TEXAS INSTRUMENTS

Improving Man's Effectiveness Through Electronics

Model 990 Computer TMS 9900/9980A Emulator and Buffer Modules

Installation and Operation Manual

MANUAL NO. 946245-9701
ORIGINAL ISSUE 15 JUNE 1977
REVISED 15 APRIL 1978

Digital Systems Division

LIST OF EFFECTIVE PAGES

INSERT LATEST CHANGED PAGES DESTROY SUPERSEDED PAGES

Note: The portion of the text affected by the changes is indicated by a vertical bar in the outer margins of the page.

Model 990 Computer TMS 9900/9980A Emulator and Buffer Modules
Installation and Operation Manual (946245-9701)

Original Issue . . . . . 15 June 1977
Revised and Reissued . . . . . 15 April 1978

Total number of pages in this publication is 64 consisting of the following:

PAGE NO.	CHANGE NO.	PAGE NO.	CHANGE NO.	PAGE NO.	CHANGE NO.
Cover	0				
Effective Pages	0				
iii-iv	0				
1-1 - 1-10	0				
2-1 - 2-14	0				
3-1 - 3-22	0				
4-1 - 4-10	0				
Alphabetical Index Div.	0				
Index-1 - Index-2	0				
User's Response	0				
Business Reply	0				
Cover Blank	0				
Cover	0				

PREFACE

This manual provides installation procedures, interface requirements, programming information and operation instructions for the emulator and buffer modules of the AMPL* Microprocessor Prototyping Lab. When employed with an FS990 System, these modules provide a convenient, effective tool for TMS 9900 and TMS 9980A microprocessor prototyping – both hardware and software development from the breadboard stage through the fully operational system.

A series of depot-level manuals are available for customers desiring to provide their own depot-level maintenance.

This manual is organized into four sections including:

I General Description – Provides a general description of the AMPL Microprocessor Prototyping Lab.

II Installation – Describes details of unpacking, mounting, and cabling the system.

III Programming – Provides interface data necessary for generating software for the requirements of the particular installation.

IV Operation – Describes loading, controls, and normal operating procedures.

The following publications contain additional information related to the AMPL Microprocessor Prototyping Lab.

Title	Part Number
Model 990 Computer Model TMS 9900/TMS 9980A Emulator and Buffer Depot Maintenance Manual	946239-9701
Model 990 Computer Logic State Trace Data Module Installation and Operation Manual	946241-9701
Model 990 Computer Logic State Trace Data Module Depot Maintenance Manual	946242-9701
AMPL Microprocessor Prototyping Lab System Operation Guide	946244-9701
Model 990 Computer Diagnostics Handbook	945400-9701
Model 990 Computer/TMS 9900 Microprocessor Assembly Language Programmer's Guide	943441-9701
Model 990/4 Computer System Hardware Reference Manual	945251-9701

*Texas Instruments Trademark

Title	Part Number
Model 990/10 Computer System Hardware Reference Manual	945417-9701
Model 990 Computer Peripheral Equipment Field Maintenance Manual	945419-9701
990 Computer Family System Handbook	945250-9701

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (Continued)

LIST OF TABLES

SECTION I

GENERAL DESCRIPTION

1.1 INTRODUCTION

The AMPL* Microprocessor Prototyping Laboratory is an integrated system of hardware and software used in the development of microprocessor systems. The AMPL system provides all the tools necessary for hardware and software development and checkout, so that the designer will have the means to build a fully operational microprocessor developmental prototype. Figures 1-1 and 1-2 illustrate the basic system implemented to interface with the designer's microprocessor breadboard (target system).

As indicated in these figures, the AMPL system consists of five major units including:

- FS990 (or DS990) host computer – 990/4 (optionally, 990/10) computer with a minimum of 24K memory which controls and exercises the complete system. The system software which executes in the computer provides the control necessary to simulate an operational environment for the target system and to check the operation of the prototype hardware and software under dynamic operating conditions.

- Emulator module (figure 1-3) -- operates in conjunction with the buffer module (TMS 9900 or TMS 9980A) to permit the host computer to control the user's microprocessor chip to debug the user's hardware and software. The emulator module contains an 8K-byte user memory and a 256-word address trace memory that can optionally be placed in the user microprocessor's address space.

- Buffer module (figure 1-4) – houses the controlling microprocessor chip (TMS 9900 or TMS 9980A) and connects to the microprocessor IC socket in the target breadboard to effect the interface between the AMPL system and the target system.

- Trace module – an optional software-controlled logic state recorder used for both hardware and software checkout. This module contains a 256-word trace memory which can be synchronized with the address trace memory on the emulator module for expanded debugging capabilities.

- Peripheral equipment -- a partial list of 990 Computer terminals and mass data storage equipment available for use with the FS990 system is provided in table 1-1. These peripheral devices increase the potential for using the prototyping system both as a microprocessor prototyping aid and as a high-performance minicomputer data processing center.

Functional description and specifications for the emulator and buffer modules are contained in the following paragraphs. For a full description of the prototyping system, refer to the *AMPL Microprocessor Prototyping Laboratory System Operation Guide,* part number 946244-9701.

1.2 THE EMULATOR MODULE

The emulator module is contained on a circuit board (figure 1-3) that plugs into a full-width CRU slot in the 990 Computer chassis. The emulator accepts data, address and control information from the host computer and provides the memory and control functions to communicate with either the TMS 9900 or TMS 9980A buffer module (figure 1-4). The emulator

*Texas Instruments Trademark

Table 1-1. FS990 Peripheral Equipment

Model	Name
911 VDT	Video Display Terminal
FD800	Dual Floppy Disk
733 ASR *	Data Terminal with cassettes
743 KSR *	Data Terminal
804 *	Card Reader
810 *	Line Printer

NOTE: *indicates optional equipment

module and associated buffer module (figure 1-5) provide for AMPL system software control for debugging and error analysis in the target system software and hardware. The target system is exercised by programmed address trace, breakpoint, and data storage of problem states. The emulator contains an 8K-byte program memory and a 256-word trace control memory which may be placed within the microprocessor's address space. The emulator module also contains provisions for interfacing with an optional trace module to increase the capability of the AMPL system to trace the microprocessor's address or data bus.

1.3 THE TMS 9900 OR TMS 9980A BUFFER MODULES

The TMS 9900 (or TMS 9980A) buffer module houses the TMS 9900 (or TMS 9980A) chip which controls the microprocessor emulation. The buffer module is then connected to the user's target system through the microprocessor chip socket on the target breadboard. The interface includes control, data, clock, and memory address signals. Two-way gated tri-state drivers are employed to emulate the high impedance microprocessor interfaces for the 8-bit or 16-bit data bus and the 16-bit address bus. The buffer module also has an internal clock for testing and standalone operation. The internal clock may be disabled through a screwdriver-actuated switch accessible through a hole in the top of the module. This permits use of the clock circuits in the target system. In the TMS 9900 buffer, logic is employed to detect phase overlays to guard against microprocessor thermal damage.

1.4 INTERFACE DESCRIPTION

The buffer module interconnects the AMPL system to the user's breadboard through flat cables and a plug that install in the user's target microprocessor socket as shown in figure 1-6. The cabling for these interconnections is shown with the buffer module in figure 1-5. The buffer module also uses two 50-line ribbon cables for connecting to the emulator card connectors. The emulator and optional trace modules each plug directly into the chassis backpanel of the 990 host computer to interface with the computer's CRU bus. Figure 1-5 shows the emulator board edge plugs for the buffer module and trace module interface cables. The optional trace module control and data interface cables are routed to the emulator module on two ribbon cables which connect to the outer edge connectors of the module. The trace module may also be connected directly to selected test points on the target system breadboard by means of a ribbon cable extension and buffered clip-on test probes.

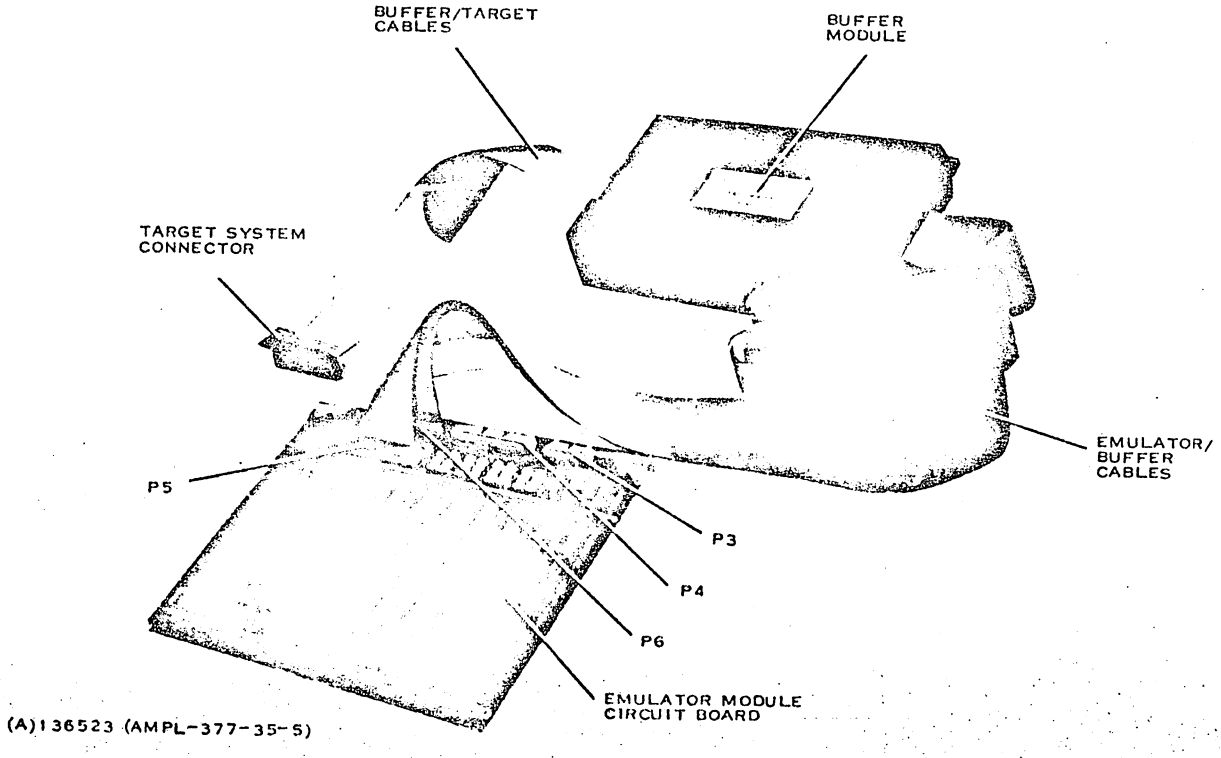


Figure 1-5. Emulator/Buffer and Interconnect Cables

1.4.1 EMULATOR MODULE OPERATION. See figure 1-6. The emulator module operates in conjunction with the software in the FS990 Computer and the control circuits in the buffer module to simulate real-time control and monitoring of target system operations. The emulator functions are:

- Controls access to the emulator's on-board target microprocessor.
- Contains programmable breakpoint logic to permit halting of microprocessor operations when a prespecified address appears on the address bus. Also sends the indication to trace module when the present memory address equals the value in the breakpoint register.
- Contains trace control logic that permits storing of memory addresses in the 256-word trace memory in either a continuous mode (overwrites old data until breakpoint is detected) or noncontinuous mode (halt when memory is full).
- Contains a maskable interrupt circuit that generates an interrupt for the FS990 each time the microprocessor is halted or detects a breakpoint.
- Contains CRU interface logic to permit communications between the serial CRU I/O bus in the FS990 system and the parallel memories and registers in the emulator.
- Contains a memory timing circuit that operates synchronously with TMS 9900/TMS 9980A memory cycles to generate an end of cycle indication to the emulator module (and trace module).

- Issues hold and trap commands to the microprocessor in response to CRU commands from the FS990.
- Accepts trace hold commands from trace module for synchronized operation.
- Contains provisions for synchronized operation with the trace module for expanded hardware and software debugging capabilities.
- Breakpoint comparison can be qualified by instruction acquisition (IAQ) to limit breakpoint to write cycles only or to instruction acquisition only.
- Tracing in the emulator can be qualified by IAQ so that only instruction accesses are recorded.

1.4.2 BUFFER MODULE OPERATION. See figure 1-6. The buffer module contains the microprocessor element (chip), memory address decoding logic, clock generator, and several networks of 3-state drivers required to interface the emulator module with the target system hardware. Through the use of an on-board clock circuit, the microprocessor can be exercised for software development before target hardware is available. When system hardware is ready for checkout, the on-board clock is disabled through a screwdriver-actuated switch, and the user's clock is used as the system clock.

The TMS 9900 buffer module includes a phase-overlap circuit that automatically removes power from the microprocessor and issues a clock error signal to the emulator if phase overlap is detected on clock phases 2 and 4. This prevents possible thermal damage to the microprocessor. The buffer module also provides overvoltage protection for the microprocessor.

The TMS 9980A buffer provides a source of CKIN (pin on the 9980A) for non-target system operation of the buffer. Clock error status is not supported since the four phases of the system clock are generated on the TMS 9980A chip.

SECTION II

INSTALLATION

2.1 GENERAL

This section provides instructions for the installation and checkout of the emulator and buffer modules. This section also details the planning requirements and unpacking instructions for these modules. To provide a basic operable system for initial checkout of the emulator and buffer modules, their interface to the computer and essential peripherals are referenced.

2.2 PLANNING

The emulator module requires one full slot of the Model 990 Computer CRU interface, from which it obtains power for itself and its associated buffer. These power requirements are as follows:

+5 volts	2.9A typical
+12 volts	0.7A typical
-5 volts	2 ma typical

The host computer can employ a variety of peripheral devices as described in Section I; however, minimum equipment required for checkout and operation of the emulator and buffer modules includes the FS990 system (990/4 or 990/10 Computer, interactive console device (773, 913, 911), FD800 dual floppy disk and TX990/TXDS operating system software) and AMPL software. Installation, operation and programming for the computer and peripherals are provided in their respective installation and operation manuals, and in the *Model 990 Computer/TMS 9900 Microprocessor Assembly Language Programmer's Guide* (part number 943441-9701).

2.3 UNPACKING

Except when they are shipped installed in the 990 Computer, the emulator and buffer modules are wrapped in bubble-pack and contained within a cardboard shipping carton. When shipped with the computer, the emulator module is usually installed in the computer chassis and the buffer module is separately wrapped and contained within the computer package. In this case, refer to the unpacking instructions in the *Model 990/4 Computer System Hardware Reference Manual* part number 945251-9701. For the individually packaged modules, unpacking and inspection instructions are as follows:

1. Before opening the shipping carton, inspect for signs of possible damage to the contents. Look for such indications as crumpled corners, tears, water stains, loosened scaling tape, etc. Notify supervisory personnel of any such abuses to the carton before proceeding.

2. Open the cardboard box and remove the bubble-pack wrapped modules. Remove the bubble-pack wrapping and verify that the modules' part numbers agree with shipping invoices. Otherwise, notify supervisor.

3. Visually inspect the modules and cables for any possible damage, especially if the shipping carton or wrapping showed any shipping mishandling. Look for breaks or cracks in the printed wiring board, loose or missing components or connectors, broken wires, stains or corrosion, and contamination or foreign material lodged between components or pins. Notify supervisory personnel of any discrepancies.

4. Inventory the parts received to verify that the shipment of components agrees with the bill of lading and that the part numbers agree with those listed in table 2-1 for the TMS 9900 or TMS 9980A configurations.

2.4 INSTALLING THE EMULATOR AND BUFFER MODULES

These instructions for installing the emulator module apply when this module is packaged and shipped separately from the host computer or when the emulator and buffer modules are to be added to an existing FS990 system. If the AMPL system is received as a unit with the modules installed in the standard configurations, the CRU base addresses have been established at the factory and are compatible with the software package. Also, in that case, the correct interrupt levels have been determined and prewired in the interrupt jumper plugs for the peripherals. No alterations or modifications of the 990 Computer chassis are required. Configurations other than the standard ones are available from Texas Instruments to modify existing FS990 systems to suit user requirements.

Table 2-1. Part Numbers for Emulator/Buffer Components

Component	Part Number
Emulator Module	949925-0001
TMS 9900 Buffer Assembly	949937-0001
Emulator/Buffer Cable	949924-0001
Emulator/Buffer Cable	949924-0002
Buffer/Target Cable	949923-0001
Buffer/Target Cable	949923-0002
Buffer/Target Clock Cable	949945-0001
TMS 9900 Target Connector	949905-0001
TMS 9900 Buffer Module	949995-0001
TMS 9980A Buffer Assembly	949937-0002
Emulator/Buffer Cable	949924-0001
Emulator/Buffer Cable	949924-0002
Buffer/Targer Cable	949923-0003
Buffer/Target Cable	949923-0004
TMS 9980A Target Connector	949955-0001
TMS 9980A Buffer Module	949940-0001

To determine new chassis slot assignments refer to figures 2-1 through 2-7, which show chassis slot numbers, the assigned CRU base addresses, and the interrupt levels for the several different configurations. The illustrations are as follows:

Figure	Title
2-1	Model 990/4 Computer 13-Slot Chassis Configuration for TX990 with 911 VDT
2-2	Model 990/10 Computer 13-Slot Chassis Configuration for TX990 with 911 VDT
2-3	Model 990/4 Computer 13-Slot Chassis Configuration for TX990 with 913 VDT
2-4	Model 990/10 Computer 13-Slot Chassis Configuration for TX990 with 913 VDT
2-5	Model 990/10 Computer 13-Slot Chassis Recommended DX10 Configuration without CRU Expansion Chassis
2-6	Model 990/10 Computer 13-Slot Chassis Recommended DX10 Configuration with CRU Expansion Chassis
2-7	Model 990/10 Computer 13-Slot CRU Expansion Chassis Recommended DX10 Configuration

	P1			P2		
CHASSIS SLOT NUMBER	CRU BASE ADDRESS	CIRCUIT BOARD	INTER-RUPT LEVEL	CRU BASE ADDRESS	CIRCUIT BOARD	INTER-RUPT LEVEL
1	N/A	990/4 AU W8/K	N/A	N/A	990/4 AU W/8K	N/A
2	02E0	40K 990/4 MEMORY EXPANSION	N/A	02C0	40K 990/4 MEMORY EXPANSION	N/A
3	02A0	8K 990/4 MEMORY EXPANSION (OPT.)	N/A	0280	8K 990/4 MEMORY EXPANSION (OPT.)	N/A
4	0260	SPARE	N/A	0240	SPARE	N/A
5	0220	SPARE	N/A	0200	SPARE	N/A
6	01E0	SPARE	N/A	01C0	SPARE	N/A
7	01A0	SPARE	N/A	0180	SPARE	N/A
8	0160	EMULATOR MODULE	6	0140	EMULATOR MODULE	6
9	0120	TRACE MODULE	6	0100	TRACE MODULE	6
10	00E0	911 VDT CONTROLLER	3	00C0	911 VDT CONTROLLER	3
11	00A0	FLOPPY DISC CONTROLLER	7	0080	FLOPPY DISC CONTROLLER	7
12	0060	LINE PRINTER (OPTIONAL)	6	0040	CARD READER (OPTIONAL)	4
13	0020	PROM PROGRAM-MER (OPTIONAL)	6	0000	733 ASR (OPTIONAL)	6

(A)137447

Figure 2-1. Model 990/4 Computer 13-Slot Chassis Configuration for TX990 with 911 VDT

