DATA          AX/BX CX/DX SI/DI SP/BP SS/DS ES/F
START
000100      MOVW      BX,#0500    000F 0000 0000 0000 0000 0000
0000:0103      BB0005          0500 1007 0000 0000 0000 F006

DEMO+000103
000103      MOVW      CX,#0005    000F 0005 0000 0000 0000 0000
0000:0106      B90500          0500 1007 0000 0000 0000 F006

000103 <BREAK      TRACE, BKPT1>
    
```

The **tra all** command displays all instructions up to and including the instruction at the breakpoint.

Set New Values in Pass Counter and Table Pointer; Check Results

Now that you've reached the breakpoint, you can change the contents of the registers while execution is stopped. The displayed breakpoint shows that Register CX, the pass counter, contains 5, and Register BX, the table pointer, contains the address 500. Use the **s** (set) command to set the number of passes to two and set the table pointer to 502:

> s bx=502 cx=2

The **s** command does not produce a display, but you can use the **ds** (display status) command to check the values in the registers you changed. The **ds** command displays the contents of each emulator register and status flag. Check the result of the previous **s** command with the following command line:

```

> ds
PC/CS:IP      HD STATUS/CLOCK      AX/BX CX/DX SI/DI SP/BP SS/DS ES/F
000106        NMI=0 INTR=0 TEST=1    000F 0002 0000 0000 0000 0000
0000:0106      8MHZ                    0502 1007 0000 0000 0000 F006

          FLAGS . . . . OF DF IF TF   SF ZF . AF   . PF . CF
          F060 X X X X   0 0 0 0     0 0 X 0   . X 1 X 0
    
```

The **ds** command shows that the pass counter and table pointer now contain the new values.

Resume Program Execution

If you enter the **g** command with no parameters, program execution starts where it left off. Resume program execution after the breakpoint with the following command:

> g

PC/CS:IP	MNEMONIC/DATA	AX/BX	CX/DX	SI/DI	SP/BP	SS/DS	ES/F
DEMO+000106							
000106	XORB AL,AL	0000	0002	0000	0000	0000	0000
0000:0108	32C0	0502	1007	0000	0000	0000	F046
ALoop							
000108	ADDB AL,[BX]	0003	0002	0000	0000	0000	0000
0000:010A	0207	0502	1007	0000	0000	0000	F006
DEMO+00010A							
00010A	INC BX	0003	0002	0000	0000	0000	0000
0000:010B	43	0503	1007	0000	0000	0000	F006
DEMO+00010B							
00010B	LOOP \$-03	0003	0001	0000	0000	0000	0000
0000:0108	E2FB	0503	1007	0000	0000	0000	F006
ALoop							
000108	ADDB AL,[BX]	0007	0001	0000	0000	0000	0000
0000:010A	0207	0503	1007	0000	0000	0000	F002
PC/CS:IP MNEMONIC/DATA AX/BX CX/DX SI/DI SP/BP SS/DS ES/F							
DEMO+00010A							
00010A	INC BX	0007	0001	0000	0000	0000	0000
0000:010B	43	0504	1007	0000	0000	0000	F002
DEMO+00010B							
00010B	LOOP \$-03	0007	0000	0000	0000	0000	0000
0000:010D	E2FB	0504	1007	0000	0000	0000	F002
DEMO+00010D							
00010D	MOVW DX, 1007	0007	0000	0000	0000	0000	0000
0000:0110	BA0710	0504	1007	0000	0000	0000	F002
DEMO+000110							
000110	OUT DX,AL	0007	0000	0000	0000	0000	0000
0000:0111	EE	0504	1007	0000	0000	0000	F002
000110	<BREAK TRACE>						

Notice that the program performed two passes through the loop, and that the program added the third and fourth numbers in the table: 3+4=7.

SUMMARY OF 80186 EMULATOR DEMONSTRATION RUN

You have assembled, loaded, executed, and monitored the demonstration run program. Review the commands you used:

- **sel**---selects the 80186/80188 Assembler and 80186/80188 Emulator
- **asm**---creates object code from an assembly language program
- **link**---links object code into a load module
- **f**---fills an area of memory with a specified value
- **d**---displays memory contents in ASCII and hexadecimal format
- **lo**---loads object code into memory
- **symlo**---loads program symbols for use in symbolic debug
- **di**---translates memory contents into assembly language mnemonics
- **p**---patches a string of bytes into memory
- **symd**---enables symbolic debug display
- **g**---begins or resumes program execution
- **tra**---selects instructions to be traced during program execution
- **bk**---sets a breakpoint
- **s**---modifies emulator registers
- **ds**---displays the contents of the emulator registers
- **adds**---adds symbols to the symbol table

DELETE THE DEMONSTRATION RUN FILES

Now that you've finished the demonstration run, you can delete the source file, object file, listing file, and load file. If you're using an 8550, the source and load files are still available to you on the 8086/8088 Emulator installation disk. If you're using an 8560, remember that once you delete the source file, asm, there is no way of recovering it.

DELETE 8550 FILES

If your files are on the 8550, use the following procedure to delete them. First use the **user** command to move from the DEMO directory back into the directory you were in at the start of the demonstration. Recall that you marked that directory with the brief name /ROOT.

```
> user /ROOT
```

Now enter the following command to delete the DEMO directory and the files it contains:

```
> del DEMO/* DEMO
Delete ASM ? y
Delete LOAD ? y
Delete OBJ ? y
Delete ASML ? y
Delete DEMO ? y
```

Before deleting each file, DOS/50 asks you whether you really want to delete it. You type "y" for yes.

DELETE 8560 FILES

If your files are on the 8560, use the following procedure to delete them. Enter the following command to remove all files in the working directory, including the source file:

```
$ rm *
```

Now move from the demo directory back into the parent directory and remove the demo directory itself:

```
$ cd ..
$ rmdir demo
```

TURN OFF YOUR SYSTEM

For instructions on turning off your 8540 or 8550, refer to the Learning Guide of your System Users Manual.

Section 4

TECHNICAL INFORMATION

INTRODUCTION

This section contains technical reference material for the 80186/80188 Emulator and its associated prototype control probes. The technical reference material includes emulator timing relationships, interface diagrams, the probe Power Supply Board calibration procedure, and specifications.

EMULATOR TIMING

The signals between the prototype and the emulating microprocessor are buffered. Therefore, some timing differences exist between the 80186/80188 Emulator and an 80186 or 80188 microprocessor inserted directly into the prototype.

Table 4-1 lists the emulator/microprocessor timing differences for the 80186/80188 Emulator operating in the RMX mode (worst-case conditions). Figures 4-1 (Parts 1 and 2), 4-2, 4-3, and 4-4 contain timing diagrams corresponding to the signals listed in Table 4-1.

NOTE

The numbers in the left column of Table 4-1 are used to identify the signals in the timing diagrams.

Table 4-1
80186 Emulator Timing Differences (RMX Mode)

No.	Symbol	Parameter	Processor		Probe		Units
			Min.	Max.	Min.	Max.	
1	TDVCL	Data in Setup (A/D)	20		42		ns
2	TCLDX	Data in Hold (A/D)	10		0		ns
3	TARYHCH	Asynchronous Ready (AREADY) active setup time (a)	20		42		ns
4	TARYLCL	AREADY Inactive Setup Time	35		57		ns
5	TCHARYX	AREADY Hold Time	15		0		ns
6	TSRYCL	Synchronous Ready (SREADY) Transition Setup Time	35		57		ns
7	TCLSRY	SREADY Transition Hold Time	15		0		ns
8	THVCL	HOLD Setup (a)	25		43		ns
9	TINVCH	INTR, NMI, TEST(L), TIMERIN, Setup (a)	25		41		ns
10	TINVCL	DRQ0, DRQ1, Setup (a)	25		41		ns
11	TCLAV	Address Valid Delay	10	44	10	46	ns
12	TCLAX	Address Hold	10		12		ns
13	TCLAZ	Address Float Delay	TCLAX	35	TCLAX-17	37	ns
14	TCHCZ	Command Lines Float Delay		45		71	ns
15	TCHCV	Command Lines Valid Delay (after float)		55		78	ns
16	TLHLL	ALE Width	TCLCL-35		1/2TCLCL-16		ns
17	TCHLH	ALE Active Delay		35		29	ns
18	TCHLL	ALE Inactive Delay		35		24	ns

Table 4-1 (cont)

No.	Symbol	Parameter	Processor		Probe		Units
			Min.	Max.	Min.	Max.	
19	TLLAX	Address Hold to ALE Inactive	TCHCL-25		TCHCL-37		ns
20	TCLDV	Data Valid Delay	10	44	0	46	ns
21	TCHDX	Data Hold Time	10		0		ns
22	TWHDX	Data Hold After WR(L)	TCLCL-40		TCLCL-51		ns
23	TCVCTV	Control Active Delay1	10	70	-4	52	ns
24	TCHCTV	Control Active Delay2	10	55	3	49	ns
25	TCVCTX	Control Inactive Delay	10	55	1	52	ns
26	TAZRL	Address Float to RD(L) Active	0		0		ns
27	TCLRL	RD(L) Active Delay	10	70	6	93	ns
28	TCLRH	RD(L) Inactive Delay	10	55	8	50	ns
29	TRHAV	RD(L) Inactive to Address Active	TCLCL-40		TCLCL-67		ns
30	TCLHAV	HLDA Valid Delay	10	50	1	60	
31	TRLRH	RD(L) Width	2TCLCL-50		2TCLCL-85		ns
32	TWLWH	WR(L) Width	2TCLCL-40		2TCLCL-70		ns
33	TAVAL	Address Valid to ALE Low	TCLCH-25		TCLCH-44		ns
34	TCHSV	Status Active Delay	10	55	0	57	ns
35	TCLSH	Status Inactive Delay	10	55	0	57	ns
36	TCLTMV	Timer Output Delay		60		62	ns
37	TCLRO	Reset Delay		60		77	ns
38	TCHQSV	Queue Status Delay		35		37	ns

Table 4-1 (cont)

No.	Symbol	Parameter	Processor		Probe		Units
			Min.	Max.	Min.	Max.	
39	TCLCSV	Chip-Select Active Delay		66		68	ns
40	TCXCSX	Chip-Select Hold from Command Inactive	35		39		ns
41	TCHCSX	Chip-Select Inactive Delay	10	35	1	40	ns
42	TCKIN	CLKIN Period	62.5	250	62.5	250	ns
43	TCKHL	CLKIN Fall Time		10		10	ns
44	TCKLH	CLKIN Rise Time		10		10	ns
45	TCLCK	CLKIN Low Time	25		25		ns
46	TCHCK	CLKIN High Time	25		25		ns
47	TCICO	CLKIN of CLKOUT Skew		50		98	ns
48	TCLCL	CLKOUT Period	125	500	125	500	ns
49	TCLCH	CLKOUT Low Time	1/2TCLCL -7.5		1/2TCLCL -8.5		ns
50	TCHCL	CLKOUT High Time	1/2TCLCL -7.5		1/2TCLCL -6.5		ns
51	TCH1CH2	CLKOUT Rise Time		15		15	ns
52	TCL2CL1	CLKOUT Fall Time		15		15	ns

a To guarantee recognition at next clock.

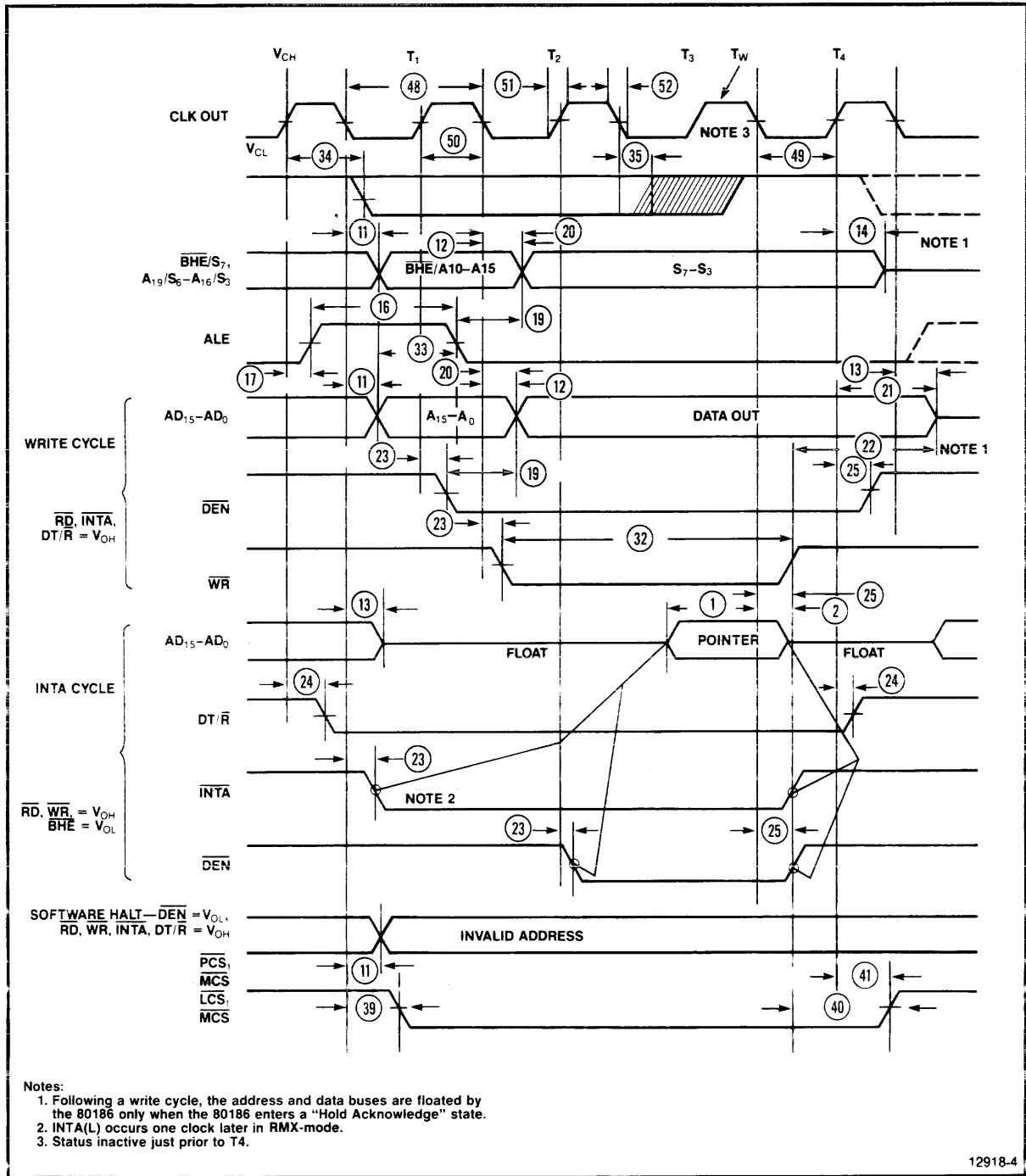


Fig. 4-1. 80186 RMX Mode timing diagram (Part 1 of 2).

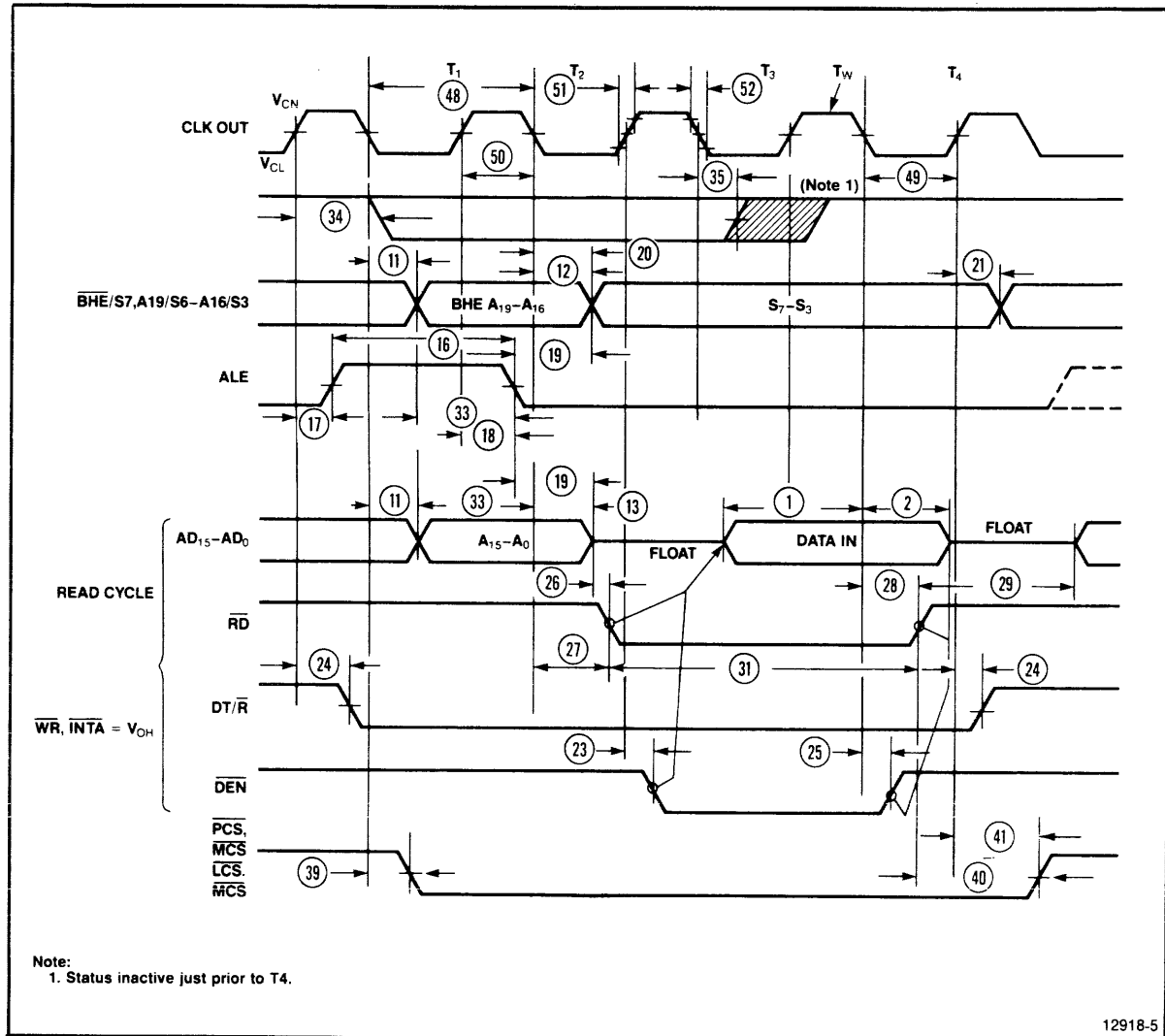
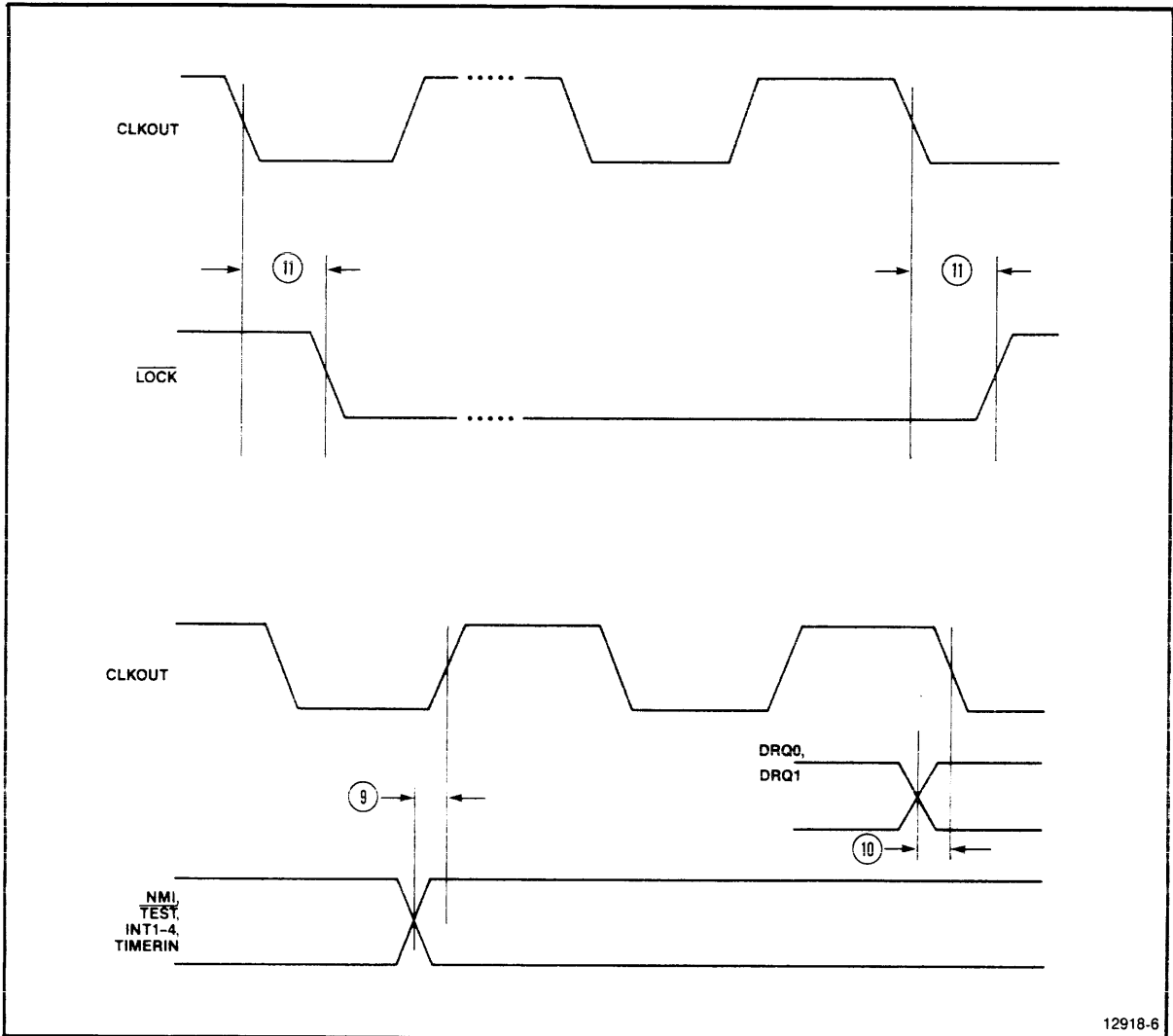
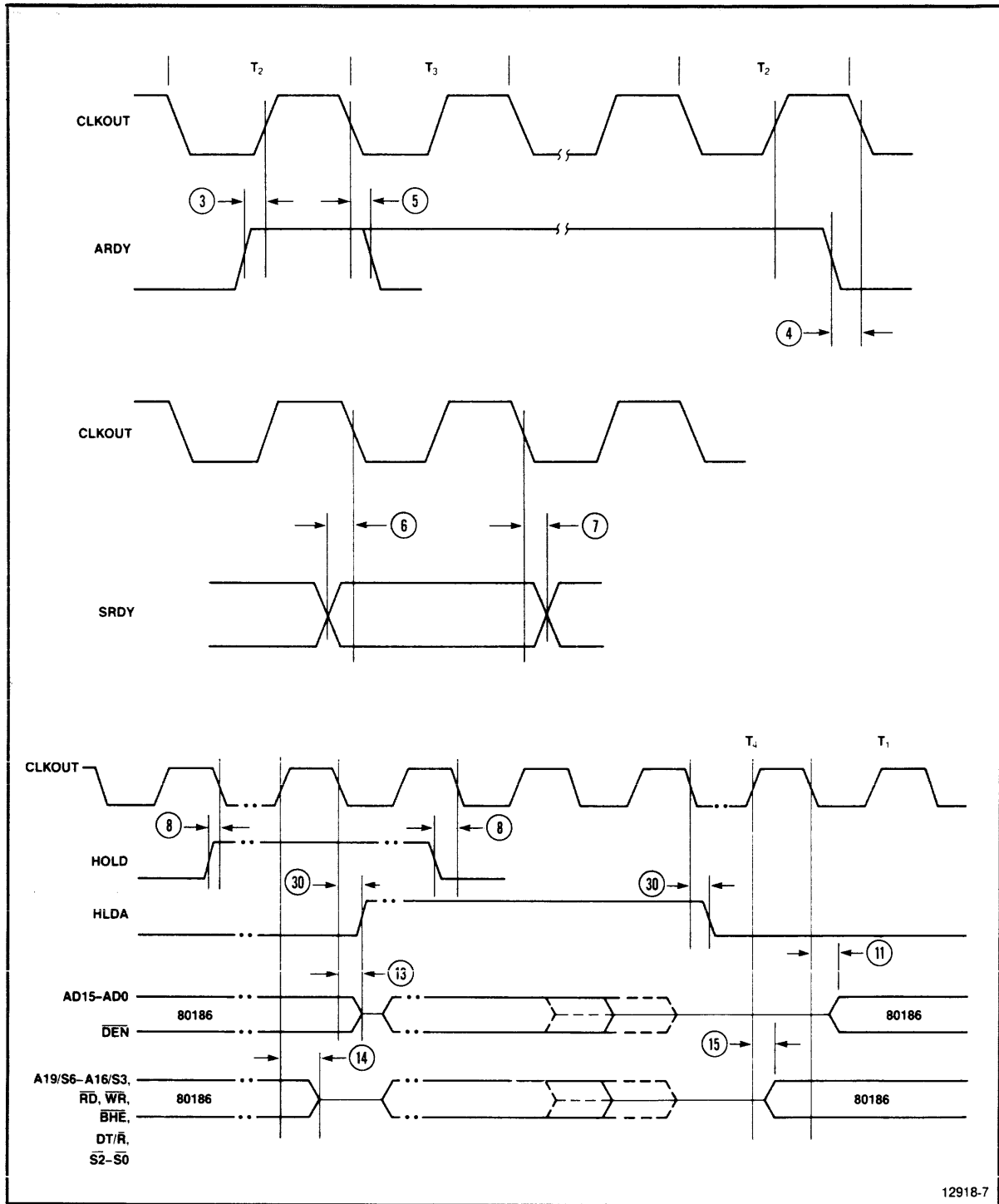


Fig. 4-1. 80186 RMX Mode timing diagram (Part 2 of 2).



12918-6

Fig. 4-2. Clock timing diagram.



12918-7

Fig. 4-3. HOLD-HOLDA timing diagram.

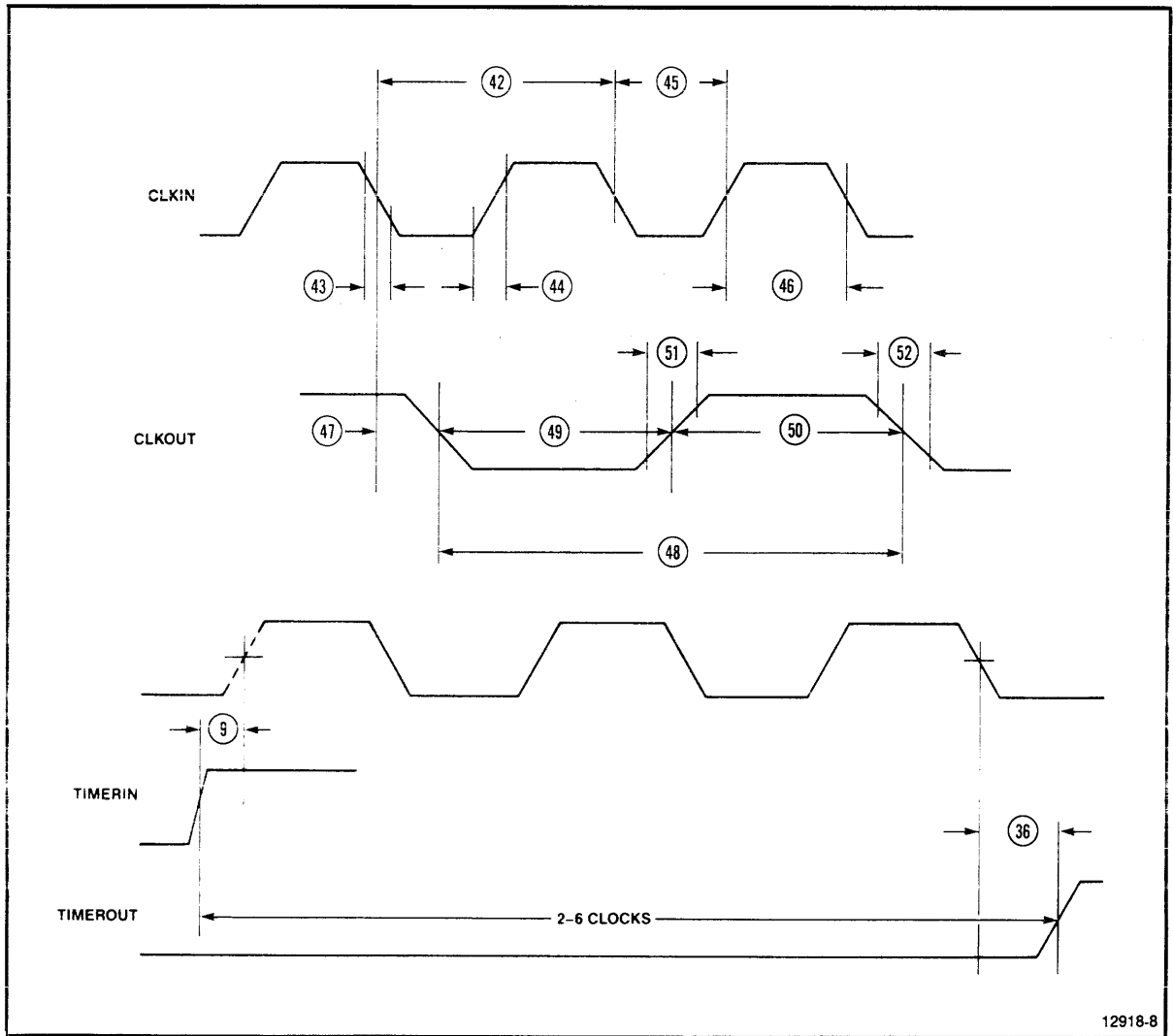


Fig. 4-4. 80186 Timer timing diagram.

PROBE/PROTOTYPE INTERFACE DIAGRAM

(To be provided.)

PROBE POWER SUPPLY CALIBRATION

The Power Supply Boards in both 80186 and 80188 Prototype Control Probes are identical. The Power Supply Board generates two voltages required by the prototype control probe: +4.9 Vdc and +5.0 Vdc. The +4.9 Vdc supply powers part of the circuitry on the Buffer Board of a prototype control probe. The +5.0 Vdc supply provides power to the 68-pin probe plug.

CAUTION

Calibrating the two power supply voltages requires the prototype control probe to be disassembled. Before starting the calibration procedure determine for sure that the power supply voltages require adjustment. Refer to the disassembly procedures for the prototype control probe contained in Section 6 of this manual under "Prototype Control Probe Disassembly Procedure".

The following text describes the calibration procedure used to adjust these two voltages. This procedure applies equally to the Power Supply Boards within each of the two probes.

EQUIPMENT REQUIRED

- TEKTRONIX 8550 Microcomputer Development Lab or 8540 Integration Unit, with 80186/80188 Emulator boards and 80186 or 80188 Prototype Control Probe installed.
- DVM with 100 uV resolution and +0.1% accuracy (TEKTRONIX DM501 or equivalent).

PROCEDURE

1. Ensure that primary power (115 Vac or 230 Vac) to the development system is OFF.
2. Disassemble the prototype control probe so that the Power Supply Board is available to perform the calibration adjustments. See the procedures in Section 6 of this manual under "Prototype Control Probe Disassembly Procedure".

3. Connect the DMV's negative test probe to one of the screws that holds the Power Supply Board to the top cover (ground).
4. Connect the DMV's positive test probe to TP1061 on the Power Supply Board. (See Fig. 4-5).
5. Turn on the DVM and the microcomputer development system.
6. Enter the following command:

 > sel 80186
7. Observe the voltage measured by the DVM.
8. Adjust R1051 on the Power Supply Board until the DVM reads +4.92 Vdc (+/- 20mV).
9. Disconnect the DMV's positive test probe from TP1061.
10. Connect the DMV's positive test probe to TP1011 on the Power Supply Board. (See Fig. 4-5.)
11. Observe the voltage measured by the DVM.
12. Adjust R1011 on the Power Supply Board until the DVM reads +5.0 Vdc.
13. Turn off power to all equipment.
14. Disconnect the DVM from the Power Supply Board.
15. Reassemble the prototype control probe interface assembly in reverse order of disassembly.

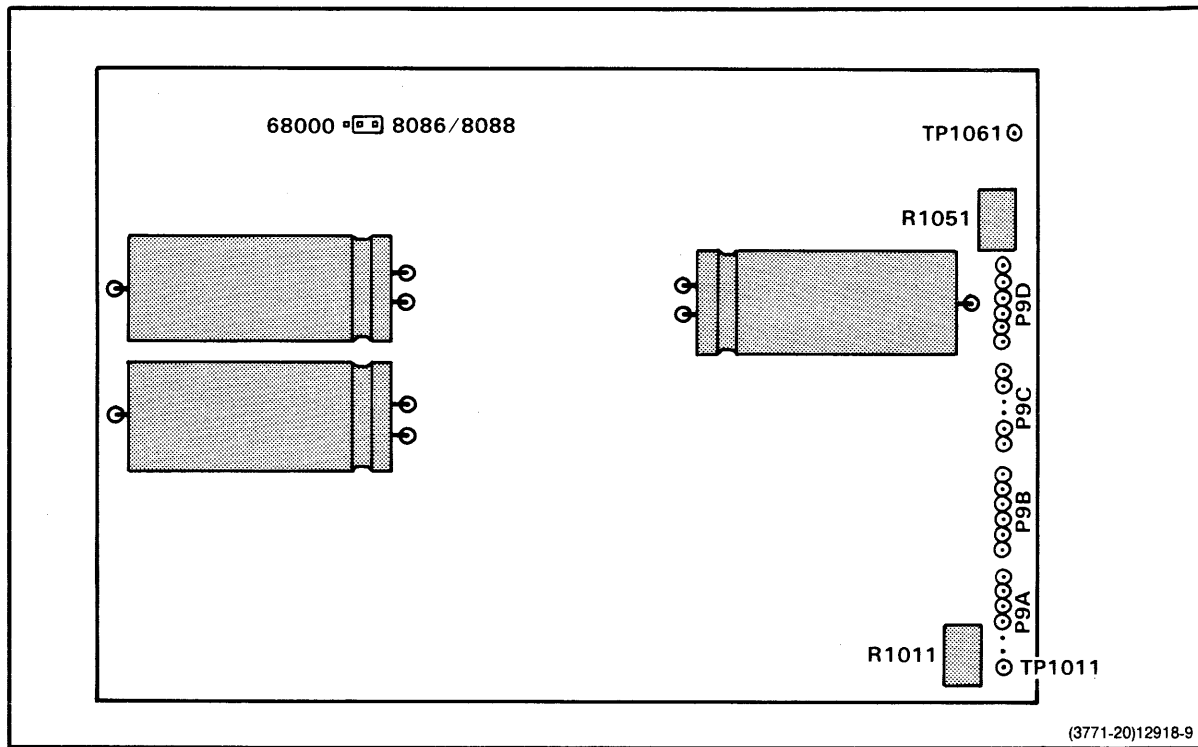


Fig. 4-5. Probe Power Supply Board.

SPECIFICATIONS

Table 4-2 contains the Electrical Characteristics, Table 4-3 contains the Environmental Characteristics, and Table 4-4 contains the Physical Characteristics, for the 80186/80188 Emulator.

**Table 4-2
80186/80188 Emulator Electrical Characteristics**

Characteristics	Performance Requirement	Supplemental Information
Supply Voltage	+5.2 Vdc $\pm 1\%$ / -2% +12.0 Vdc $\pm 5\%$ -12.0 Vdc $\pm 5\%$	-12.0 Vdc used by prototype control probe only
Current (typical)		+5.2 Vdc @ 6.94 A +12.0 Vdc @ 1.03 A -12.0 Vdc @ 0.40 A
Power Dissipation		80 W (max.) Selected 65 W (max.) Not Selected

**Table 4-3
80186/80188 Emulator Environmental Characteristics**

Characteristics	Description
Temperature	
Operating	0 C to +50 C (+32 F to +122 F)
Storage	-55 C to +75 C (-67 F to +167 F)
Relative Humidity	90% maximum non-condensing
Altitude	
Operating	5 000 m (15,000 ft) maximum
Storage	16 400 m (50,000 ft) maximum

Table 4-4
80186/80188 Emulator Physical Characteristics

Characteristics	Height	Length	Width
80186/80188 Emulator Boards			
Board I	195 mm (7.68 in)	280 mm (11.0 in)	
Board II	195 mm (7.68 in)	280 mm (11.0 in)	
Board III	195 mm (7.68 in)	280 mm (11.0 in)	
80186/80188 Prototype Control Probe			
Interface Assembly	102 mm (4.0 in)	236 mm (9.25 in)	185 mm (7.25 in)
Power Supply Board	166 mm (6.5 in)	107 mm (4.2 in)	
Control Board	166 mm (6.5 in)	107 mm (4.2 in)	
Buffer Board	166 mm (6.5 in)	107 mm (4.2 in)	
CPU Board	166 mm (6.5 in)	107 mm (4.2 in)	

Section 5

JUMPERS

INTRODUCTION

This section defines the jumpers and straps that are located on the emulator boards and prototype control probe boards. These jumpers and straps enable you to configure your 80186/80188 Emulator to suit your prototype application. In addition, this section contains procedures to make changes to the jumpers on the development system's program memory boards.

NOTE

In this section, jumpers are described in terms of "Normal" and "Option" positions. "Normal" refers to the factory setting or default position. "Option" refers to an optional setting for the jumper. In both cases, the terms "Normal" and "Option" have been followed by pin numbers (x-x) to indicate where the jumper block must be placed for that setting.

EMULATOR BOARDS JUMPERS AND STRAPS

The following text describes each jumper and strap located on Board I and Board II. There are no jumpers on Board III.

NOTE

To access the jumpers on the emulator boards, you must remove the top cover from the development system and take the emulator boards from the card cage. Disassembly procedures are contained in Section 6 of this manual under "Installing the Emulator Boards and Prototype Control Probe". Complete steps 1--3 of these procedures.

BOARD I JUMPERS AND STRAP

Board I has one jumper and one strap. Figure 5-1 shows the jumper and strap locations and their factory settings (default positions).

P1086

This jumper controls Trigger Trace Analyzer (TTA) strobing during mode 3 operation. P1086 has two positions:

Normal (2-3) TTA strobing is disabled during mode 3.

Option (1-2) TTA strobing is enabled during mode 3.
(This position is used only as a diagnostic tool.)

W4037

This three-pin strap is reserved for future use.

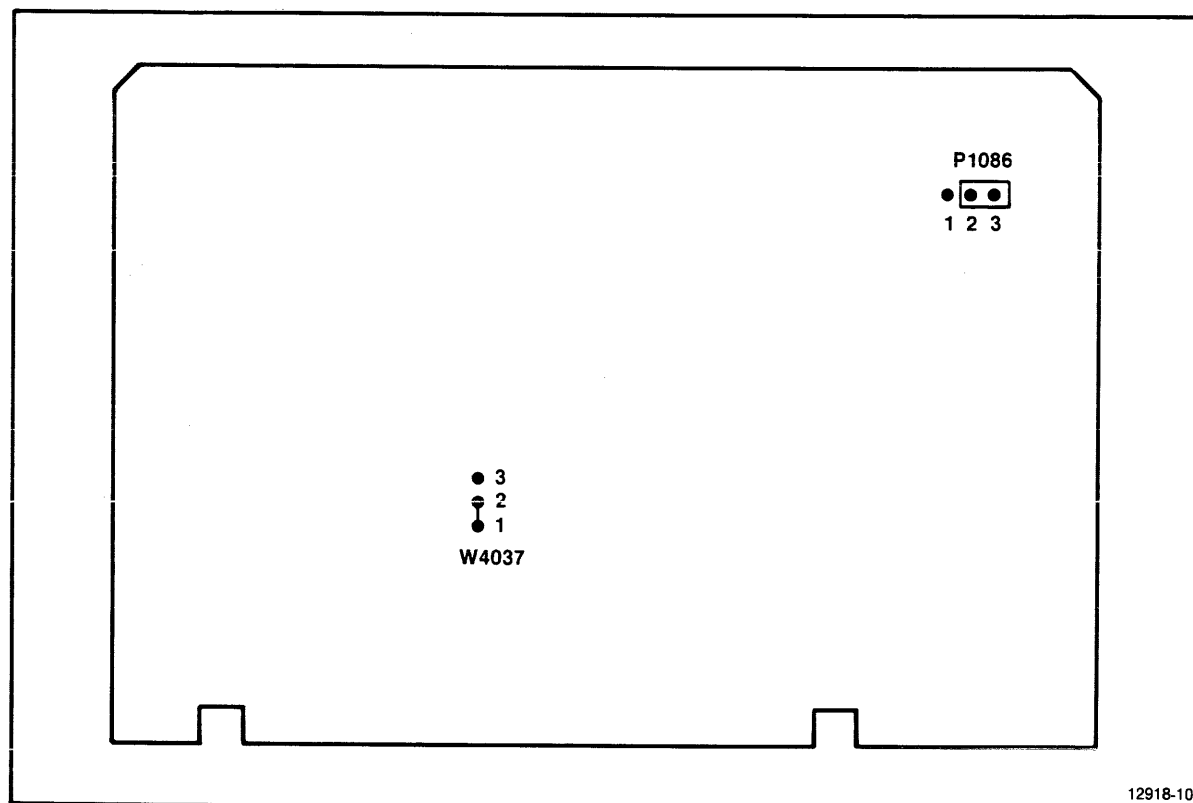


Fig. 5-1. Board I jumper and strap locations.

BOARD II JUMPERS

Board II has four jumpers and one indicator. Figure 5-2 shows the jumper locations and their factory settings (default positions). Figure 5-2 also shows DS3077, the Processor Halt indicator. When lit, this LED indicates that the emulating microprocessor has halted.

P1011

This jumper controls a break option for a write to a protected area of memory. P1011 has two positions:

Normal (2-3) YES. The emulator breaks on a write to protected memory.

Option (1-2) NO. The emulator does not break on a write to protected memory.

P2089

This jumper controls a break option for a read or write to non-existent memory. P2089 has two positions:

Normal (1-2) YES. The emulator breaks when nonexistent memory is accessed. An error message is displayed.

Option (2-3) NO. The emulator does not break when nonexistent memory is accessed.

P6102

This jumper controls wait state generation during program memory access. P6102 has three positions:

Normal (2-4) (D) During program memory access, wait states are generated only for those locations with the `al -s` option. The number of wait states is determined by P7105's position.

Option (1-2) (S) During program memory access, wait states are generated for all locations. The number of wait states is determined by P7105's position.

Option (2-3) (F) No wait states are generated during program memory access.

P7105

This jumper controls the number of wait states (2--8) generated during program memory access (when a jumper is placed across pins 2-4 or pins 1-2 of jumper P6102).

P7105 consists of 14 pins. With the component side of the circuit board facing you, and the 100-tab edge connector down, P7105 forms seven two-pin columns numbered from left to right.

Each two-pin column generates one more wait state than the corresponding column number. For example, if a jumper is placed across column 1, two wait states are generated. If a jumper is placed across column 6, seven wait states are generated.

P7105 is used when you want to duplicate user wait states in the development system. The `al` (allocate) command changes the slow memory. Then P7105 is positioned to generate the desired number of wait states.

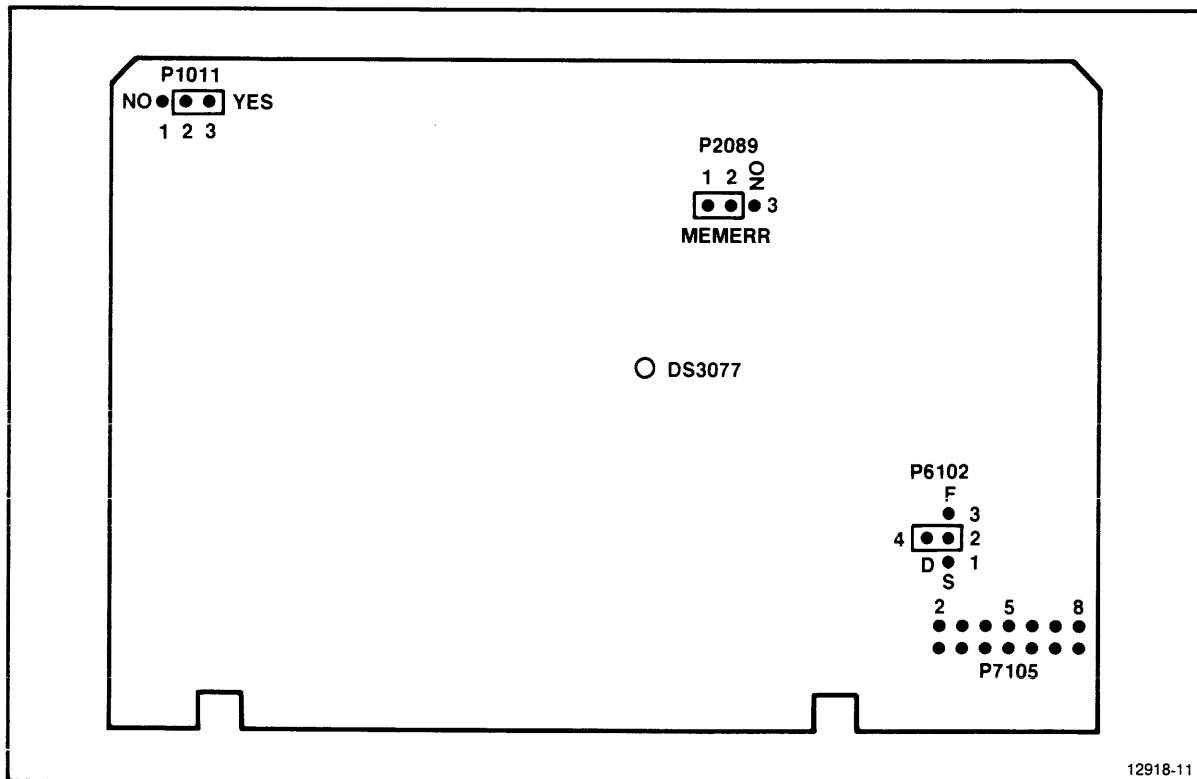


Fig. 5-2. Board II jumper and indicator locations.

PROTOTYPE CONTROL PROBE JUMPERS AND STRAPS

The following text describes each jumper and strap located in the 80186/80188 Prototype Control Probe.

NOTE

To access the jumpers in the prototype control probe, you must disassemble the probe's Interface Assembly. To disassemble the probe housing, refer to Section 6 of this manual under "Prototype Control Probe Disassembly Procedure".

CONTROL BOARD JUMPERS

The Control Board has three jumpers. The function of these jumpers is identical for both prototype control probes. Figure 5-3 shows the jumper locations and their factory settings (default positions).

P3021

This jumper controls a break option for DMA operations. P3021 has two positions.

Normal (2-3) The emulator waits indefinitely for a hold operation to finish.

Option (1-2) If a break request occurs during a hold operation, the emulator breaks after a 0.5-second delay.

P4021

This jumper controls emulator-to-prototype DMA operations during mode 3. P4021 has two positions:

Normal (2-3) User Hold to the emulator is enabled during internal operations.

Option (1-2) User Hold to the emulator is disabled during internal operations. Use this position with the dump or examine command to examine memory affected by DMA operations.

P4023

This jumper controls a path for data from program or mapped-in memory to prototype circuitry. The data path allows devices in a prototype, such as a co-processor, to follow program flow during emulation mode 1. P4023 has two positions:

Normal (1-2) The data path is disabled.

Option (2-3) The data path is enabled during emulation mode 1. To avoid bus contention, the buffers in prototype circuitry must be designed to support this option.

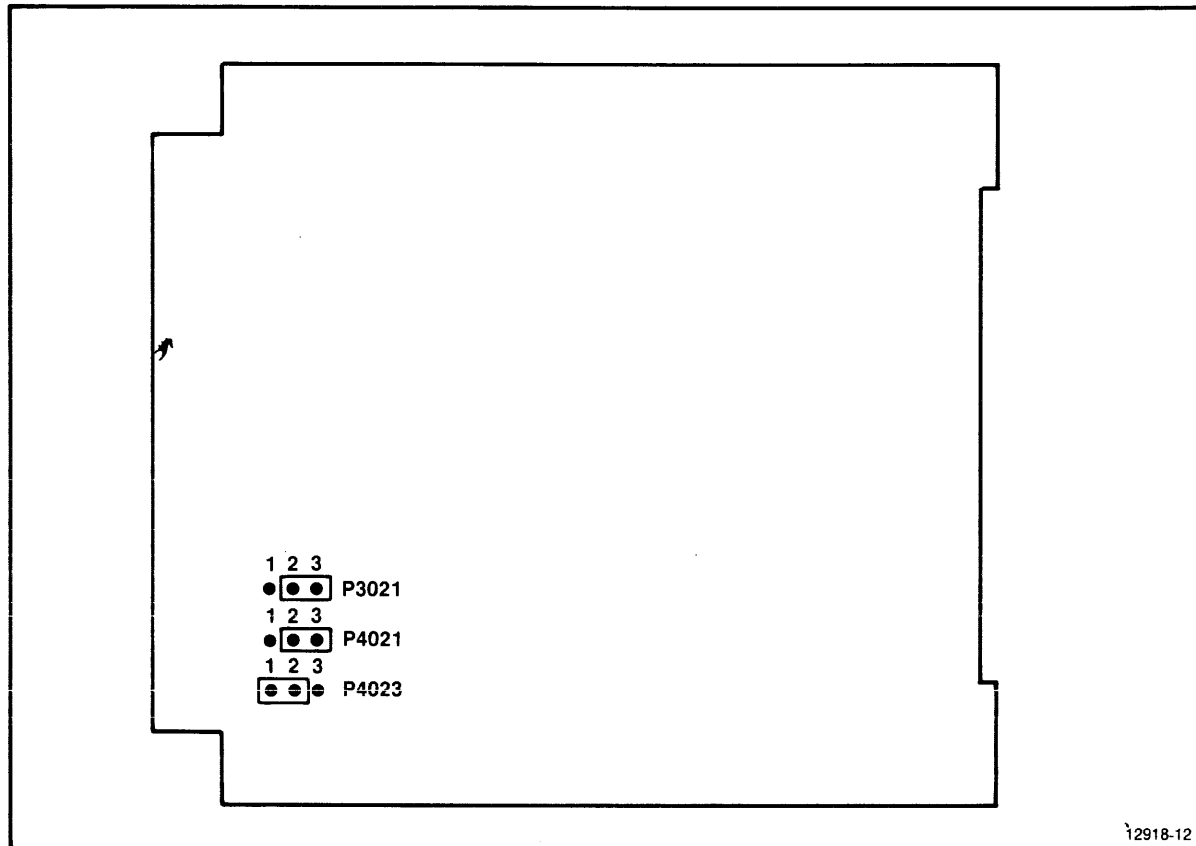


Fig. 5-3. Control Board jumper locations.

BUFFER BOARD JUMPERS

The Buffer Board has six jumpers. The function of these jumpers is identical for both prototype control probes. Figure 5-4 shows the jumper locations and their factory settings (default positions). Figure 5-4 also shows the locations of three fuses.

P4023 and P6023

These jumpers select an option that allows the prototype's SRDY and ARDY signals to control program memory access in emulation mode 1. P4023 affects SRDY control and P6023 affects ARDY control. In emulation mode 1, each program memory access causes a corresponding prototype memory access. Although data in prototype memory is not read, accessing this memory may cause the prototype to hang, if it is part of a multibus system. P4023 and P6023 have two positions:

Normal (2-3) The ARDY or SRDY control option is disabled.

Option (1-2) The ARDY or SRDY control option is enabled.

P5023

This jumper controls a READY fault break option. When selected, the option causes a break (with an error message) following a READY fault. P5023 has two positions:

Normal (1-2) The break option is enabled.

Option (2-3) The break option is disabled.

P7011

This jumper selects the number of wait states that precede a READY fault. P7011 is used with P5023 and P7023. Refer to Table 5-1 for information about the relationship between these three jumpers. P7011 has four positions:

Normal (1-2) This position selects zero wait states.

Option (3-4) This position selects one or two wait states, depending on prototype timing.

Option (5-6) This position selects four or five wait states, depending on prototype timing.

Option (7-8) This position selects eight or nine wait states, depending on prototype timing.

P7023

This jumper controls a READY fault "time-out" option that forces READY true, after a specified number of wait states (determined by P7011). P7023 has two positions:

Normal (1-2) The time-out option is disabled.

Option (2-3) The time-out option is enabled.

P8027

This jumper selects an option that can synchronize the reset signal from the prototype circuit with emulator bus cycles. When enabled, the option minimizes the likelihood of random writes to memory. P8027 has two positions:

Normal (1-2) The reset synchronization option is disabled.

Option (2-3) The reset synchronization option is enabled.

Table 5-1
Ready Fault/Timeout Jumpers

P7011	P5023	P7023	Function Produced
0 wait states	Pins (1-2)	Pins (2-3)	Fully transparent. The emulator waits indefinitely for READY to go true.
0 wait states	Pins (1-2)	Pins (1-2)	Fully transparent. The emulator waits indefinitely for READY to go true.
0 wait states	Pins (2-3)	Pins (2-3)	Fully transparent. The emulator waits indefinitely for READY to go true.
0 wait states	Pins (2-3)	Pins (1-2)	Fully transparent. The emulator waits indefinitely for READY to go true.
n wait states	Pins (1-2)	Pins (2-3)	The emulator waits indefinitely for READY to go true. However, after "n" wait-states, any other break condition (such as CTRL-C) forces READY true and generates a break. No error message is generated.
n wait states	Pins (1-2)	Pins (1-2)	After "n" wait-states, the emulator times out and forces READY true. A break is generated; an error message is displayed.
n wait states	Pins (2-3)	Pins (2-3)	After "n" wait-states, the emulator times out and forces READY true. A break is not generated; no error message is generated.
n wait states	Pins (2-3)	Pins (1-2)	After "n" wait-states, the emulator times out and forces READY true. A break is generated; an error message is displayed.

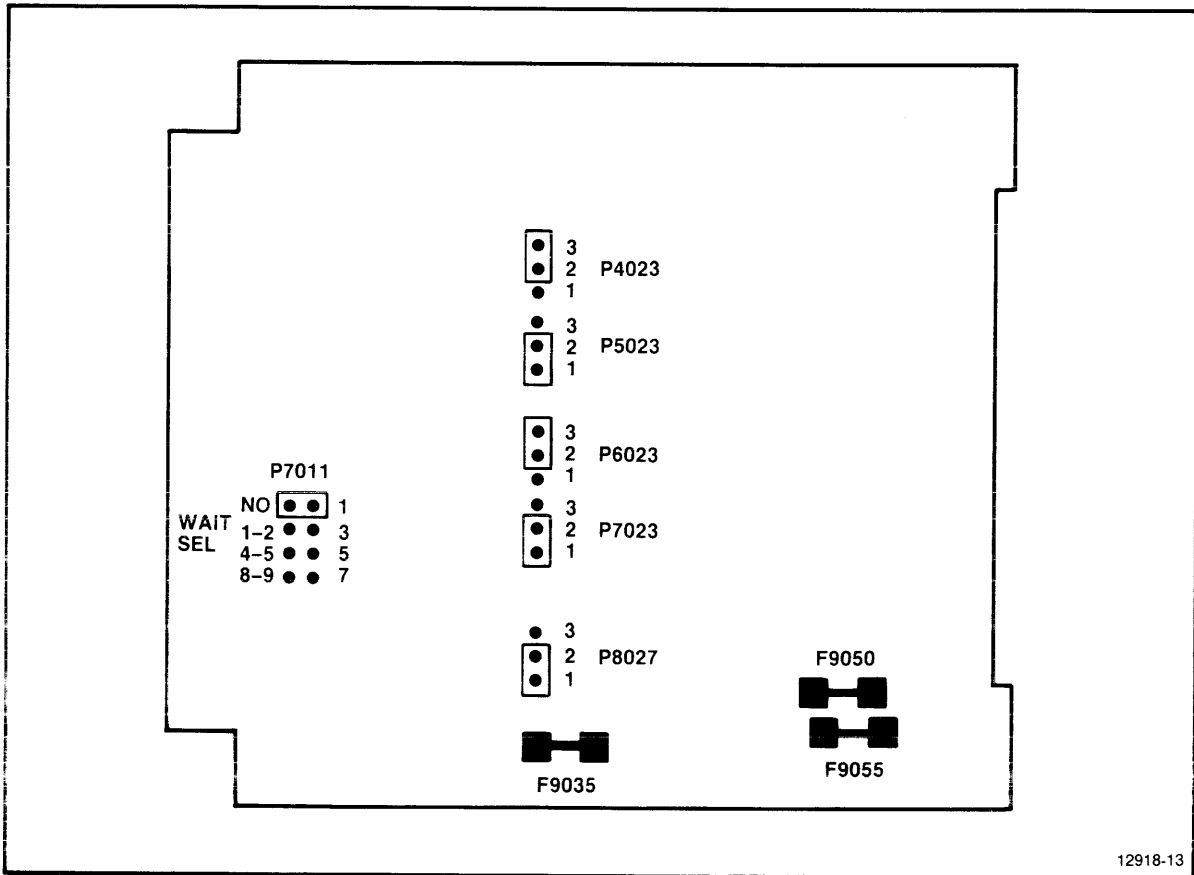


Fig. 5-4. Buffer Board jumper and fuse locations.

POWER SUPPLY BOARD JUMPER

The Power Supply Board is used in the prototype control probes of several emulators. The Power Supply Board has one jumper. Figure 5-5 shows the physical jumper location and its factory setting (default position). Figure 5-5 also shows an LED indicator DS1021. When lit, this LED indicates that the prototype control probe is shut down because of an overvoltage condition at the power supply's output.

P5061

This jumper's function is identical for both prototype control probes. P5061 has two positions:

8086/8088 (1-2) This configures the power supply for the 80186/80188 Emulator. This is the same position for the 8086/8088 Emulator.

68000 (2-3) This configures the power supply for the 68000 Emulator. Note that this is not an option. The jumper exists only because the Power Supply Board is common to the prototype control probes of several emulators.

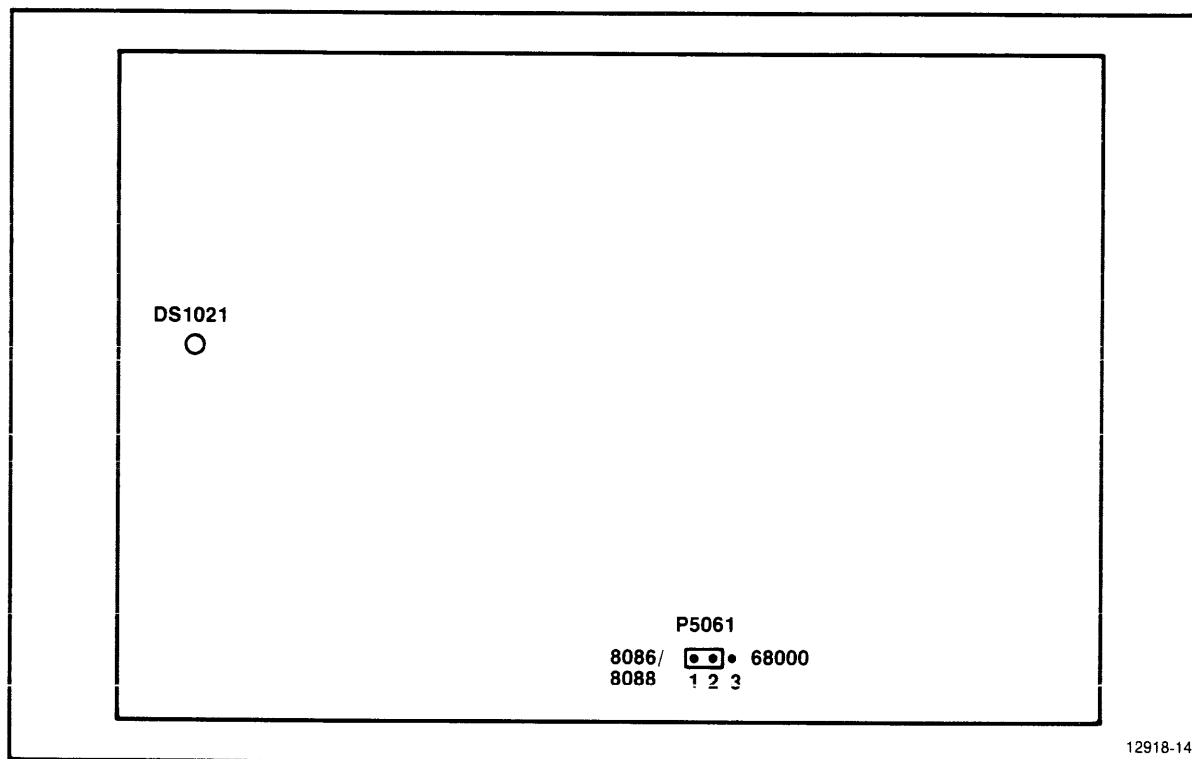


Fig. 5-5. Power Supply Board jumper and indicator locations.

CPU BOARD JUMPERS

The CPU Board has 17 jumpers. The function of these jumpers is identical for both prototype control probes. Figure 5-6 shows the jumper locations and their factory settings (default positions).

P2067

This jumper controls the emulator interrupt type. P2067 has two positions:

Normal (1-2) This position selects Nested Interrupt 3.

Option (2-3) This position selects Cascade INTRA1.

P2068

This jumper controls the emulator interrupt type. P2068 has two positions:

Normal (1-2) This position selects Nested Interrupt 2.

Option (2-3) This position selects Cascade INTRA0.

Dummy Address Jumpers

The Dummy Address Jumpers select a block of 256 "dummy" addresses to which pseudocode fetches can be directed. These addresses are required because a system access to prototype memory or I/O may cause superfluous read cycles. The dummy addresses must not include any sequential memory-mapped I/O devices or memory used for another purpose.

There are 12 dummy address jumpers (DA8---DA19). Each jumper corresponds to one of the twelve most significant address lines. Once the jumpers select a dummy address block, any superfluous read cycles are directed to addresses in that block. The individual address that is accessed will be determined by the eight least significant address lines (A0---A7).

Each dummy address position has three pins. If a jumper is placed across pins 1 and 2 (default), the address line associated with that jumper is LOW. If a jumper is placed across pins 2 and 3, the address line is held HIGH. Table 5-2 shows the address line associated with the 12 dummy address jumpers.

Table 5-2
Dummy Address Jumpers

Address Line	Jumper	Address Line	Jumper
A8	P2019	A14	P2013
A9	P2018	A15	P2012
A10	P2017	A16	P4044
A11	P2016	A17	P4043
A12	P2015	A18	P4042
A13	P2014	A19	P4041

P6042

This jumper identifies the emulating microprocessor. P6042 has two positions:

Option (1-2) This position is used for an 80186 emulating microprocessor.

Option (2-3) This position is used for an 80188 emulating microprocessor.

P6043 and P6044

These jumpers select a clock frequency of 4 or 8 MHz for the emulating microprocessor in mode 0. The following table shows the jumper positions.

Clock Frequency	Jumper Positions	
	P6043	P6044
4 MHz	2-3	1-2
8 MHz	1-2	2-3

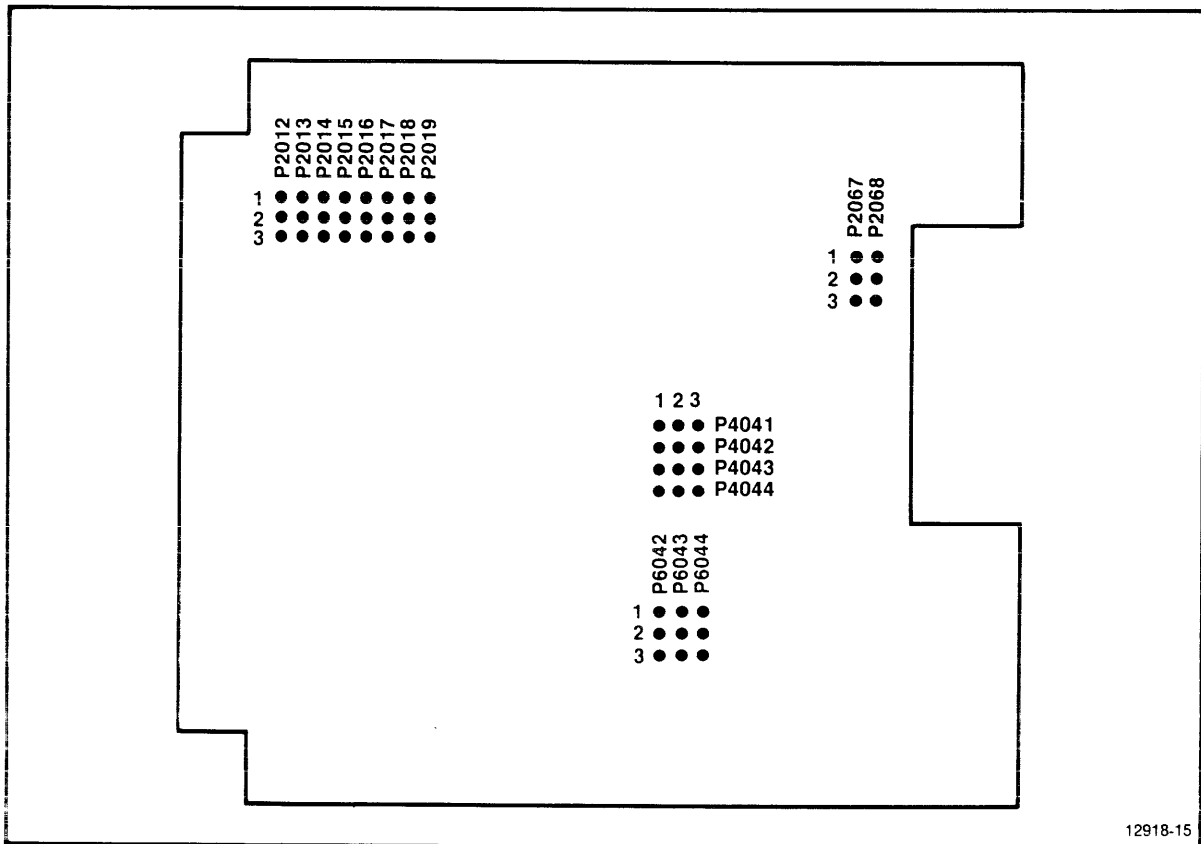