CHASSIS SLOT NUMBER	P1 CRU BASE ADDRESS	P1 CIRCUIT BOARD	P1 INTER-RUPT LEVEL	P2 CRU BASE ADDRESS	P2 CIRCUIT BOARD	P2 INTER-RUPT LEVEL
1	N/A	990/10/AU2	N/A	N/A	990/10/AU2	N/A
2	02E0	990/10/AU1	N/A	02C0	990/10/AU1	N/A
3	02A0	16K MEMORY EXPANSION	N/A	0280	16K MEMORY EXPANSION	N/A
4	0260	48K MEMORY EXPANSION	N/A	0240	48K MEMORY EXPANSION	N/A
5	0220	SPARE	N/A	0200	SPARE	N/A
6	01E0	SPARE	N/A	01C0	SPARE	N/A
7	01A0	SPARE	N/A	0180	SPARE	N/A
8	0160	EMULATOR MODULE	6	0140	EMULATOR MODULE	6
9	0120	TRACE MODULE	6	0100	TRACE MODULE	6
10	00E0	911VDT CONTROLLER	3	00C0	911 VDT CONTROLLER	3
11	00A0	FLOPPY DISC CONTROLLER	7	0080	FLOPPY DISC CONTROLLER	7
12	0060	LINE PRINTER (OPTIONAL)	6	0040	CARD READER (OPTIONAL)	4
13	0020	PROM PROGRAM-MER (OPTIONAL)	6	0000	733 ASR (OPTIONAL)	6

(A)137448

Figure 2-2. Model 990/10 Computer 13-Slot Chassis Configuration for TX990 with 911 VDT

CHASSIS SLOT NUMBER	P1 CRU BASE ADDRESS	P1 CIRCUIT BOARD	P1 INTER-RUPT LEVEL	P2 CRU BASE ADDRESS	P2 CIRCUIT BOARD	P2 INTER-RUPT LEVEL
1	N/A	990/4 AU W/8K	N/A	N/A	990/4 AU W/8K	N/A
2	02E0	40K 990/4 MEMORY EXPANSION	N/A	02C0	40K 990/4 MEMORY EXPANSION	N/A
3	02A0	8K 990/4 MEMORY EXPANSION (OPT.)	N/A	0280	8K 990/4 MEMORY EXPANSION (OPT.)	N/A
4	0260	SPARE	N/A	0240	SPARE	N/A
5	0220	SPARE	N/A	0200	SPARE	N/A
6	01E0	SPARE	N/A	01C0	SPARE	N/A
7	01A0	SPARE	N/A	0180	SPARE	N/A
8	0160	EMULATOR MODULE	4	0140	EMULATOR MODULE	4
9	0120	TRACE MODULE	4	0100	TRACE MODULE	4
10	00E0	913 VDT CONTROLLER	3	00C0	913 VDT CONTROLLER	3
11	00A0	FLOPPY DISC CONTROLLER	7	0080	FLOPPY DISC CONTROLLER	7
12	0060	LINE PRINTER (OPTIONAL)	4	0040	SPARE	4
13	0020	PROM PROGRAM-MER (OPTIONAL)	N/A	0000	733 ASR (OPTIONAL)	6

(A)137449

Figure 2-3. Model 990/4 Computer 13-Slot Chassis Configuration for TX990 with 913 VDT

CHASSIS SLOT NUMBER	P1 CRU BASE ADDRESS	P1 CIRCUIT BOARD	P1 INTER-RUPT LEVEL	P2 CRU BASE ADDRESS	P2 CIRCUIT BOARD	P2 INTER-RUPT LEVEL
1	N/A	990/10/AU2	N/A	N/A	990/10/AU2	N/A
2	02E0	990/10/AU1	N/A	02C0	990/10/AU1	N/A
3	02A0	16K MEMORY EXPANSION	N/A	0280	16K MEMORY EXPANSION	N/A
4	0260	48K MEMORY EXPANSION	N/A	0240	48K MEMORY EXPANSION	N/A
5	0220	SPARE	N/A	0200	SPARE	N/A
6	01E0	SPARE	N/A	01C0	SPARE	N/A
7	01A0	SPARE	N/A	0180	SPARE	N/A
8	0160	EMULATOR MODULE	4	0140	EMULATOR MODULE	4
9	0120	TRACE MODULE	4	0100	TRACE MODULE	4
10	00E0	913 VDT CONTROLLER	3	00C0	913 VDT CONTROLLER	3
11	00A0	FLOPPY DISC CONTROLLER	7	0080	FLOPPY DISC CONTROLLER	7
12	0060	LINE PRINTER (OPTIONAL)	4	0040	SPARE	4
13	0020	PROM PROGRAM-MER (OPTIONAL)	N/A	0000	733 ASR (OPTIONAL)	6

(A)137450

Figure 2-4. Model 990/10 Computer 13-Slot Chassis Configuration for TX990 with 913 VDT

CHASSIS SLOT NUMBER	P1 CRU BASE ADDRESS	P1 CIRCUIT BOARD	P1 INTER-RUPT LEVEL	P2 CRU BASE ADDRESS	P2 CIRCUIT BOARD	P2 INTER-RUPT LEVEL
1	N/A	990/10/AU2	N/A	N/A	990/10/AU2	N/A
2	02E0	990/10/AU1	N/A	02C0	990/10/AU1	N/A
3	02A0	16K MEMORY EXPANSION	N/A	0280	16K MEMORY EXPANSION	N/A
4	0260	48K MEMORY EXPANSION	N/A	0240	48K MEMORY EXPANSION	N/A
5	0220	16K MEMORY EXPANSION	N/A	0200	16K MEMORY EXPANSION	N/A
6	01E0	48K MEMORY EXPANSION	N/A	01C0	48K MEMORY EXPANSION	N/A
7	01A0	DISC CONTROLLER	13	0180	DISC CONTROLLER	13
8	0160	MAG. TAPE CONTR. OR TILINE COUPLER	9	0140	MAG. TAPE CONTR. OR TILINE COUPLER	9
9	0120	911 VDT CONTROLLER	8	0100	911 VDT CONTROLLER	10
10	00E0	EMULATOR MODULE	12	00C0	EMULATOR MODULE	11
11	00A0	TRACE MODULE	3	0080	TRACE MODULE	7
12	0060	LINE PRINTER (OPTIONAL)	14	0040	CARD READER (OPTIONAL)	4
13	0020	PROM PROGRAM-MER (OPTIONAL)	15	0000	733 ASR (OPTIONAL)	6

(A)137451

Figure 2-5. Model 990/10 Computer 13-Slot Chassis Recommended DX10 Configuration without CRU Expansion Chassis

CHASSIS SLOT NUMBER	P1 CRU BASE ADDRESS	P1 CIRCUIT BOARD	P1 INTER-RUPT LEVEL	P2 CRU BASE ADDRESS	P2 CIRCUIT BOARD	P2 INTER-RUPT LEVEL
1	N/A	990/10 AU2	N/A	N/A	990/10 AU2	N/A
2	02E0	990/10 AU1	N/A	02C0	990/10 AU1	N/A
3	02A0	16K MEMORY EXPANSION	N/A	0280	16K MEMORY EXPANSION	N/A
4	0260	48K MEMORY EXPANSION	N/A	0240	48K MEMORY EXPANSION	N/A
5	0220	16K MEMORY EXPANSION	N/A	0200	16K MEMORY EXPANSION	N/A
6	01E0	48K MEMORY EXPANSION	N/A	01C0	48K MEMORY EXPANSION	N/A
7	01A0	DISK CONTROLLER	13	0180	DISK CONTROLLER	13
8	0160	MAG. TAPE CONTR. OR TILINE COUPLER	9	0140	MAG. TAPE CONTR. OR TILINE COUPLER	9
9	0120	911 VDT CONTROLLER	8	0100	911 VDT CONTROLLER	10
10	00E0	911 VDT CONTROLLER	12	00C0	911 VDT CONTROLLER	11
11	00A0	CRU EXPANDER	3	0080	CRU EXPANDER	7
12	0060	LINE PRINTER (OPTIONAL)	14	0040	CARD READER (OPTIONAL)	4
13	0020	PROM PROGRAM-MER (OPTIONAL)	15	0000	733 ASR (OPTIONAL)	6

(A)137452

Figure 2-6. Model 990/10 Computer 13-Slot Chassis Recommended DX10 Configuration with CRU Expansion Chassis

CHASSIS SLOT NUMBER	P1 CRU BASE ADDRESS	P1 CIRCUIT BOARD	P1 INTER-RUPT LEVEL	P2 CRU BASE ADDRESS	P2 CIRCUIT BOARD	P2 INTER-RUPT LEVEL
1	N.A	CRU BUFFER	N/A	N/A	CRU BUFFER	N/A
2	06E0	TILINE COUPLER (OPTIONAL)	N/A	06C0	TILINE COUPLER (OPTIONAL)	N/A
3	06A0	SPARE	N/A	0680	SPARE	N/A
4	0660	SPARE	N/A	0640	SPARE	N/A
5	0620	SPARE	N/A	0600	SPARE	N/A
6	05E0	SPARE	N/A	05C0	SPARE	N/A
7	05A0	DISK CONTROLLER (OPTIONAL)	13	0580	DISK CONTR. OR LINE PRINTER(OPT.)	13
8	0560	MAGNETIC TAPE CONTROLLER (OPT.)	9	0540	MAGNETIC TAPE CONTROLLER (OPT.)	9
9	0520	911 VDT CONTROLLER	8	0500	911 VDT CONTROLLER	10
10	04E0	EMULATOR MODULE	12	04C0	EMULATOR MODULE	11
11	04A0	TRACE MODULE	3	0480	TRACE MODULE	7
12	0460	EMULATOR MODULE	14	0440	EMULATOR MODULE	4
13	0420	TRACE MODULE	15	0400	TRACE MODULE	6

NOTE: THE CRU BASE ADDRESSES SHOWN APPLY TO EXPANSION CHASSIS 1 (CONNECTED TO P3 OF CRU EXPANDER BOARD). ADD $(N-1) * 400_{16}$ TO THE ABOVE ADDRESSES FOR CRU BASE ADDRESSES IN EXPANSION CHASSIS N.

(A) 137453

Figure 2-7. Model 990/10 Computer 13-Slot CRU Expansion Chassis Recommended DX10 Configuration

Where the emulator module is preinstalled in the 990 Computer, steps 1 and 2 of the following instructions may be ignored. After the chassis slot assignments have been determined, the circuit board is installed by steps 1 and 2. Step 3 is the attachment of the emulator/buffer module cables to the emulator module circuit board.

NOTE

The buffer module is shipped as an assembly with the emulator cables and the target system cables attached.

1. Verify that computer chassis power is off.

NOTE

Before pushing the emulator circuit board fully into the backpanel connectors, attach the buffer module cable connectors as in step 3.

2. Insert the emulator circuit board into the assigned CRU slot with the component side up, so that the two PCB connectors will mate with the computer backpanel.

3. Since the mating connector to one of the emulator/buffer cables is slightly inset on the emulator circuit board (connector P5 as shown in figure 2-8), these connectors should be joined before the emulator is fully inserted into the computer backpanel. These connectors must be installed so that embossed arrowheads on the index end of the connectors are aligned point-to-point for the mating connector pairs, as shown in figure 2-8.

NOTE

Pin 1 of the connector is located on the same side as the red stripe on the ribbon cable.

4. Attach the buffer module cable connectors P5 and P6 to the emulator module connectors P5 and P6, respectively.

5. Push the emulator module circuit board fully into the backpanel connectors. When the emulator is fully installed, the plastic ejectors on the outer corners of the circuit board should settle snugly against the frame.

2.5 CABLING CONFIGURATIONS.

The emulator works in conjunction with the buffer module to replace the microprocessor and/or the program memory in the user's target system. The buffer module houses the microprocessor elements and completes the interface between the emulator and the target system.

CAUTION

Damage to the equipment can occur if plugs and connectors are not oriented and aligned correctly.

Two ribbon cables are routed from the buffer module to the emulator; and three other ribbon cables (two cables for the TMS 9980A) are routed between the buffer module and the target connector. Configurations of the cabling from the buffer module to the emulator and to the target system connector are shown in figure 2-9. When installing any plug/receptacle always be sure that the embossed arrowheads are aligned for correct orientation before mating the pair together. Then align the pins, match the connectors together evenly, and press on the plug until it is firmly seated.

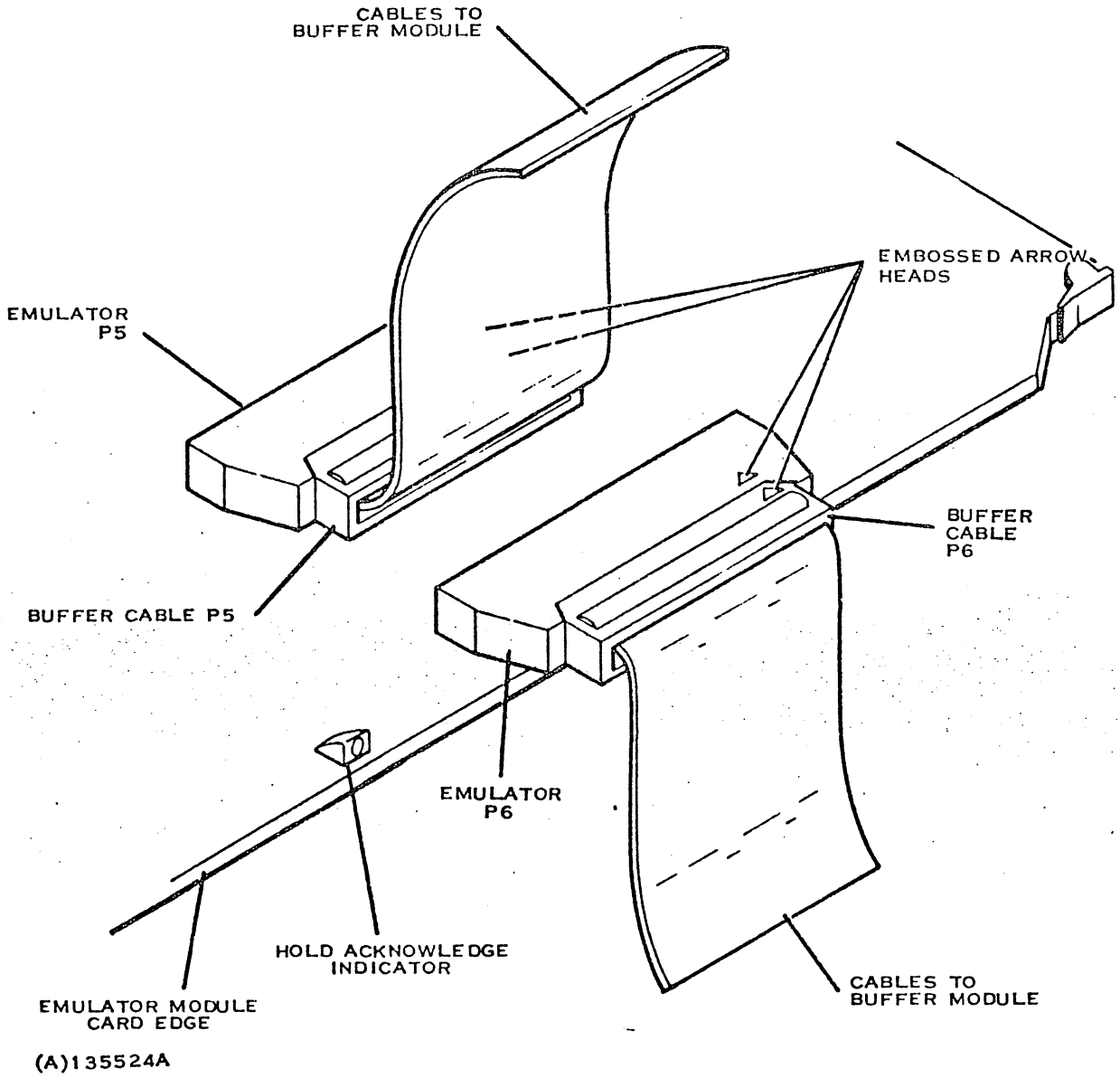


Figure 2-8. Emulator/Buffer Cable Connections at Emulator

2.5.1 CABLE CONNECTIONS BETWEEN EMULATOR AND TRACE MODULES. When a trace module is used in the AMPL system, two cables are connected between the emulator module and the trace module as shown in figure 2-10. These cables are connected P3-to-P3 and P4-to-P4. However, when the trace module is connected to a target system via a trace data probe, the data cable between the two modules (P4-to-P4) is not used.

2.5.2 TARGET SYSTEM CONNECTORS.

CAUTION

Be sure that system power is "off" before removing the microprocessor element or connecting the target system connector. If system power is "on", do not permit the connector pins of the element to contact any conductive, energized, or grounded surface.