Fig. 5-6. CPU Board jumper locations.

DEVELOPMENT SYSTEM JUMPERS AND STRAPS

When the 80186/80188 Emulator is installed in an 8500 Series development system, each 32K Program Memory Board or 64/128 Program Memory Board in the development system must be configured to support emulator operation. This requires that several jumpers be checked for proper positioning.

NOTE

To access jumpers on the memory boards, you must remove the top cover from the development system and take the memory boards out of the card cage. Disassembly procedures are contained in Section 6 of this manual under "Installing the Emulator Boards and Prototype Control Probe". Complete steps 1--3 of these procedures.

32K PROGRAM MEMORY BOARD JUMPER ADJUSTMENTS

The following text tells you how to modify the development system's 32K Program Memory Board to provide 80186/80188 Emulator support. Refer to your development system's installation manual for information on the location and function of these jumpers and switch.

- W5011 Delayed read strap. Cut the ECB run between pins 1 and 2, then solder a strap between pins 2 and 3.
- J6175 Low/High board jumper. If there is only one 32K memory board in the development system's Program Section, place the jumper block across pins 2 and 3 of J6175. If there are two 32K memory boards, place the jumper blocks across pins 1 and 2 on one board and across pins 2 and 3 on the other board.
- J7171 Extended bank jumper. This jumper block must be placed across pins 1 and 2.
- S7170 Extended memory DIP switch. All switches on this DIP switch must be set to 0 (the ON position).

64K/128K PROGRAM MEMORY BOARD JUMPER ADJUSTMENTS

The following text tells you how to set two jumpers on a 64K Program Memory Board to provide 80186/80188 Emulator support. Refer to your development system's installation manual for information on the location and function of these jumpers.

- J7090 Read delay jumper. This jumper block must be set to the 0 ns position (no delay).
- J8100 through J8160 Address select jumpers. These jumpers must be set to select continuous memory from 0 to 128K. This is done by setting jumper blocks of group A to 0, of group B to 32K, of group C to 64K, and of group D to 96K. Refer to the 64K/128K Static Program Memory Installation Manual for more information on setting these jumpers.

80186/80188 AND DEVELOPMENT SYSTEM JUMPER SUMMARY

Table 5-3 is a jumper summary for the 80186/80188 Emulator showing the jumper default positions. Table 5-4 is a jumper summary for the memory boards in the development system.

Table 5-3
80186/80188 Jumper Default Position Summary

Circuit Board	Jumper	Default position
Board 1	P1086	Jumper across pins 2--3
Board II	P1011	Jumper across pins 2--3
Board II	P2089	Jumper across pins 1--2
Board II	P6102	Jumper across pins 2--4
Board II	P7105	No default, refer to description
Probe Control Board	P3021	Jumper across pins 2--3
Probe Control Board	P4021	Jumper across pins 2--3
Probe Control Board	P4023	Jumper across pins 1--2
Probe Buffer Board	P4023	Jumper across pins 2--3
Probe Buffer Board	P5023	Jumper across pins 1--2
Probe Buffer Board	P6023	Jumper across pins 2--3
Probe Buffer Board	P7011	Jumper across pins 1--2
Probe Buffer Board	P7023	Jumper across pins 1--2
Probe Buffer Board	P8027	Jumper across pins 1--2
Probe CPU Board	P2067	Jumper across pins 1--2
Probe CPU Board	P2068	Jumper across pins 1--2
Probe CPU Board	P2012	Dummy address jumper, pins 1--2
Probe CPU Board	P2013	Dummy address jumper, pins 1--2
Probe CPU Board	P2014	Dummy address jumper, pins 1--2
Probe CPU Board	P2015	Dummy address jumper, pins 1--2
Probe CPU Board	P2016	Dummy address jumper, pins 1--2
Probe CPU Board	P2017	Dummy address jumper, pins 1--2
Probe CPU Board	P2018	Dummy address jumper, pins 1--2
Probe CPU Board	P2019	Dummy address jumper, pins 1--2
Probe CPU Board	P4041	Dummy address jumper, pins 1--2
Probe CPU Board	P4042	Dummy address jumper, pins 1--2
Probe CPU Board	P4043	Dummy address jumper, pins 1--2
Probe CPU Board	P4044	Dummy address jumper, pins 1--2
Probe CPU Board	P6042	No default, refer to description
Probe CPU Board	P6043	No default, refer to description
Probe CPU Board	P6044	No default, refer to description
Probe Power Supply	P5061	Jumper across pins 1--2

Table 5-4
System Jumper Default Position Summary

Circuit Board	Jumper	Default position
32K Memory Board	W5011	Cut the ECB run between pins 1 and 2, then solder a strap between pins 2 and 3.
32K Memory Board	J6175	Place jumper across pins 1--2 if you have only one board. If you have two boards, place jumper across pins 1--2 on one board and across pins 2--3 on the other board.
32K Memory Board	J7171	Place jumper across pins 1--2.
32K Memory Board	S7170	Set each position of this DIP switch to 0 (the ON position).
64K/128K Memory Board	J7090	Set to "0" ns position (no delay).
64K/128K Memory Board	J8100 through J8160	Set jumper group A to 0, group B to 32K, group C to 64K, and group D to 96K.

Section 6

INSTALLATION

INTRODUCTION

This section tells how to install the 80186/80188 Emulator boards and prototype control probe in your Tektronix development system. This section also contains information for connecting the 68-pin probe plug to your prototype circuit, and disassembly instructions for the prototype control probe. In addition, this section describes how to install the control software for the various development systems.

HARDWARE INSTALLATION

CAUTION

Before inserting or removing any circuit board, ensure that primary power to the development system is **OFF**. Inserting or removing a board while the power is **ON** may result in component damage.

Under no circumstances can any other emulator be installed in your Tektronix development system while the 80186/80188 Emulator is installed. Excessive power supply loading will result.

NOTE

The 80186/80188 Emulator and the Memory Allocation Controller Board cannot be installed in the same development system.

INSTALLING THE EMULATOR BOARDS AND PROTOTYPE CONTROL PROBE

The 80186 or 80188 Prototype Control Probe, attaches to the 80186/80188 Emulator boards by two 6-foot ribbon cables at connectors P100 and P200 on Board II. To install the 80186/80188 Emulator, perform the following procedure.

1. Verify that primary power (115 Vac or 230 Vac) to the development system is **OFF**.
2. Remove the two cover retainers at the upper rear corners of the mainframe. (Refer to Fig. 6-1.)

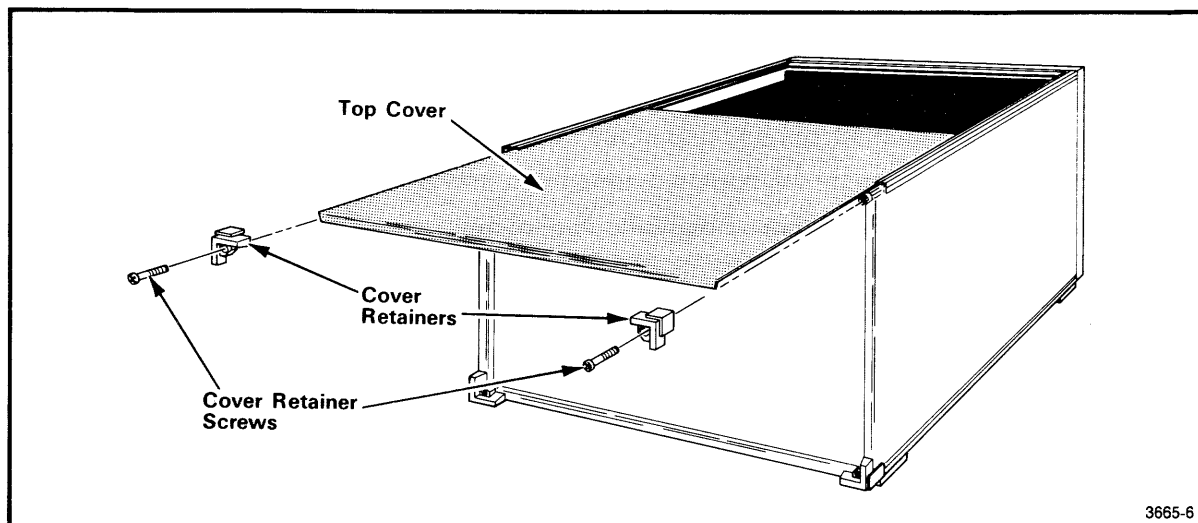


Fig. 6-1. Removal/installation of the top cover.

3. Remove the top cover by sliding it straight back, then set the cover aside.

NOTE

There are jumpers on several development system boards (32K and 64/128K Memory Boards) that must be changed when the 80186/80188 Emulator is installed. If you have these boards installed in your system, check to make sure these jumpers are correctly set before installing the 80186/80188 Emulator boards. Section 5 of this manual describes the correct settings for these jumpers.

4. Remove any emulator board or Memory Allocation Controller Board that is installed in the development system's card cage. Refer to the Caution and Note paragraphs at the beginning of this section.
5. Connect Boards I, II, and III together, using the six 50-conductor ribbon cables provided. Dress the cables as illustrated in Fig. 6-2.

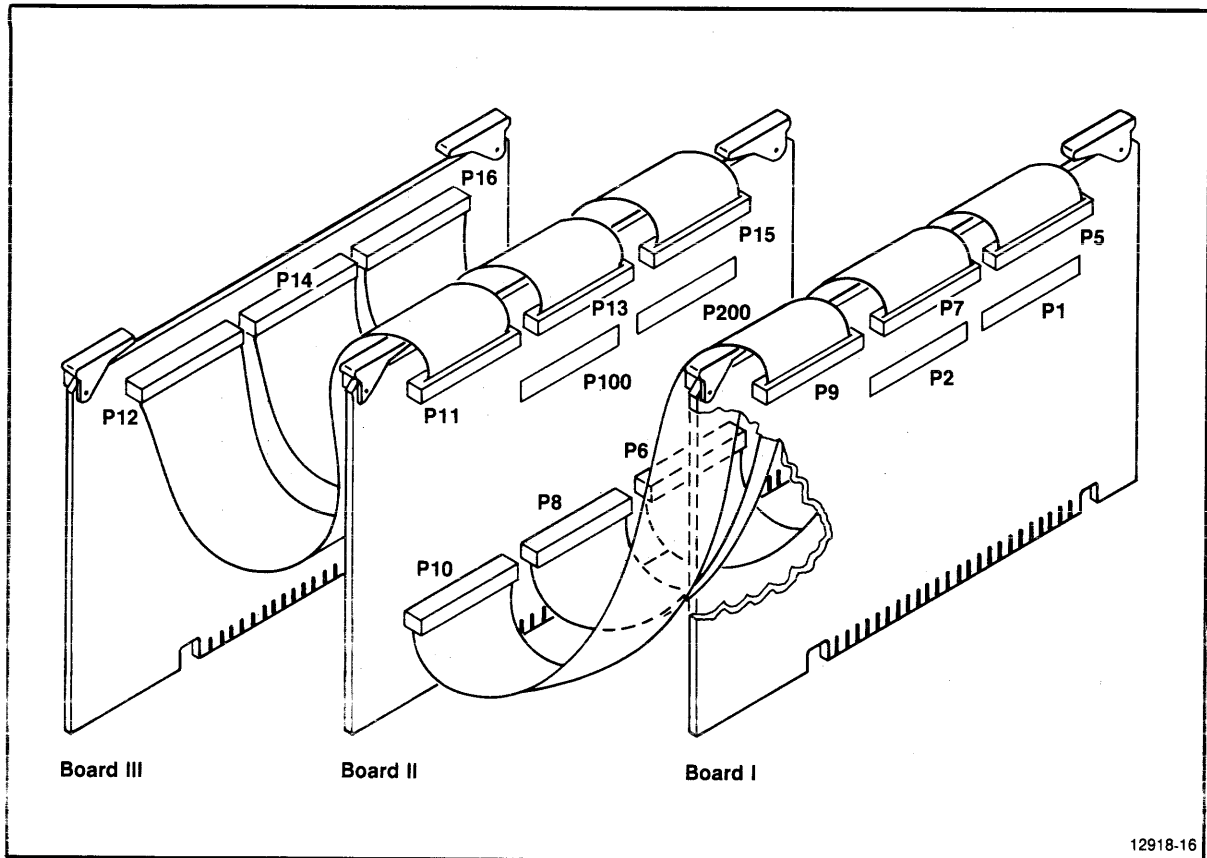


Fig. 6-2. 80186/80188 Emulator board interconnections.

6. Face the front of the mainframe and hold the three emulator boards by their upper side edges. With the component sides facing left, align the boards with other circuit boards in the development system's card cage.
7. While holding all three boards, guide Board I into the vertical channel to J14 on the Main Interconnect Board. Next, guide Board II into the vertical channel to J15, then guide Board III into the vertical channel to J16. Refer to Fig. 6-3 (if your development system is an 8301 MDU) or Fig. 6-4 (if your system is an 8540 IU).

NOTE

When the emulator boards reach their connectors on the Main Interconnect Board, do not snap them into place yet. You may need to lift the boards slightly when performing later steps in this procedure.

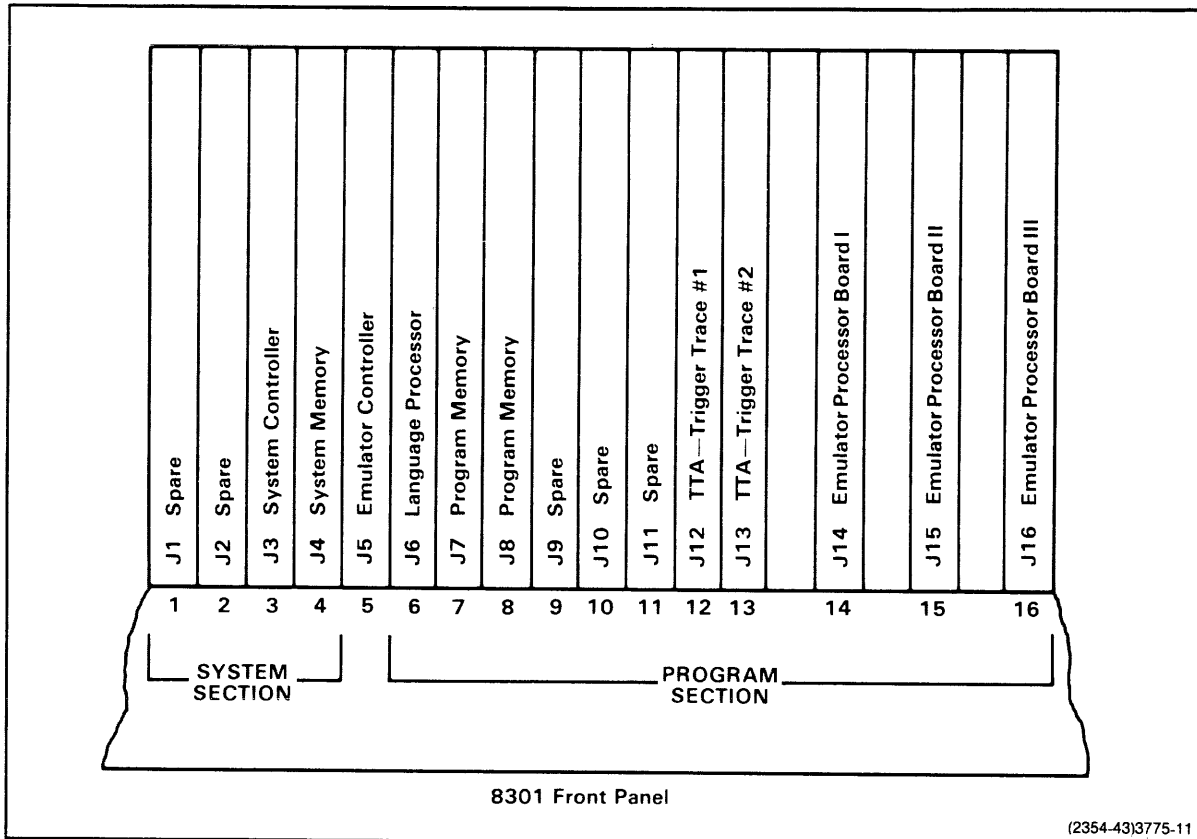


Fig. 6-3. Recommended board arrangement for the 8301.

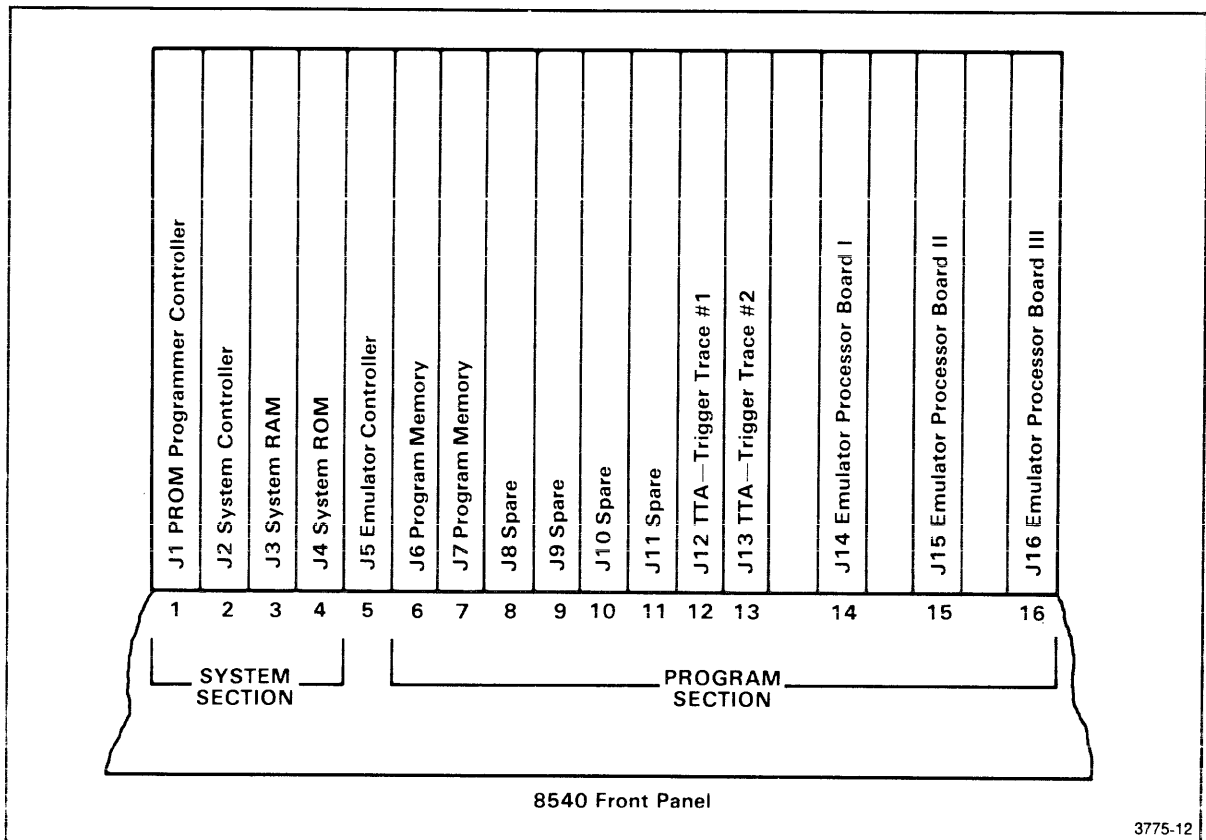


Fig. 6-4. Recommended board arrangement for the 8540.

8. Remove the two mounting screws at the top and bottom of the strain relief plate, as shown in Fig. 6-5. Then remove the strain relief/cable clamp assembly from the rear panel.

NOTE

The standard cable clamps provided with your development system are not compatible with the 80186/80188 Prototype Control Probe ribbon cables. The standard clamps must be replaced with the clamps provided with your 80186 or 80188 Prototype Control Probe.

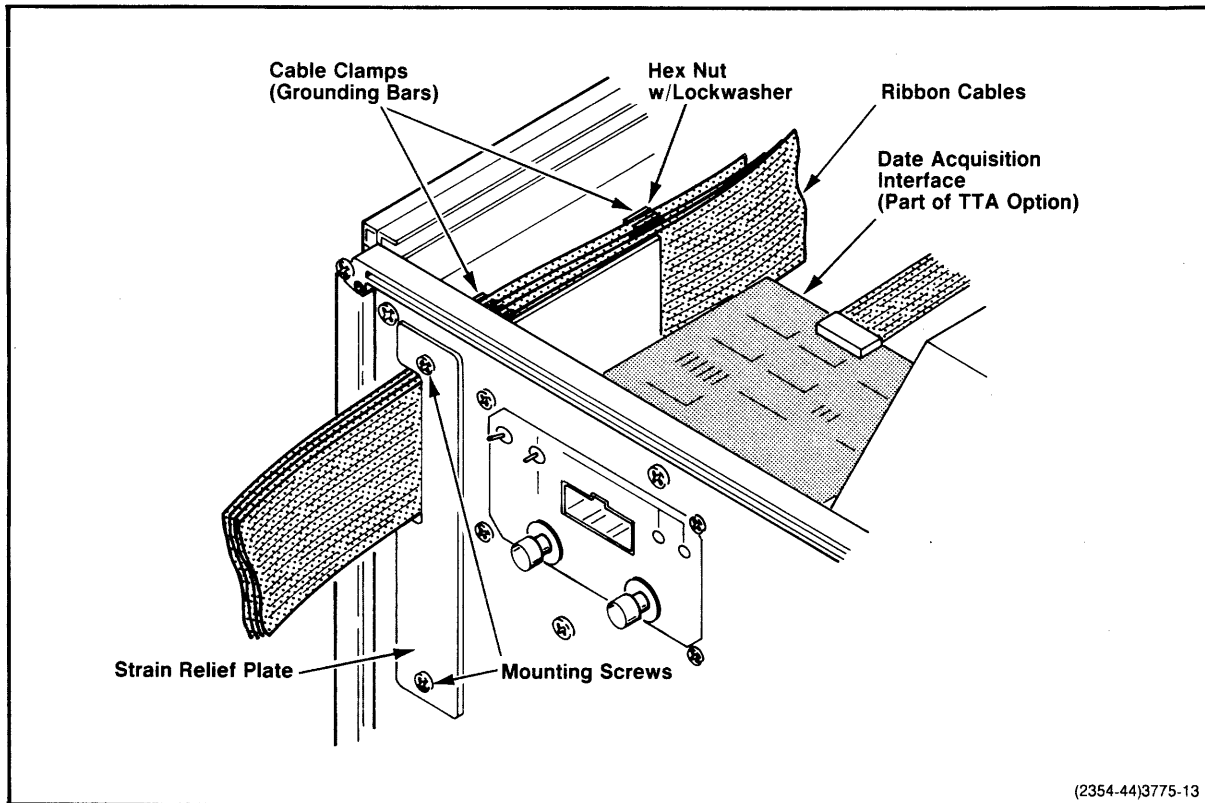


Fig. 6-5. Strain relief plate installation/removal.

9. Mount the two 6-foot prototype control probe ribbon cables to the cable clamp assembly using the new clamps provided. Secure the cable clamps with hex nuts and lock washers. Refer to Fig. 6-6. Allow enough ribbon cable to reach connectors P100 and P200 on Board II in J15 of the Main Interconnect Board in the development system's card cage.
10. Feed the prototype control probe cable connectors (J100 and J200) through the cableway opening. Then guide the strain relief/cable clamp assembly into the cableway opening. Attach the strain relief plate to the rear panel using the two screws removed in step 8.

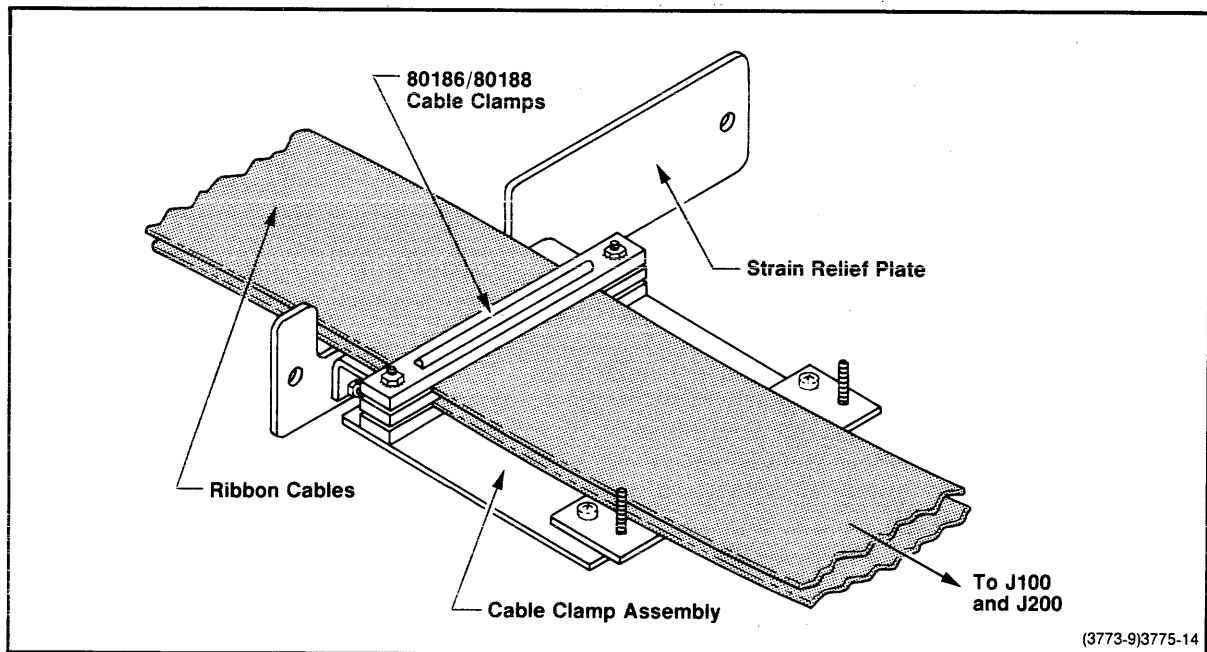


Fig. 6-6. Ribbon cable installation.

11. Attach the two ribbon cable connectors labeled J100 and J200 to P100 and P200, respectively. Refer to Fig. 6-7. Be sure that pin 1 of each cable connector (blue stripe) and pin 1 of each socket are aligned. Pin 1 of P100 and P200 is to the left when viewed from the component side of the board.

NOTE

If you are installing your emulator on an 8540 and have not yet installed the option ROMs shipped with your emulator, refer to "Installing the 8540 Firmware," later in this section.

If you intend to install, or have already installed, the optional Trigger Trace Analyzer in your development system, refer to "Connecting to the Trigger Trace Analyzer (Optional)," later in this section. Complete the connections to the Trigger Trace Analyzer before proceeding.

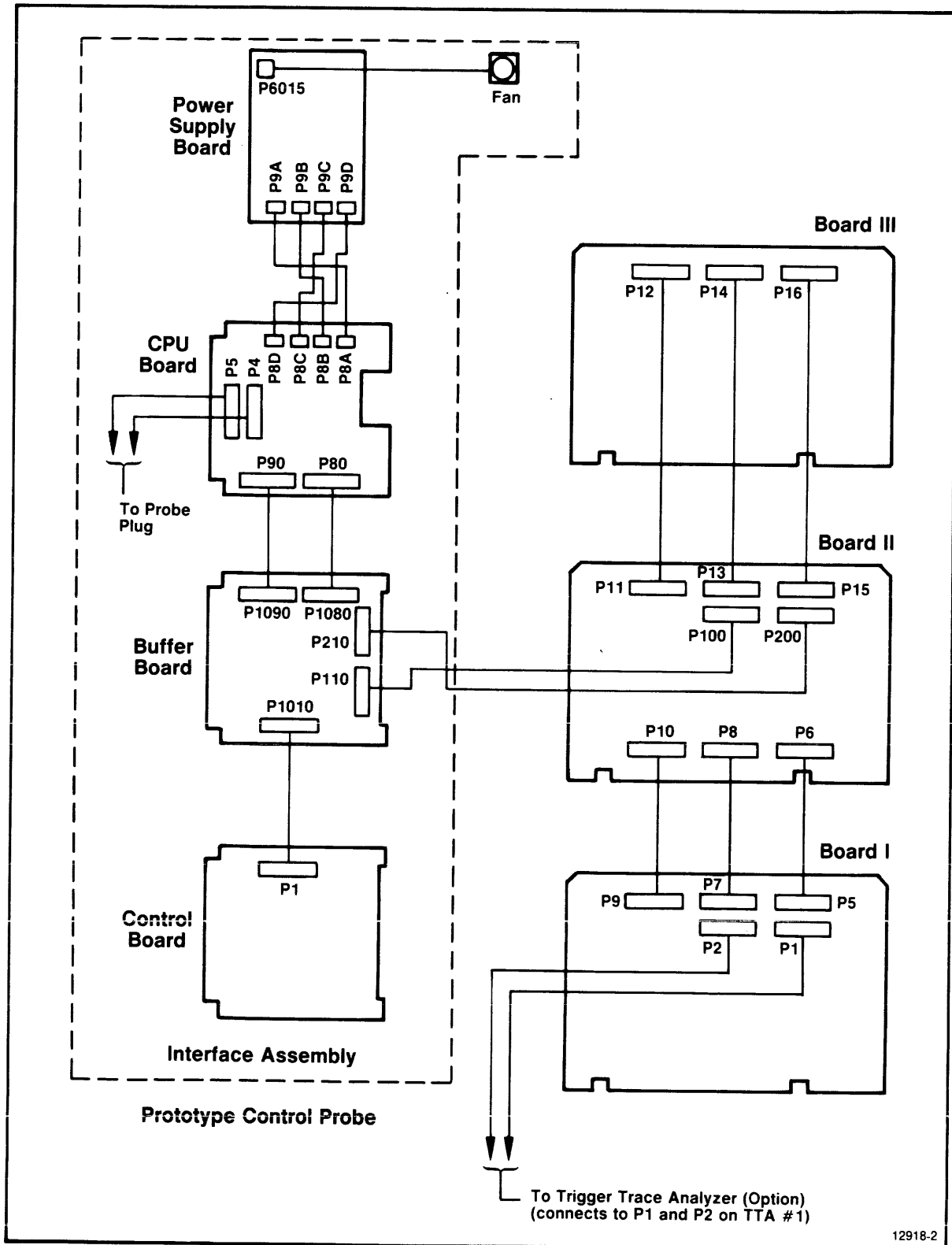


Fig. 6-7. Emulator boards with prototype control probe.

12. Slide the three emulator boards down until they reach their respective connectors on the Main Interconnect Board in your development system's mainframe. Then press down firmly and evenly on the top edges, one board at a time, until each board snaps into place.