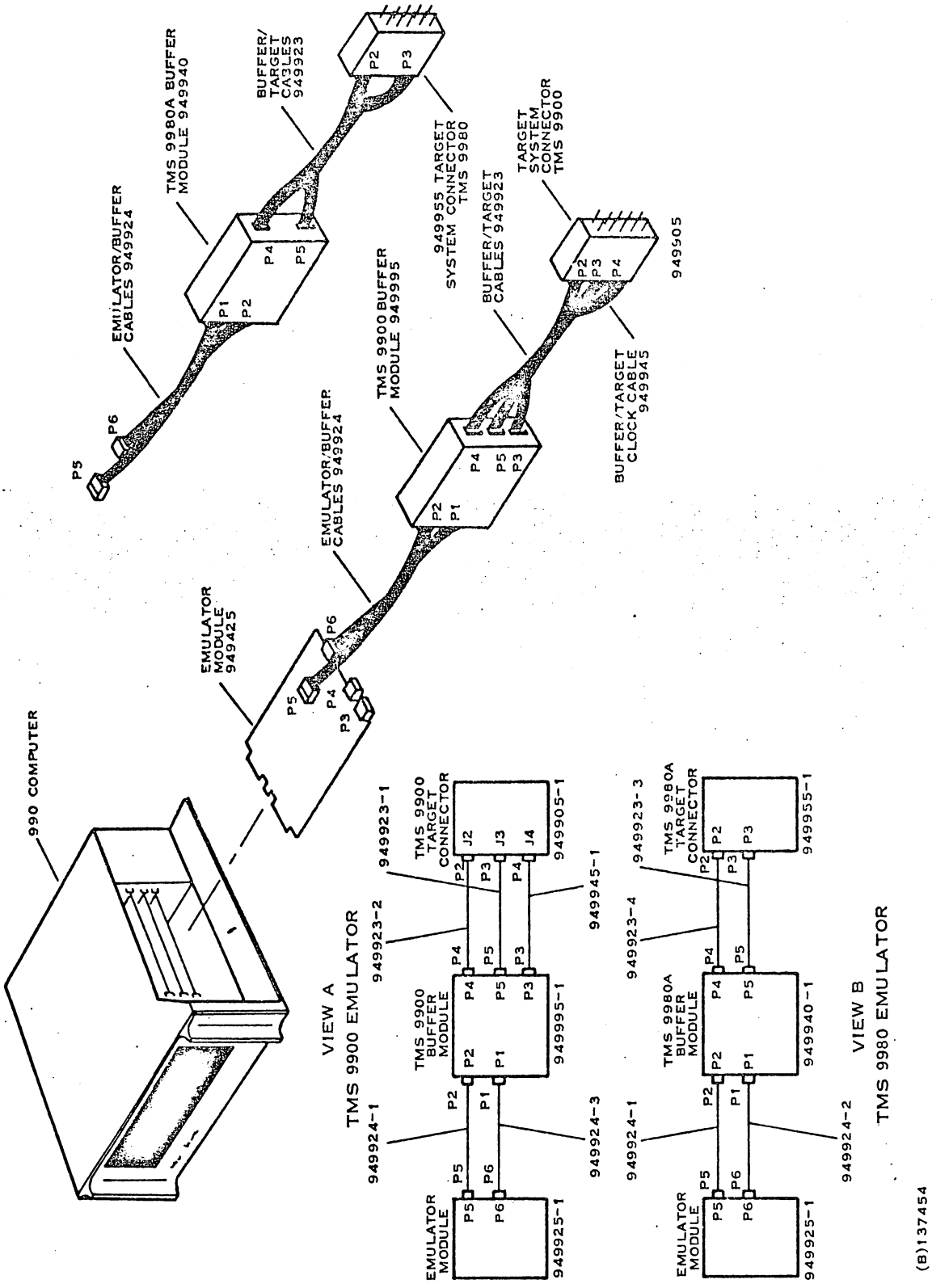
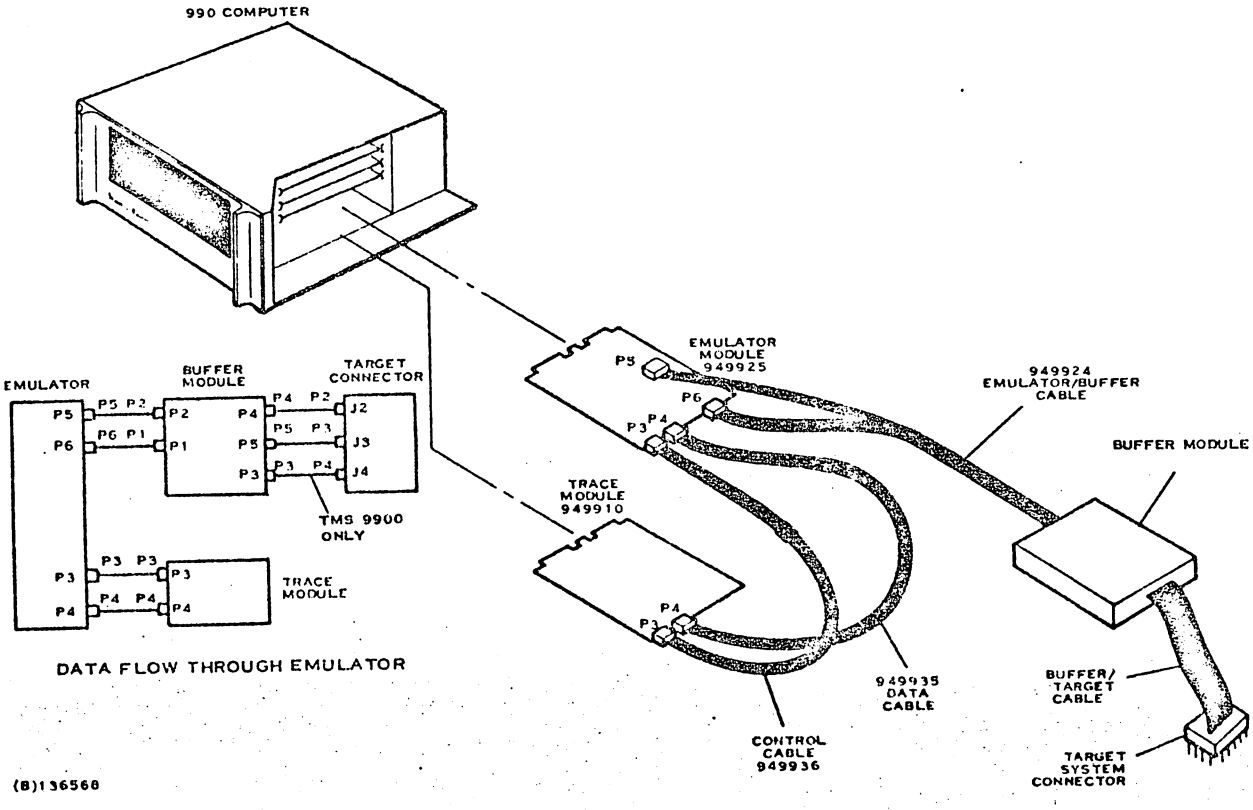


Figure 2-9. Emulator/Buffer Cabling Diagram



NOTE: Data Cable, P/N 949935, is not used when trace module (P4) is connected to trace data probe

Figure 2-10. Trace Module and Emulator Module Interconnecting Cabling

As shown in figure 2-11, the target system connector for the TMS 9900, part number 949905, receives three cable connectors from the buffer module: P2, P3, and P4. These cable connectors are installed in the target connector and oriented as shown in the illustration.

CAUTION

Be sure the microprocessor target connectors are properly polarized (pin 1-to-pin 1) and aligned when inserted into the socket. The beveled index corner (adjacent to pin 1) must be aligned with pin 1 of the target system receptacle.

If it is necessary to remove any of the plugs, always check polarization before reinstalling. Note that pin 1 of each plug is located adjacent to the embossed arrowhead, and that pin 1 of the mating socket is at the opposite end of the marking for that socket.

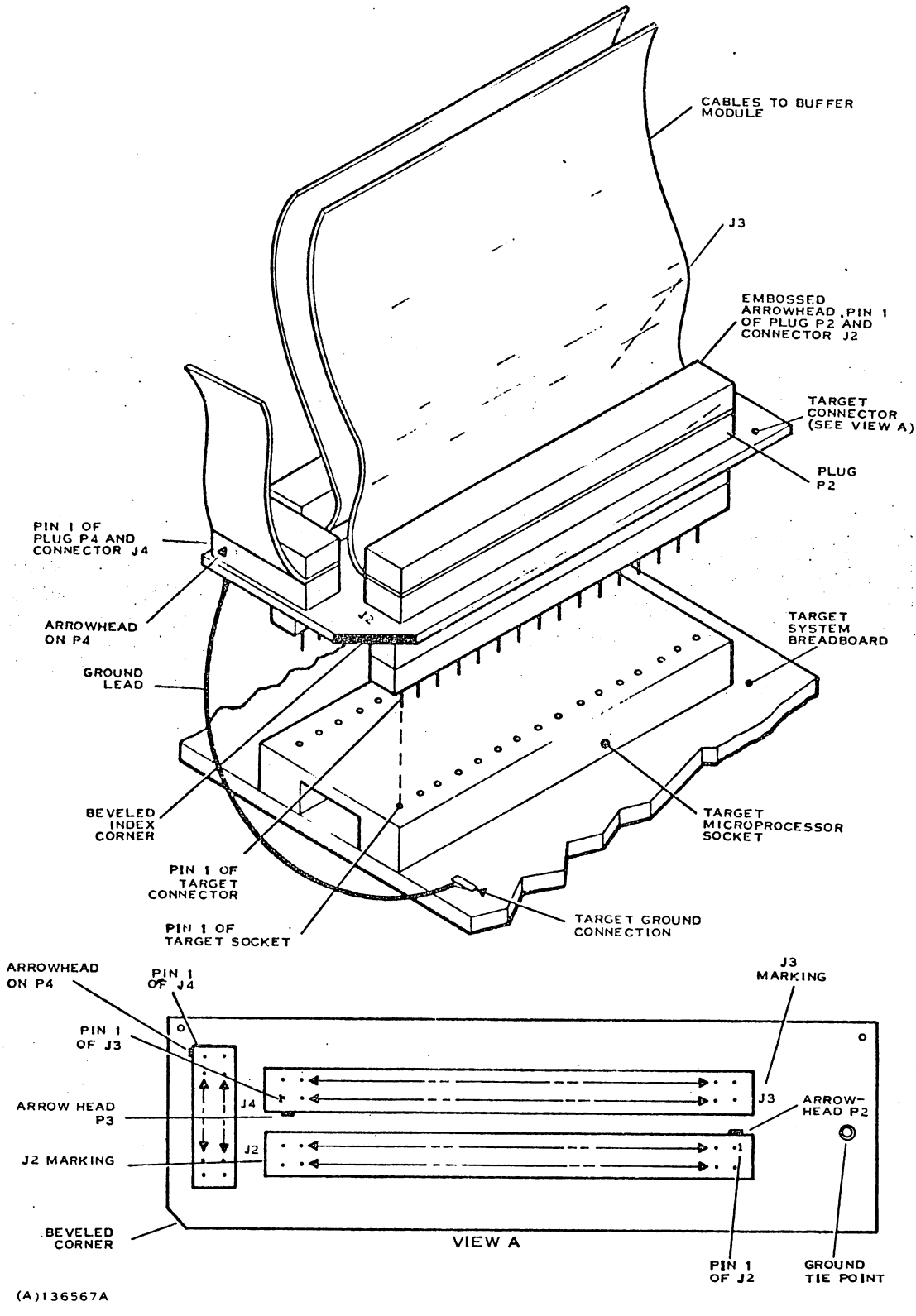


Figure 2-11. Installation of Buffer Cables and Connector at TMS 9900 Target System

As shown in figure 2-12, the target system connector for the TMS 9980A, part number 949955, receives two cable connectors from the buffer module: P2 and P3. The illustration shows the correct orientation of the cable connectors in regards to the target system connector. Also, note that pin 1 of the target system is located adjacent to the beveled corner of the connector. When the system cable connectors have been installed in the target system connector, remove the TMS 9900 or TMS 9980A microprocessor element from the target system and install the target system connector in place of the microprocessor element. For TMS 9900 only, when the target system connector has been installed and seated in the microprocessor element socket, connect the alligator grounding clip (P5) to any convenient signal ground on the target system breadboard.

2.5.3 **SYSTEM CLOCK SELECTION.** When all of the connections have been made, insert a slot screwdriver into the cutout of the buffer module cover and select either an INTERNAL prototyping system clock or the TARGET SYS external clock, as required by the program. See figure 2-13.

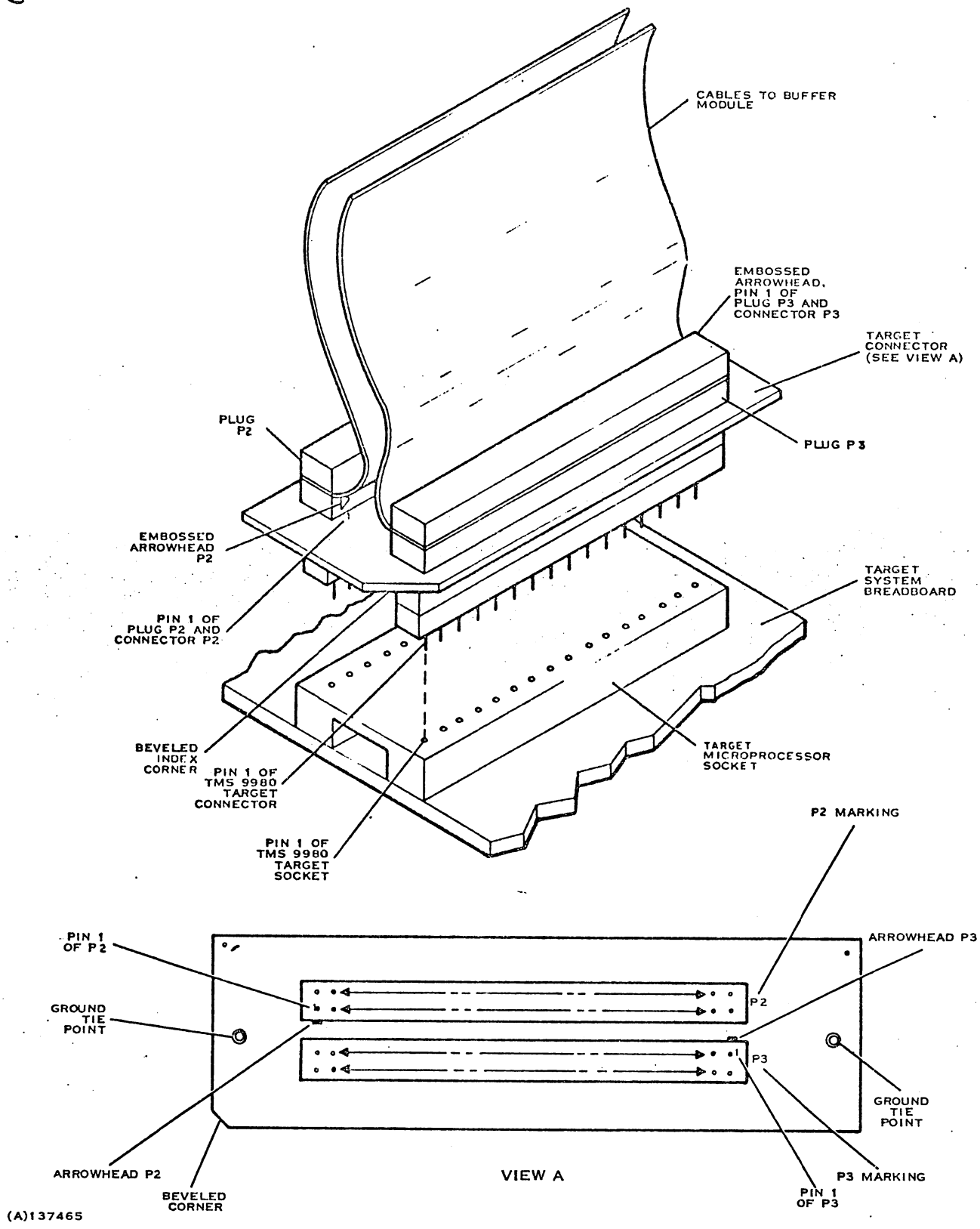


Figure 2-12. Installation of Buffer Cables and Connector at TMS 9980A Target System

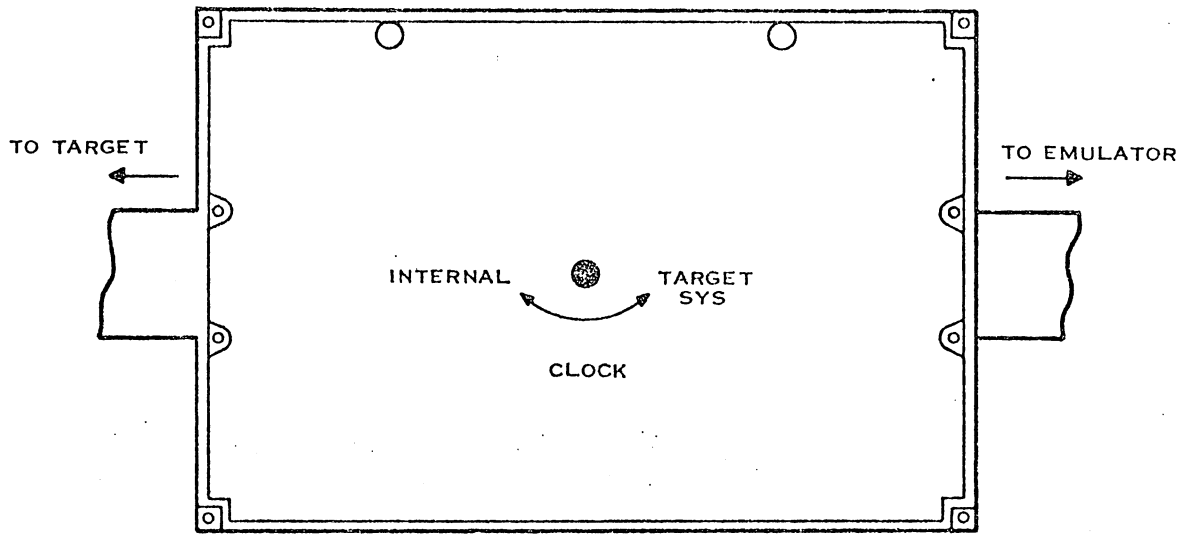


Figure 2-13. Buffer Module

SECTION III

PROGRAMMING

3.1 GENERAL

This section describes the interface between the emulator and buffer modules and the target system, and provides information that an assembly language programmer will need to develop program routines for the particular software and hardware requirements of his prototyping system. Interface information and programming definitions are described in the following paragraphs. Programming techniques and sample programs for typical target system analysis are also presented. This information is intended for programmers experienced with the 990 Computer; therefore, only basic programming requirements are given here. For more programming information, refer to the *Model 990 Computer/TMS 9900 Microprocessor Assembly Language Programmer's Guide,* part number 943441-9701.

3.2 INTERFACE INFORMATION

The emulator module interface with the host 990 Computer CRU bus is described in figures 3-1 and 3-2.

3.3 PROGRAMMING DEFINITIONS

The following paragraphs define the emulator control functions.

3.3.1 HOLD. A hold is asserted to the target microprocessor by the host CRU command to halt execution in the target system. Once the hold is acknowledged by the microprocessor, the host may invoke the emulator functions.

3.3.2 TRAP. Two trap vectors (WP and PC pointers) are provided. Reset at address 0000_{16} and Load at address $FFFC_{16}$ control the program execution and debug functions.

3.3.3 MEMORY SELECTION. The emulator has an 8K-byte memory (beginning at address 0) and a 256-word trace/control memory (beginning at address $FE00_{16}$). These memories may be selectively placed in the target system address space or taken out, in which case the target system uses its own memory.

NOTE

The trace/control memory serves a dual purpose, either as user memory or as the trace buffer during the trace operation of the emulator (see paragraph 3.3.6).