13. Dress the cables along the cableway at the side of the mainframe and over the side of the card cage. Make sure the cables are dressed to lie flat allowing clearance for the top cover.
14. Slide the top cover back into the guide tracks at the top of the mainframe. Be sure the cover is properly seated in the slot at the front of the mainframe.
15. Install the cover retainers at the upper corners on the rear of the mainframe. (See Fig. 6-1.) Tighten the cover retainer screws securely.

CONNECTING TO THE PROTOTYPE

CAUTION

Static discharge can damage the prototype control probe when the probe plug is not installed in a 68-pin Textool socket. When handling the probe plug, hold the probe plug by the case only and do not touch the pins. The probe plug should be stored in conductive foam when not in use.

The 68-pin JEDEC Type A probe plug at the end of the four 1-foot flexible cables fits into a 3M Textool 68-pin Chip Carrier Socket in the prototype hardware. Pin 1 on the plug must be mated with the corresponding pin 1 in the socket. A mark is located near pin 1 on both the plug and the socket to aid in pin identification. Refer to Fig. 6-8.

CAUTION

If the prototype control probe plug is incorrectly inserted in the prototype socket, damage to the prototype control probe or to the prototype may result. Refer to Figure 6-8 and the following procedure to ensure proper plug insertion.

Probe Plug Insertion Procedure

To ensure that the probe plug is properly inserted into the 3M Textool socket adhere to the following procedures.

NOTE

Three of the probe plug's corners contain square notches that are used as indexes to ensure correct plug alignment. The fourth corner (Pin No. 1 Mark) is cut on a 45 degree bias. (Refer to Fig. 6-8.) The 3M Textool socket has three guide boss pins that mate with the square notches and a spring lever that mates with the corner marked as Pin No. 1.

1. Before inserting the probe plug, inspect the socket to verify that all the contact fingers are in their proper position. If the socket is bent, damaged, or has contaminated contact fingers, the socket should be replaced with a new socket before installing the probe.
2. Move the wire bail on the socket to its open position. Ensure that the slide fasteners on the probe plug are in their retracted positions. Fig. 6-8 shows the slide fasteners in their retracted positions.
3. Lower the probe plug (perpendicular to the plane of the socket) carefully into the socket. Do not slide or twist the plug into the socket. These movements may damage or bend the contact fingers. The two stationary tabs on the plug should be toward the wire bail on the socket.
4. Move the probe plug around very gently to verify that the square notches on the plug are aligned with the guide boss pins in the socket. When the plug is properly seated (not hung up on one or more of the guide boss pins), push the probe plug back toward the corner opposite the corner marked Pin No. 1. This ensures that the contact pads are aligned properly.
5. With the probe plug in this position, press down on the plug case until the stationary tabs are low enough to pull the wire bail over them, holding that end of the plug in place.
6. While still maintaining pressure on the plug, move the slide fasteners on each side of the plug to their fully extended position so that the tabs on the fasteners fit into the holes on the raised back corners of the socket. When the wire bail and slide fasteners are in place, remove pressure from the plug case.

CAUTION

Be very careful when handling the flexible cable attached to the probe plug. If the cable is creased or nicked at the edges, excessive pulling on the cable could cause a tear. Tears can propagate rapidly through the circuit runs, causing open and shorted circuits.

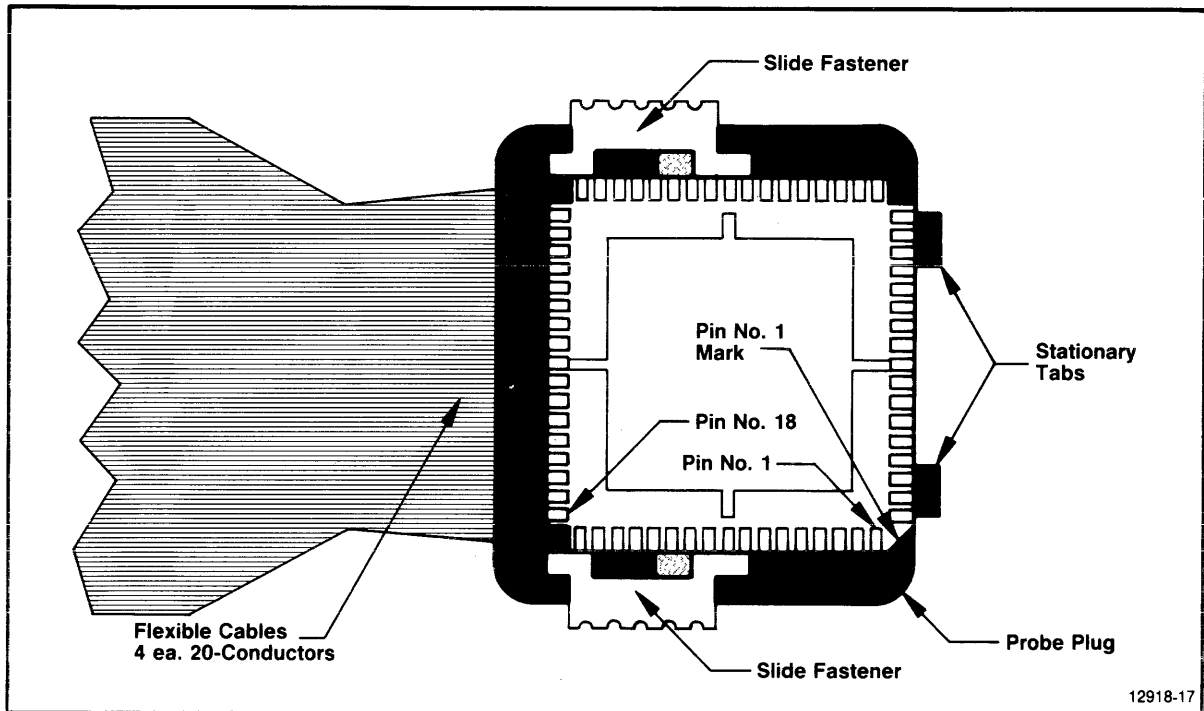


Fig. 6-8. Pin identification and proper plug insertion.

PROTOTYPE CONTROL PROBE DISASSEMBLY PROCEDURE

The following procedure tells you how to disassemble the prototype control probe. This procedure should be followed whenever changing the jumpers on any of the boards or calibrating the power supplies. The prototype control probe consists of an Interface Assembly which houses the four circuit boards and fan. Two 6-foot, 40-conductor ribbon cables (connecting to P110 and P210 on the Buffer Board) are the interfacing cables to the emulator boards. Four 1-foot, 20-conductor flexible cables connecting to P4 and P5 on the CPU Board are the interfacing cables to the probe plug.

The Interface Assembly housing consists of a top cover, bottom cover, and spacer ring (the spacer ring forms the sides of the housing). The disassembly procedures are presented in three parts depending on how far you want to disassemble the probe:

- Interface Assembly Disassembly
- Top Cover Disassembly
- Bottom Cover Disassembly

Interface Assembly and Disassembly Procedure

1. Ensure that the primary power (115 Vac or 230 Vac) to the development system is OFF.
2. Place the prototype control probe, top cover down, on a flat nonconductive working surface with the flexible cables and probe plug facing you.
3. Remove the four screws from the bottom cover of the Interface Assembly's housing (one screw in each corner of the bottom cover).
4. Grasp the Interface Assembly's housing with both hands (holding both top and bottom of the housing together) and turn the assembly onto its bottom.
5. Lift up carefully on the top cover and spacer ring, while rotating the cover and ring to the left (counterclockwise) 180 degrees. Set the top cover and spacer ring beside the bottom cover.

NOTE

The CPU Board and Power Supply Board are attached to the top cover. The Buffer Board and Control Board are attached to the bottom cover. With the top and bottom covers laying side by side, the CPU Board and the Buffer Board are exposed, and the jumpers for both boards are readily accessible. If you want to calibrate the power supplies, follow the procedure for "Top Cover Disassembly Procedure." If you want to change the jumpers on the Control Board, follow the procedure for "Bottom Cover Disassembly Procedure."

Top Cover Disassembly Procedure

1. Using a 1/4-inch nut driver, remove the six nuts and lock washers attaching the CPU Board to the spacer studs.

NOTE

The following procedure may require more than one person to prevent damage to the circuit boards and interconnecting cables.

2. Lift the CPU Board carefully off the spacer studs, lifting the bottom cover at the same time. Rotate both the CPU Board and bottom cover to the left (counterclockwise) 180 degrees and lay them beside the top cover. (The CPU Board is left of the top cover and the bottom cover is beside the CPU Board.)

NOTE

The Buffer Board (attached to the bottom cover) is attached to the CPU Board with two ribbon cables. The Power Supply Board is attached to the CPU Board with four ribbon cables. Take care not to disconnect or damage these cables when moving the bottom cover and the CPU Board.

3. Remove the six spacer studs from the shield covering. Remove the metal shield and set it aside.

The Power Supply Board is now accessible and the power supplies may be calibrated. Refer to "Probe Power Supply Calibration" in Section 4 of this manual.

Bottom Cover Disassembly Procedure

NOTE

This procedure assumes that the top and bottom covers are laying side by side.

1. Using a 1/4-inch nut driver, remove the six nuts and lock washers attaching the Buffer Board to the spacer studs.
2. Remove the ribbon cable connector P1010 from the Buffer Board.
3. Lift the Buffer Board carefully from the spacer studs. While lifting the Buffer Board, at the same time move the bottom cover to the right. Lay the Buffer Board between the top and bottom covers.
4. Using a 1/4 inch nut driver, remove the six spacer studs from the back side of the Control Board.
5. Remove the Control Board board and turn its component side upwards. The jumpers are now accessible.

6. Reassemble the Interface Assembly in reverse order of disassembly.

CONNECTING TO THE TRIGGER TRACE ANALYZER (OPTIONAL)

The following procedure explains how to connect the 80186/80188 Emulator to the optional Trigger Trace Analyzer (TTA). Refer to the Trigger Trace Analyzer Installation Manual for further information.

Figures 6-7 and 6-9 shows the connections between the TTA and the 80186/80188 Emulator. To install the TTA with the emulator, perform the following procedure:

1. Install the 80186/80188 Emulator in your development system using the procedures described earlier in this section. Don't replace the cover of the development system until this entire procedure is completed.
2. On the back side of TTA Board No. 1, attach one of the emulator interconnecting cables to P1 and the other cable to P2.
3. Install the TTA, using the procedures described in the Trigger Trace Analyzer Installation Manual.
4. Attach the two interconnect cables from the TTA to P1 and P2 of Emulator Board I, respectively.
5. Replace the top cover of the development system using the procedures described earlier in this section.

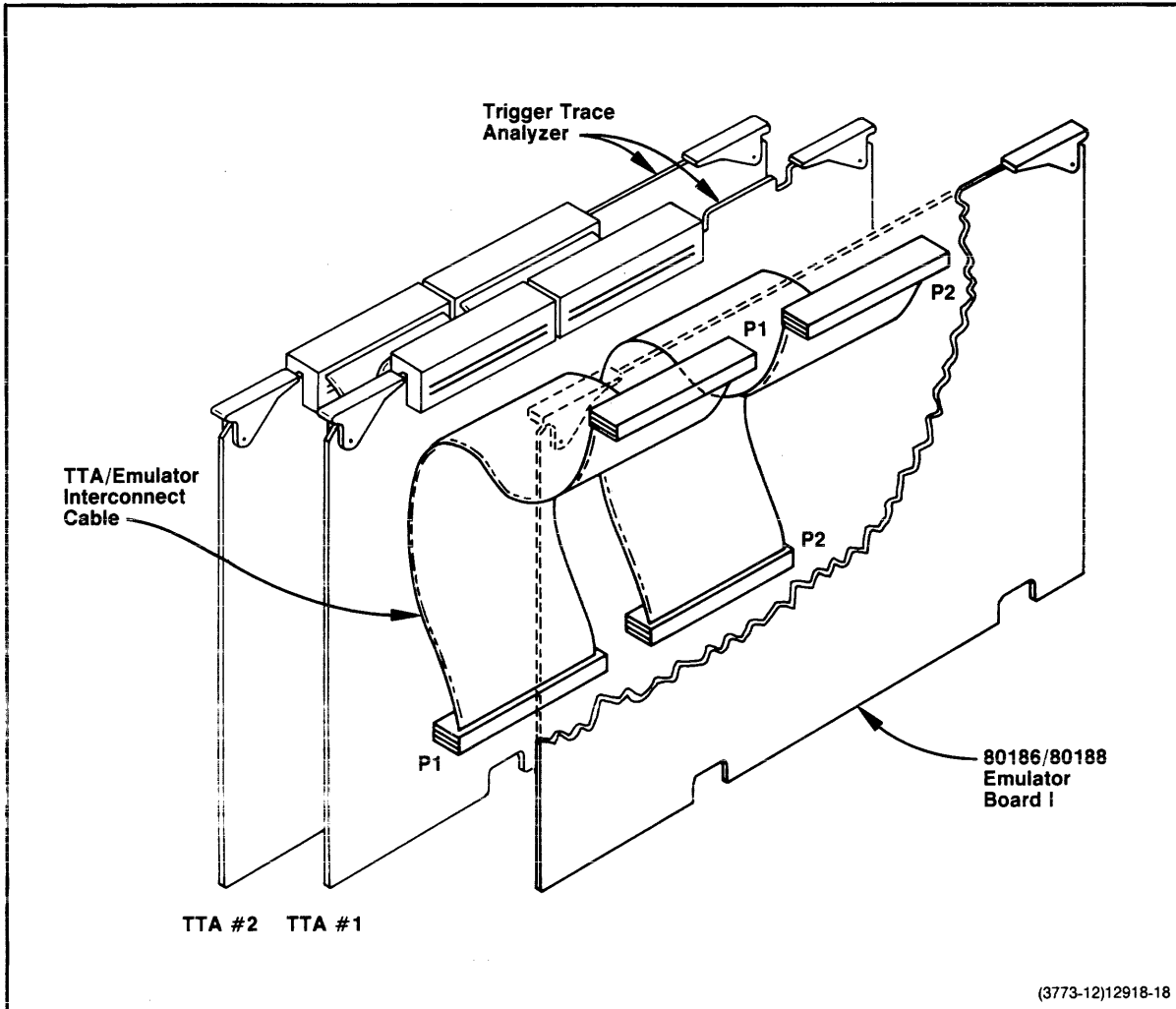


Fig. 6-9. 80186/80188 Emulator Board I connections to the TTA.

GROUNDING

A proper ground system is necessary for the satisfactory operation of the 8301 or the 8540. The emulator boards and prototype control probe, as well as any other optional and peripheral equipment, must be properly grounded to eliminate ground loops and prevent static discharge damage. Ensure that primary power cords of all units, including the prototype circuitry, are connected to outlets on the same ground system.

Refer to your development system's Installation Guide for additional grounding procedures.

SOFTWARE INSTALLATION

Software installation procedures are provided in the following pages. These procedures consist of:

- Setting up the 8560 Multi-User Software Development Unit
- Installing the firmware for an 8540 Integration Unit
- Installing the software for an 8550 Microcomputer Development Lab

SETTING UP THE 8560

The following text describes procedures that you must perform if you are using an 8560 with your 8540 or 8550.

Setting Your Shell Variable

If you use your 8540 or 8550 in TERM mode with an 8560, and want to assemble 80186/80188 code, you must set your shell variable to the assembler you are going to use. To use the 80186/80188 Assembler (ASM80186 version x.xx), you must set your shell variable as follows every time you login:

```
$ uP=80186; export uP
```

INSTALLING THE 8540 FIRMWARE

The firmware shipped with your emulator contains emulator control and diagnostic ROMs. These ROMs are installed in your 8540's System ROM Board. The firmware package consists of one type-2764 ROM and four type-27128 ROMs:

- One type-2764 ROM labeled "BASE DIAG 1" is a replacement for one of the diagnostic ROMs presently installed in your 8540's System ROM Board.
- Three type-27128 ROMs labeled "80186/188-1", "80186/188-2", and "80186/188-3" contain the emulator control firmware.
- One type-27128 ROM labeled "80186/188 DIAGNOSTICS" contains the emulator diagnostics.

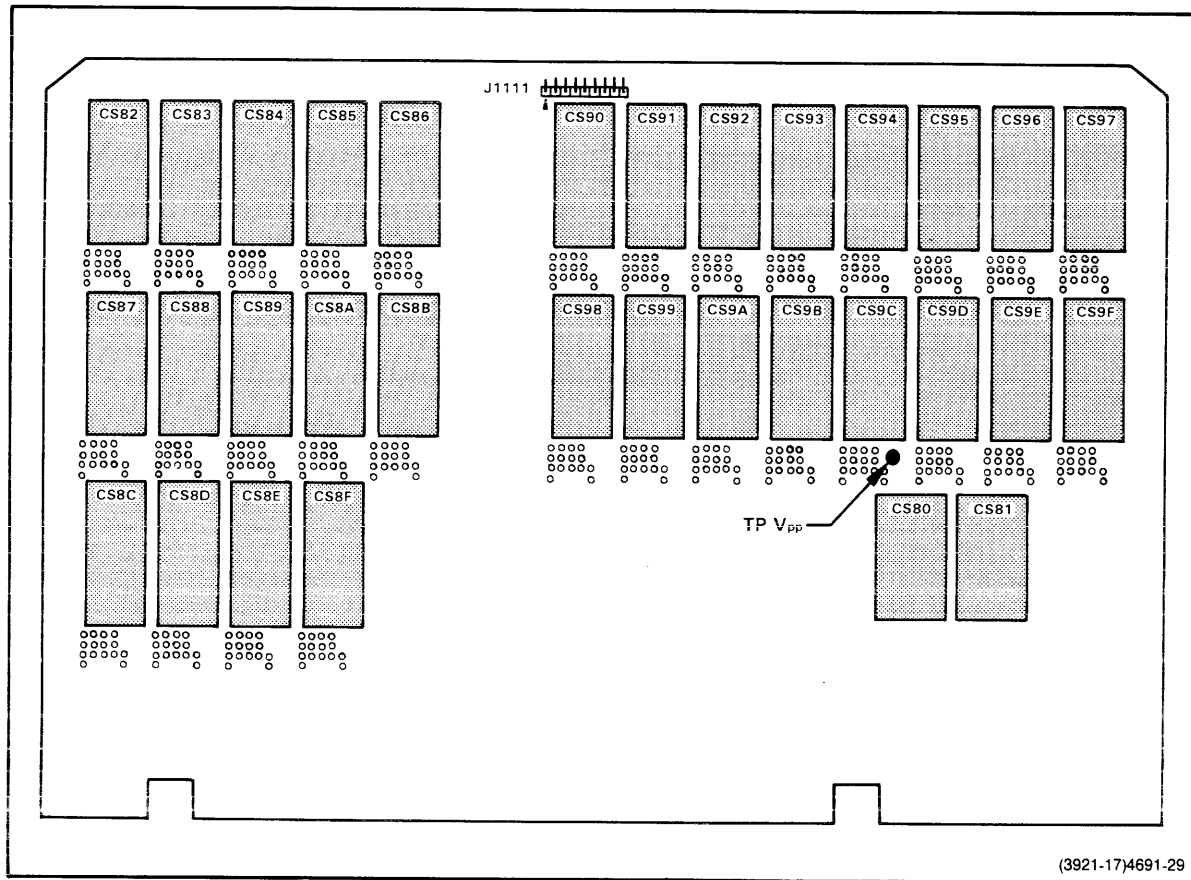


Fig. 6-10. System ROM Board socket locations.

The following procedure explains how to perform this installation.

1. Remove the 8540's top cover by completing steps 1 through 3 described earlier in this section under "Installing the Emulator Boards and Prototype Control Probe."
2. Remove the System ROM Board from your system's Main Interconnect Board and place the board on a flat static-free work area.
3. Check the part number on the ROM labeled "BASE DIAG 1" installed in your System ROM Board. BASE DIAG 1 is normally installed in the socket labeled CS94 shown in Fig. 6-10. The socket circuit designator is U1160. If the BASE DIAG 1 part number (160-1378-xx) is lower than the part number on the BASE DIAG 1 ROM shipped with your emulator, then replace the installed ROM with the new ROM. (Install the BASE DIAG 1 ROM with the highest extension number.)

4. The four type-27128 ROMs shipped with your emulator may be installed in any available sockets you choose. However, it is recommended that emulator option ROMs be installed in any of the ten sockets labeled CS82--CS8B shown in Fig. 6-10. (The socket circuit designators are U1010, U1030, U1050, U1060, U1070, U3010, U3030, U3050, U3060, and U3070).

NOTE

The spare ROM sockets for options are limited. It may be necessary to remove unneeded ROMs to install the required 80186/80188 Emulator ROMs. (No other emulator can be installed in the 8540 with the 80186/80188 Emulator.)

NOTE

The 8540's System ROM Board was originally shipped from the factory with the ROM sockets configured for only type-2764 ROMs. In recent shipments of the System ROM Board, the ROM sockets are configured for either type-2746 or type-27128 ROMs. See Fig. 6-11. If your 8540's System ROM Board is not configured for type-27128 ROMs, four 0-ohm resistors are provided with the 80186/80188 Emulator to make this modification to the ROM sockets before the ROMs are installed.

5. Solder a 0-ohm resistor (provided with your emulator) across the lower strap positions of each socket that will contain the type-27128 ROMs, if required. Figure 6-11 shows the position of the 0-ohm resistor. Insert the four type-27128 ROMs into these sockets. After a socket is modified, either a type-2764 or a type-27128 ROM can be installed in the socket.

NOTE

If the 0-ohm resistor is missing or damaged, you may substitute a solid wire strap.

6. Install other optional ROMs on the System ROM Board as follows:
 - Install the PROM Programmer option ROMs in locations CS90 and CS91 shown in Fig. 6-10. The socket circuit designators are U1110 and U1120.
 - Install the Communications option (COMM Option) ROM in location CS8F shown in Fig. 6-10. The socket circuit designator is U4060.
 - Install the Trigger Trace Analyzer option ROMs in locations CS8E, CS8D, and CS8C shown in Fig. 6-10. The socket circuit designators are U4050, U4030, and U4010.

7. Reinstall the System ROM Board in the mainframe exactly as it was before. Replace the top cover by completing steps 14 and 15 as described earlier in this section under "Installing the Emulator Boards and Prototype Control Probe."

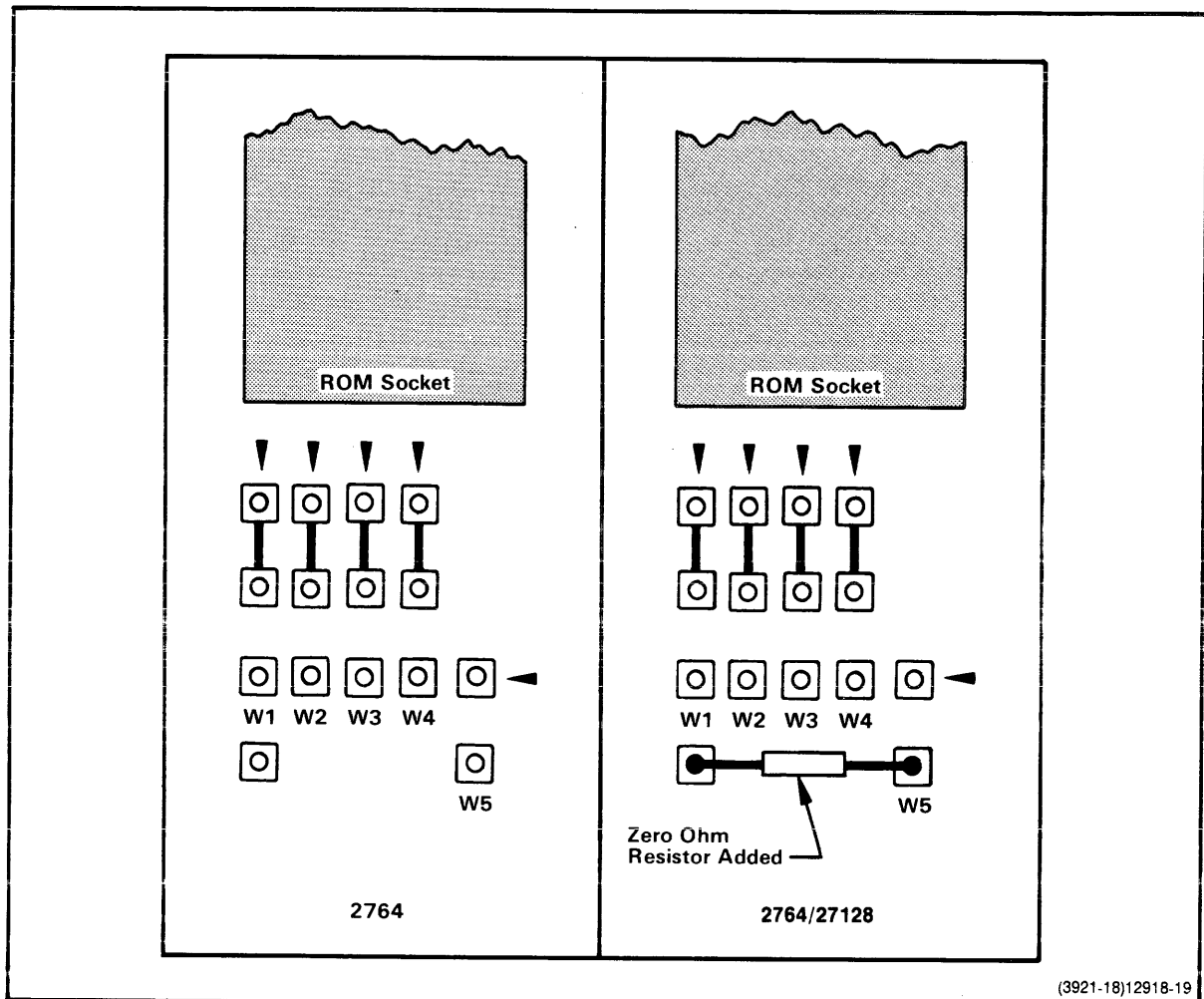


Fig. 6-11. Type-2764 and type-27128 ROM strapping.

INSTALLING THE 8550 SOFTWARE

The following text explains how to install the flexible disks shipped with your emulator. This discussion also explains how to use the 8086/8088 Assembler to assemble 8086 and 80186 code providing the code does not contain instructions unique to the 80186 or 80188 microprocessors.

NOTE

An 80186/80188 Assembler is not available for the 8550 development system. If you have an 8086/8088 Assembler for the 8550, the control software provided with your 8086/8088 Emulator can be installed onto your DOS/50 system disk. Installing this software permits you to assemble and load an 8086 program or 80186 program providing the code does not contain the unique instructions for the 80186 or 80188 microprocessors. The 8086/8088 Assembler is not compatible with instructions that are unique to 80186 or 80188 microprocessors.

Your emulator software package for 8550 installation consists of two disks:

- A disk containing emulator control software, which you install onto your DOS/50 system disk so that DOS/50 can control your emulator hardware.
- An 8550 System Diagnostic Disk, which contains the latest version of your development system's diagnostic program.

The following paragraphs describe how to install the control software for your 80186/80188 Emulator.

To complete this installation procedure you need the following items:

- An 8550 system (with or without an 80186/80188 Emulator)
- A DOS/50 system disk with a write-enable tab over the write-protect slot
- An 80186/80188 Emulator Software Installation Disk without a write-enable tab

Start Up and Set the Date

Turn on your 8550 system. (For start-up instructions, refer to the Learning Guide of your System Users Manual.) Place your system disk in drive 0 and shut the drive 0 door. When you see the > prompt on your system terminal, place your installation disk in drive 1 and shut the drive 1 door.

Use the **dat** command to set the date and time. For example, if it is 11:05 a.m. on October 12, 1984, type:

```
> dat 12-oct-84/11:05
```

The system uses this information when it sets the CREATION time attribute of each file copied from your installation disk.

Installation Procedure

The command file INSTALL2, which installs the emulator control software, resides on the installation disk. To execute the command file, simply type its filespec:

```
> /VOL/EMU.80186/INSTALL2
```

DOS/50 responds with the following message:

```
* During this installation procedure, one or more of the
* following messages may appear.  IGNORE THESE MESSAGES:
*
*      Error 6E - Directory alteration invalid
*      Error 7E - Error in command execution
*      Error 1D - File not found
*
* If any OTHER error message appears, see your
* Users Manual for further instructions.
*
* If no other error message appears, you'll receive a
* message when the installation procedure is complete.
*
T,OFF
```

In this installation procedure, you may disregard error messages 6E, 7E, and 1D; these messages have no bearing on the success of the installation. However, if a message other than 6E, 7E, or 1D appears, take the following steps:

1. Make sure you are using the right disks.
2. Make sure your system disk has a write-enable tab.
3. Make sure there are at least 16 free files and 150 free blocks on your system disk.
4. Begin the installation procedure again.

If the installation procedure fails again, copy down the error message and contact your Tektronix service representative.

The **T,OFF** command in the preceding display suppresses subsequent output to your system terminal (except error messages) until **INSTALL** finishes executing. Within about five minutes, **INSTALL** will finish and your system terminal will display the following message:

```
*
* Your installation has been completed.
>
```

Once your software is installed, you can:

- Remove your disks and turn off your 8550 system
- Install more software onto your system disk
- Continue with the 80186/80188 Emulator demonstration run provided in Section 3 of this manual if your 80186/80188 Emulator is installed. If you continue, you do not have to restart the system or reset the date and time.

NOTE

At this point, "no.name" is the current user. To change the current user back to "yourname," enter **user,,yourname**.

Installing the Diagnostic Software

The Diagnostic Software Disk provided with your 80186/80188 Emulator replaces the Diagnostic Software Disk provided with your development system. Therefore, you may discard or write over your original disk. For more information on how to run the diagnostic software, refer to Section 7 of this manual or to the 80186/80188 Emulator Processor With Prototype Control Probe Service Manual.

Using Your 8086/8088 Assembler on the 8550

If you are using an 8550 and want to assemble 80186/80188 code (providing the code does not contain instructions unique to the 80186 or 80188 microprocessors), you must use the 8086/8088 Assembler. To use the 8086/8088 Assembler, enter the following commands one time only. Your 8086/8088 Assembler and 80186/80188 Emulator Control Software must be installed and the user set to **TEKTRONIX**.

```
> fl /EOS/8086/ASM[] /EOS/8086/ASM[]  
Flink ASM[]          to ASM[]    ?y  
  
> fl /EOS/8086/ASM.3 /EOS/8086/ASM.3  
Flink ASM.3          to ASM.3    ?y
```

Section 7

VERIFICATION PROCEDURES

INTRODUCTION

After the 80186/80188 Emulator is installed in your development system, it should be checked for proper operation. The first part of this section discusses the equipment necessary to verify proper emulator operation. The second part of this section discusses the equipment necessary to verify timing relationships of signals at the pins of the prototype control probe's probe plug. Finally, this section describes how to perform a system functional verification.

Every time you install an emulator, you should run the system functional verification checks. Emulator diagnostics are run automatically as part of the system diagnostic test program. Procedures for running this diagnostic test program are included in this section and in your system's Installation Guide.

EQUIPMENT REQUIRED

To check the operation of the installed 80186/80188 Emulator, the following equipment is required:

- TEKTRONIX MicroLab I (067-0892-xx)
- 80186/80188 Personality Card (018-0211-00)

The MicroLab I checks the 80186/80188 Emulator by providing a circuit with known characteristics (the personality card). The personality card is monitored by MicroLab I circuitry, and test results are indicated on the MicroLab I display. The MicroLab I operating system also contains tests that exercise the functions of the prototype control probe.

Throughout this section it is assumed that you are familiar with the MicroLab I and its characteristics. For more information about this equipment, refer to the MicroLab I Instruction Manual and the 80186/80188 Personality Card Supplement.

FUNCTIONAL TEST PROCEDURES**EQUIPMENT SETUP**

1. Ensure that the power to the development system and to the MicroLab I is turned OFF.
2. Verify that all jumpers are installed correctly on the personality card. (Refer to the 80186/80188 Personality Card supplement.)
3. Install the personality card in the MicroLab I.
4. Insert the 68-pin probe plug of the prototype control probe in the Textool 68-pin Chip Carrier socket on the 80186/80188 Personality Card. For proper probe plug insertion, refer to "Connecting to the Prototype" in Section 6 of this manual.

TEST PROCEDURE

1. Turn ON power to the development system and to the MicroLab I.
2. Start up the operating system. For detailed information on system operation, refer to your System Users Manual.
3. Enter the following command to identify the emulator to be tested. If you are using an 8550, enter:

```
> sel 80186          or          > sel 80188
```

If you are using an 8540, enter:

```
> sel 80186
```

4. Enter the desired emulation mode. Perform step a or b:
 - a. To verify memory mapping capability, select emulation mode 1, enter:

```
> em 1
```

When the prompt character (>) is displayed, enter:

```
> map u 00000 0FFFFFF
```

This transfers all memory to the MicroLab I.

- b. If only the operation of the emulator is to be tested, select emulation mode 2, enter:

```
> em 2
```

5. Enter the following command to start program execution:

```
> reset
```

Then, while holding down the RESET key on the MicroLab I, enter:

```
> g 0
```

Release the RESET key.

If the 80186/80188 Emulator is operating properly, the MicroLab I will display "HELLO". This display indicates that most of the emulator circuitry is working properly. However, several control lines are not checked during initialization and should be verified with the MicroLab I Processor Tests, explained later in this section.

No HELLO Display

If the MicroLab I does not display "HELLO", a problem exists with the 80186/80188 Emulator boards, the prototype control probe, or the MicroLab I. Use the following procedure to determine which unit is not working:

1. Turn off the power to the MicroLab I and to the development system.
2. Disconnect the probe plug from the personality card's 68-pin chip carrier socket.

CAUTION

The 80186 or 80188 microprocessor is subject to damage by static discharge when not installed in a socket. Use extreme caution and observe anti-static precautions when handling the leadless chip carrier. Do not touch the pins. The microprocessor should be stored in conductive foam when not in use.

3. Install the personality card's 80186/80188 microprocessor device in the 68-pin chip carrier socket.
4. Turn on the power to the MicroLab I.

If the MicroLab I now displays "HELLO", the problem is with the 80186/80188 Emulator or the prototype control probe. Refer to the 80186/80188 Emulator Processor With Prototype Control Probes Service Manual for servicing information.

If the MicroLab I does not display "HELLO", a problem exists with the MicroLab I. Refer to the MicroLab I Instruction Manual and 80186/80188 Personality Card Supplement for servicing information.

PROCESSOR TEST PROCEDURE

If the emulator has passed the Functional Test Procedure, you are ready to run the Processor Tests. While each test is being performed, the display will show "PrOC n"; n is the number of the test being performed.

Perform the following procedure:

1. Press the RESET key on the MicroLab I keypad. This initializes the Processor Test hardware in the MicroLab I.
2. Press the PROC TEST (Shift 1) key to start the Processor Tests. The display will show "Pn".
3. Press the 0 key. The MicroLab I will execute the PROC 0 test and then display "SPECIAL".
4. Press the SPECIAL key. The MicroLab I performs PROC 1 through PROC 4 tests and part of PROC 5 test automatically and then displays "SPECIAL" again.

NOTE

Test PROC 1 is executed if jumper P1091 on the Personality Card is in the QUEUE STATUS mode. If P1091 is in the NORMAL mode, test PROC 1 is skipped and test PROC 2 is executed.

5. Press the SPECIAL key again to complete PROC 5 test and automatically perform PROC 6 through PROC 8 tests. When PROC 8 test is finished, the MicroLab I displays "rEAdY".

NOTE

Processor tests PROC 9, PROC A, PROC B, and PROC C are stand-alone tests. Only one test is conducted depending on the interrupt configuration of your emulator.

6. Press the key 9, A, B, or C depending on which test you want. When this test is finished, the MicroLab I again displays "rEAdY".

Refer to the appropriate personality card supplement to the MicroLab I Instruction Manual for processor test error codes.

EMULATOR TIMING VERIFICATION

Occasionally you may want to verify timing relationships between signals at the pins of the prototype control probe's probe plug. The following paragraphs discuss equipment necessary to perform these timing verifications.

Section 4 of this manual provides timing diagrams which show timing relationships.

MEASUREMENT CONSIDERATIONS

Emulator timing verification involves measurement of extremely fast signals. Test equipment used for these measurements should resolve timing differences of less than 5 ns between two signals. A resolution of 1 ns is best for examining the most critical timings.

Be careful that the test equipment does not introduce errors. Check test equipment calibration carefully. If you are using a dual trace oscilloscope rather than a dual beam model, be sure to account for any possible skew between the the two input channels. In general, good laboratory measurement practices ensure accurate results.

Equipment Required

To measure timing relationships with the preferred accuracy and resolution, you may use the following equipment:

- A TEKTRONIX 7844 Dual Beam Oscilloscope, or equivalent
- Two TEKTRONIX 7A26 Vertical Amplifiers, or equivalent
- A TEKTRONIX 7B85 Delaying Time Base, or equivalent

CONTROLLING THE SIGNAL LINES UNDER TEST

Some processor signal lines, such as interrupt lines, are normally connected to asynchronous circuits. Timing relationships of asynchronous signals may be difficult to measure with an oscilloscope. To exercise these signal lines in a periodic manner, it may be necessary to develop software routines or to use an external test fixture such as the TEKTRONIX MicroLab I.

SYSTEM FUNCTIONAL VERIFICATION

These paragraphs tell how to verify the operation of the 80186/80188 Emulator installed in an 8550 Microcomputer Development Lab or an 8540 Integration Unit. To perform verification with the 8550, you'll need a system terminal connected to the 8301 and the 8550 System Diagnostics Disk. To perform verification with the 8540, you'll need a system terminal connected to the 8540 and the 80186/80188 Diagnostics ROM installed in the 8540's System ROM Board.

This section does not provide a detailed description of the 8550 disk-resident diagnostics or the 8540 ROM-resident diagnostics. The information provided here gives you a procedure to verify the 80186/80188 Emulator's operation in the quickest way possible.

For detailed information about your development system's disk-resident or ROM-resident diagnostics, refer to the following optional service manuals:

- 8301 Microprocessor Development Unit Service Manual
- 8540 Integration Unit Service Manual

For more information on the emulator-specific diagnostics, refer to the 80186/80188 Emulator Processor With Prototype Control Probes Service Manual.

8550 MICROCOMPUTER DEVELOPMENT LAB VERIFICATION

When both the 8501 and 8301 have displayed the boot messages on the system terminal, you're ready to run the 8550 disk-resident diagnostics.

Procedure

To verify 80186/80188 Emulator operation, perform the following procedure:

1. Insert the 8550 System Diagnostics Disk label side up into the 8501's drive 0 (top drive).
2. Close the disk-drive door.

Various "test running" messages for both the 8301 and 8501 are displayed while the verification tests are executing. The 8301 (including the 80186/80188 Emulator) is tested first, followed by the 8501. The diagnostic tests take approximately 10 minutes to execute. At that time, the system terminal will display either of two messages:

SYSTEM VERIFICATION PASSED

or

SYSTEM VERIFICATION FAILED

If the SYSTEM VERIFICATION FAILED message is displayed, refer to the 8301 Microprocessor Development Unit Service Manual and 8501 Data Management Unit Service Manual for information on performing exhaustive diagnostic troubleshooting.

This completes the functional test procedure for an 80186/80188 Emulator installed in an 8550.

8540 INTEGRATION UNIT VERIFICATION

Procedure

NOTE

The 80186/80188 Diagnostic ROM must be installed in your 8540's System ROM Board before you can verify the emulator's operation. If not already installed, refer to Section 6 of this manual or the 8540 Integration Unit Installation Guide for ROM installation procedures.

1. Check that the Mode Selector switch on the 8540's System Controller Board is in its normal operating configuration: switch positions 0 through 2 and 4 through 7 set to 0, switch position 3 set to 1.
2. Power up the 8540.

After the 8540 has displayed its boot message on the system terminal, you're ready to run the 8540 ROM-based diagnostics.

3. Enter the following command at the system terminal:

```
> sel diags
```

The system terminal will display the diagnostic greeting and the Run Mode Menu, as shown in Display 7-2.

```
-----  
*  
*   TEKTRONIX INC.                               *  
*   8540 ROM-RESIDENT DIAGNOSTIC SYSTEM         *  
*   VERSION X.X                                 *  
*   Copyright (C) 1981 Tektronix, Inc.         *  
*  
*****  
  
                        RUN MODE MENU  
                        -----  
  
1 - AUTOMATIC MODE      ***** Default *****  
2 - SELECT MODE  
  
Type mode :
```

Display 7-2

4. Press the RETURN key to select the Automatic Mode (system verification) as soon as the system terminal displays the diagnostic greeting message and Run Mode Menu. Immediately, the Automatic Mode Menu is displayed.
5. Press the RETURN key again to select the automatic system verification tests. Immediately, the Display Option Menu is displayed.
6. Press the RETURN key a third time to select terminal display and to start execution of the automatic system verification tests. No further intervention is required.

Various "test running" messages are displayed while the verification tests are executing. The diagnostic tests take a approximately 10 minutes to execute. At that time, the system terminal will display either of two messages:

SYSTEM VERIFICATION PASSED

or

SYSTEM VERIFICATION FAILED

If the SYSTEM VERIFICATION FAILED message is displayed, refer to the 8540 Integration Unit Service Manual for information on performing exhaustive diagnostic troubleshooting.

This completes the functional test procedure for an 80186/80188 Emulator installed in an 8540.