3.3.4 MEMORY CONTROL. The emulator may read/write data at any target memory location.

3.3.5 RUN. Run is the state of the emulator when a hold is released.

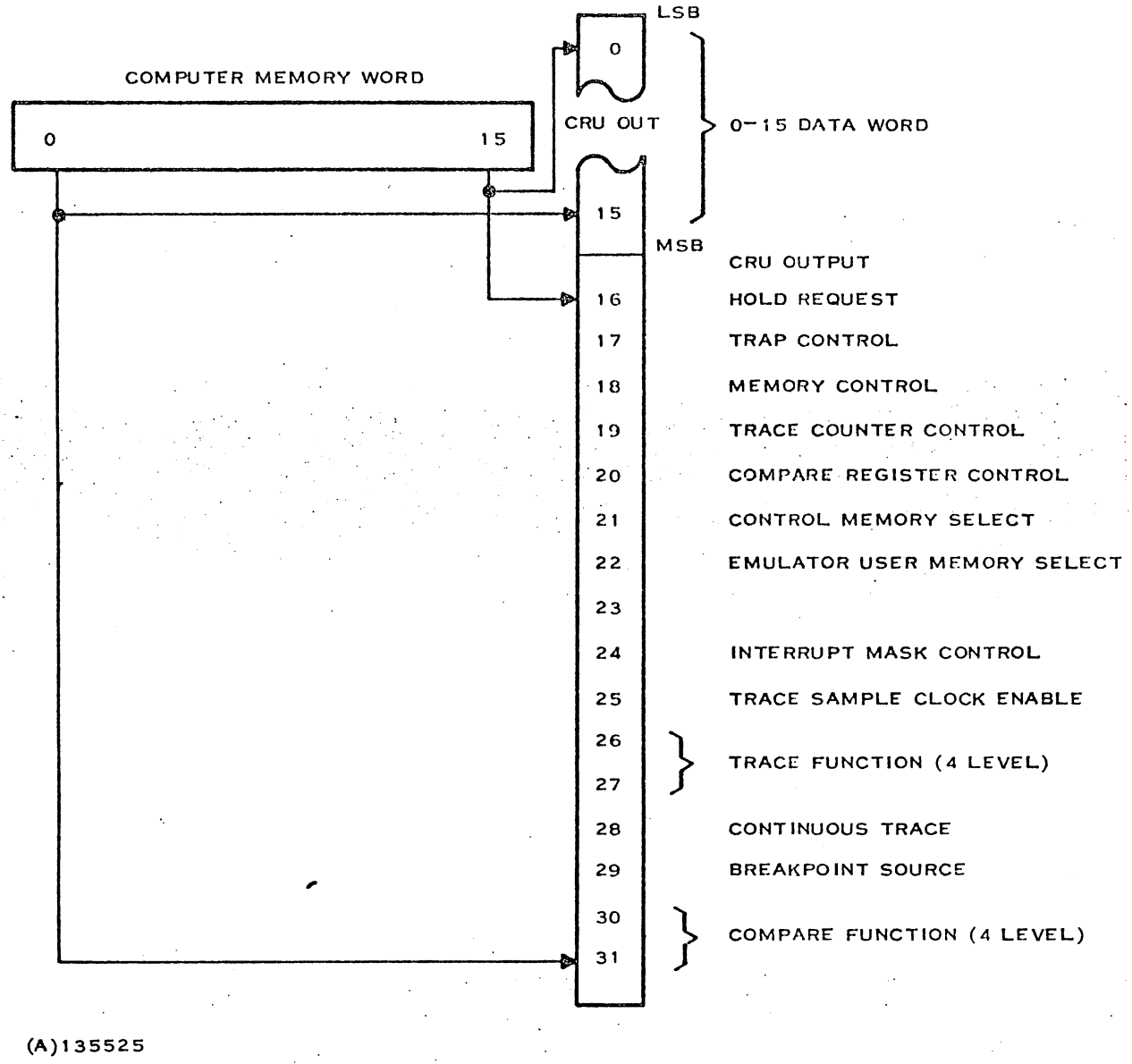


Figure 3-1. CRU to Emulator Data/Control Word Format

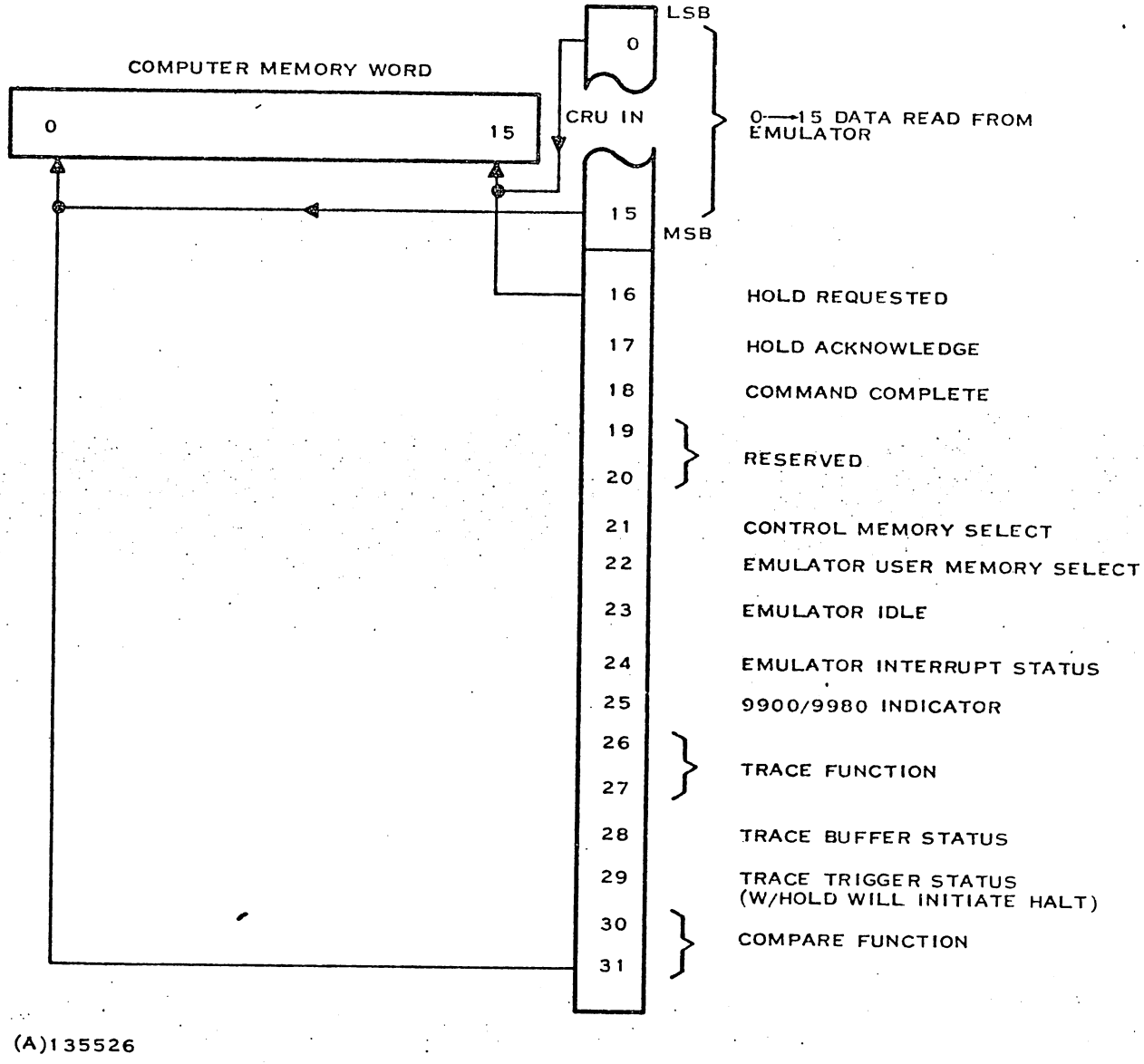


Figure 3-2. Emulator to CRU Data/Status Word Format

3.3.6 HALT SEQUENCE. A halt sequence may be initiated by any of four operations:

- The trace buffer filled in the continuous trace mode
- An external trigger
- An internal breakpoint condition
- The trace module halt trigger (when the trace module is connected to the emulator).

The emulator reacts to a halt sequence in two ways:

- If interrupts are enabled, the host system receives an interrupt from the emulator.
- If automatic hold is selected, the emulator exerts the hold request and the target system will acknowledge hold after completing the current instruction.

3.3.7 TRACE. The emulator can store information, while the target system is executing, in the internal trace memory. The information which is stored may be selected from either the target program counter or from memory addresses on the target system address bus. At the end of the memory cycle, the selected information is stored in the 256 word trace/control memory, relative to the trace counter, and the trace counter is then decremented. The trace buffer full status is indicated when the trace counter counts from one to zero.

Two modes of address tracing are available, continuous and noncontinuous. In continuous trace mode, a halt sequence is initiated when the trace buffer is full. In the noncontinuous mode, processing will continue (overwriting the oldest data) until the emulator is halted. Up to 256 addresses may be placed in the trace buffer.

3.3.8 BREAKPOINT. The breakpoint is a matched comparison of the Compare (CMPR) register and the address bus. The breakpoint address may be selected on program counter, memory address or write only memory address. The current address is compared in the CMPR register, and when a match occurs, the emulator will begin to process a halt sequence.

3.3.9 INTERRUPT. The emulator may generate one maskable interrupt to the host computer when the trace or breakpoint function requests a halt sequence.

3.3.10 TRACE MODULE INTERFACE. The emulator interfaces to the trace module to permit access to the address bus, data bus and other qualifying control signals in order to synchronize trace and emulator operations.

3.3.11 DMA RESTRICTIONS. The emulator memory cannot be accessed by DMA bus-interfaced devices; however, direct memory access may be performed in the target system.

3.3.12 CRU TO EMULATOR DATA/CONTROL DEFINITIONS. The CRU output field is illustrated in figure 3-1. The first 16 bits compose the data word to transfer data and addresses to the emulator. They may be transferred by LDCR instructions while the emulator is in hold.

The second word (field of 16 bits), depicted in figure 3-1 and table 3-1, defines the emulator program control functions and status. Figure 3-2 illustrates the emulator input data word describing the target system and the emulator status word. Table 3-2 defines the emulator input status conditions.

Table 3-1. CRU to Emulator Bit Definitions

Mnemonic	Displacement	No. of Bits	Definitions
E.D	0	16	Data Register
E.H	16	1	Hold 0=Hold request 1=Release hold
E.T	17	1	Trap 0=Force a load trap 1=Force a reset trap
E.M	18	1	Memory Control 1=Move data register to memory address register, then read from the memory location addressed by the contents of the memory address (MA) into the data register 0=Write data register into the location addressed by the contents of MA (with automatic read after write)
E.TC	19	1	Trace Counter Control 0=Move data register to trace counter 1=Move trace counter to data register
E.CR	20	1	Compare Register Control 0=Move data register to compare register 1=Move compare register to data register
E.TM	21	1	Trace/Control Memory Select 0=Address $FE00_{16}$ to $FFFF_{16}$ in the target system 1=Address $FE00_{16}$ to $FFFF_{16}$ in the trace/control memory of the emulator module
E.UM	22	1	User Memory Select 0=Address 0000_{16} to $1FFF_{16}$ in the target system 1=Address 0000_{16} to $1FFF_{16}$ from the emulator user memory (this memory is optional)
E.AH	23	1	Automatic Hold 1=Automatic hold upon breakpoint condition 0=No hold request upon breakpoint condition
E.I	24	1	Interrupt Mask Control 1=Mask interrupts (disable interrupts) 0=Unmask interrupts (enable interrupts)
E.CS	25	1	Trace Clock Select 1=Use every memory cycle for trace clock 0=Use only trace clocks which are externally enabled.

Table 3-1. CRU to Emulator Bit Definitions (Continued)

Mnemonic	Displacement	No. of Bits	Definitions
E.TF	26	2	Trace Function Memory Data 0=Off, Memory Data (MD) to edge connnector 1=Off, MA to edge connector 2=MA on all cycles 3=Program Counter (PC) on instruction fetch cycles
E.TB	28	1	Trace Breakpoint 1=Generate a breakpoint when the trace buffer is full (trace counter goes from 1 to 0) 0=Do not generate a breakpoint
E.XB	29	1	External Breakpoint 1=Enable external breakpoint source 0=Expect internal breakpoint (from the compare function)
E.CF	30	2	Compare Function 0=Off 1=Memory address on write cycle (memory write) 2=Memory address on all cycles 3=PC on instruction fetches

3.4 EXAMPLE PROGRAM ROUTINES

This paragraph gives some example routines for the use of the emulator. Five program routines illustrating the basic emulator functions (i.e., run, read, write, break and trace) are presented. Several common subroutines are described, and examples are given of two higher level routines which use these subroutines. (CRU bit definitions appear as variable mnemonics in tables 3-1 and 3-2.)

In the following routines, the REF variable ECRU contains the CRU base address of the emulator. To implement these routines the following statements must be made:

```
      DEF   ECRU

ECRU  DATA  >140
```

3.4.1 TGTRUN, RUN THE TARGET SYSTEM. The TGTRUN routine first initializes the emulator module to:

- Ignore interrupts
- Expect an internal breakpoint condition
- Exert hold automatically when a breakpoint occurs.

It next releases hold (to let the target system run). It then waits until the target system is automatically returned to hold.

Table 3-2. Emulator to CRU Bit Definitions

Emulator Input Bits			
Mnemonic	Displacement	No. of Bits	Definitions
E.D	0	16	Data Register
E.H	16	1	Hold 0=Hold has been requested 1=Hold has not been requested
E.HA	17	1	Hold Acknowledged 0=Target system is in hold 1=Target system is running (not in hold)
E.CC	18	1	Command Completed 0=Command has been completed 1=Command not complete yet
	19	1	Reserved.
E.CE	20	1	Clock Error 0=No clock error (everything is OK) 1=Target system clock error
E.TM	21	1	Trace/Control memory selected 0=Address $FE00_{16}$ to $FFFE_{16}$ in the target system 1=Address $FE00_{16}$ to $FFFE_{16}$ in the trace control memory of the emulator module
E.UM	22	1	Emulator User Memory Selected 0=Address 0000_{16} to 4000_{16} in the target system 1=Address 0000_{16} to 4000_{16} from the emulator user memory (this memory is optional)
E.ID	23	1	Emulator Idle 1=Target system is in an idle state 0=Target system is not in an idle state
E.I	24	1	Interrupt Status 1=Interrupt is pending 0=Interrupt not pending
E.MT	25	1	Microprocessor Type 1=Target system is a TMS 9900 0=Target system is a TMS 9980A
E.TF	26	2	Trace Function (see above)
E.BS	28	1	Trace Buffer Status 1=Trace buffer is full 0=Trace buffer is not full

Table 3-2. Emulator to CRU Bit Definitions (Continued)

Mnemonic	Emulator Input Bits Displacement	No. of Bits	Definitions
E.TS	29	1	Trace Trigger Status 1=Trace has generated a breakpoint 0=Breakpoint not generated by the trace
E.CF	30	2	Compare Function 0=Off 1=Memory address on write cycles (memory write) 2=Memory address on all cycles 3=PC on instruction fetches

This routine is called with a branch and link instruction (BL @TGTRUN). No arguments are expected; however, the emulator should be initialized to breakpoint on some condition (otherwise, hold will never be asserted and this routine will continue looping until a manual restart). The contents of register 12 are destroyed.

```
        IDT     'TGTRUN'
        DEF     TGTRUN
        REF     ECRU        Emulator CRU base address
```

Emulator Output Bits:

```
E.H.    EQU     16      1   0=Hold request
                            1=Release hold
E.AH    EQU     23      1   0=Automatic hold
E.I     EQU     24      1   0=Unmask interrupt
                            1=Mask interrupt
E.XB    EQU     29      1   0=Internal
                            1=External breakpoint
```

Emulator Input Bits:

```
E.HA    EQU     17      1   0=Hold acknowledged
```

Program Routine:

```
TRUN    EQU     $
        MOV     @ECRU,R12   Initialize the CRU base address
        SBO     E.I         Mask out interrupts
        SBZ     E.XB        Expect an internal breakpoint
```

```
        SBO     E.AH        Do an automatic hold when a breakpoint
                            occurs

        SBO     E.H         Release hold. (Start the target system)

LP1     TB      E.HA        Wait until the target system is auto-
        JEQ     LP1         matically placed in hold

        RT                  Return to caller

        END
```

3.4.2 TREAD, TARGET MEMORY READ. TREAD reads one word (16 bits) from a specified memory location of the target system.

TREAD is called with a branch and link instruction (BL @TREAD). The target memory address to read is expected to be in register 1. The value read at that memory address is returned in register 0. The contents of register 0 and 12 are destroyed.

```
        IDT     'TREAD'
        DEF     TREAD
        REF     ECRU        Emulator CRU base address
```

Emulator Output Bits:

```
E.D     EQU     0       16      Data register
E.M     EQU     18      1       0=Write data to memory
                                1=Read from target memory
```

Emulator Input Bits:

```
E.CC    EQU     18      1       0=Command completed
```

Program Routine:

```
TREAD   EQU     $
        MOV     @ECRU,R12       Initialize the CRU base address.
        LDCR    R1,0            Memory address to the data register
        SBO     E.M             Move the data register to memory address
                                (MA). This simultaneously moves MA into
                                the data register
```

```
LP1     TB      E.CC
        JEQ     LP1             Wait for command completion
        STCR    R0,0            Return the value read in register 0
        RT                      Return to caller
        END
```

3.4.3 TWRIT, TARGET MEMORY WRITE. The TWRIT routine writes one word (16 bits) into a specified memory location of the target system.

TWRIT is called with a branch and link instruction (BL @TWRIT). Two arguments are expected:

- Register 1 to contain the address to write into
- Register 0 to contain the value to write.

Registers 1 and 12 are destroyed by this routine.

```
        IDT     'TWRIT'
        DEF     TWRIT
        REF     ECRU            Emulator CRU base address
```

Emulator Output Bits:

```
E.D     EQU     0       16      Data register
E.M.    EQU     18      1       0=Write data to memory
                                1=Read from target memory
```

Emulator Input Bits:

```
E.CC    EQU     18      1       0=Command completed
```

Program Routine:

```
TWRIT   EQU     $
        MOV     @ECRU,R12       Initialize the CRU base address
        LDCR    R1,0            Memory address to the data register
        SBO     E.M             Move the data register to MA
LP1     TB      E.CC            Wait for command to complete
        JEQ     LP1
        LDCR    R0,0            Value to write into the data register
```

```
        SBZ     E.M             Write the data register at MA
LP2     TB      E.CC            Wait for command to complete
        JEQ     LP2
        STCR    R1,0            Read the emulator data register
        C       R1,R0           Was target memory written correctly?
        JEQ     RETRN           NE indicates a target memory write
                                failure
        JMP     $               The user may want to insert an error
                                message routine here
RETRN   RT                      Return to caller
        END
```

3.4.4 EBRK, SET EMULATOR BREAKPOINT. EBRK initializes the compare register and compare function of the emulator module to provide the breakpoint condition.

EBRK is called with a branch and link instruction (BL @EBRK). Two arguments are expected:

- Register 0, the value to place in the compare register.
- Register 1, the mode to which to set the compare function.

Legal modes of the compare function:

0 = Off

1 = Memory address on write cycles (memory write)

2 = Memory address on all cycles

3 = Program counter on instruction fetches

If the compare function is not off, and the value in the compare register is equal to MA or PC as selected, then a breakpoint will be generated by the emulator. Register 12 is destroyed by this routine.

```
        IDT     'EBRK'
        DEF     EBRK
        REF     ECRU            Emulator CRU base address.
```

Emulator Output Bits:

E.D	EQU	0	16	Data register
E.CR	EQU	20	1	0=Move data register to compare register 1=Move compare register to data register
E.CF	EQU	30	2	Compare function

Emulator Input Bits:

E.CC	EQU	18	1	0=Command completed

Program Routine:

```
EBRK    EQU     $
        MOV     @ECRU,R12       Initialize the CRU base register
        LDCR    R0,0            Desired compare register value to data
                                register
        SBZ     E.CR            Move from the data register to the com-
                                pare register
LP1     TB      E.CC
        JEQ     LP1             Wait for command completion
        AI      R12,E.CF*2      Point R12 at the compare function
                                bits
        SWPB    R1              Prepare to fetch left byte
        LDCR    R1,2            Set the desired compare function mode
        RT                      Return to caller
        END
```

3.4.5 ETRC, SETUP EMULATOR TO TRACE. ETRC initializes the trace counter and trace function to a useful state of the emulator module.

ETRC is called with a branch and link instruction (BL @ETRC). Two arguments are expected:

- Register 0, the value to place in the trace counter.
- Register 1, the mode to which to set the trace function, right-justified, within the register.

Legal modes of the trace function are:

0 = Off, memory data (MD) to the trace module

1 = Off, MA to the trace module

2 = Memory address on all cycles

3 = Program counter on instruction fetches

ETRC sets E.TB to 1 (generates a breakpoint when the trace buffer is full). If N equals the trace counter initialized by register 0, when hold is released on the target system, N samples will be placed in the trace buffer and then a breakpoint will be generated. E.CS is set to 1 so that a sample will be taken on each appropriate memory cycle. Register 12 is destroyed by this routine.

```
        IDT     'ETRC'
        DEF     ETRC
        REF     ECRU            Emulator CRU base address
```

Emulator Output Bits:

```
E.D.    EQU     0       16      Data register
E.TC    EQU     19      1       0=Move data register to trace counter
                                1=Move trace counter to data register
E.CS    EQU     25      1       0=External clock selected
                                1=Clock every memory cycle
E.TF    EQU     26      2       Trace function:
                                0=Off, MD to edge connector
                                1=Off, MA to edge connector
                                2=MA on all cycles
                                3=PC on instruction fetch cycles
E.TB    EQU     28      1       0=No breakpoint when trace buffer is full
                                1=Breakpoint when trace buffer is full
```

Emulator Input Bits:

```
E.CC    EQU     18      1       0=Command completed
```

Program Routine:

```
ETRC    EQU     $
        MOV     @ECRU,R12       Initialize the CRU base address
        SBO     E.TB            Breakpoint when the trace buffer is
                                full
```

```
        SBO     E.CS            Sample on each appropriate memory
                                cycle
        LDCR    R0,0            Desired trace counter value to data
                                register
        SBZ     E.TC            Move from the data register to the
                                trace counter
LP1     TB      E.CC            Wait for command completion
        JEQ     LP1
        AI      R12,E.TF*2      Point R12 at the trace function
                                bits
        SWPB    R1              Prepare to fetch left byte
        LDCR    R1,2            Set the desired trace function mode
        RT                      Return to caller
        END
```

3.4.6 SUBROUTINES. Two subroutines, TRWBLK and SAVRST, will now be presented for the more complex example routines, GETREG and SETREG, which follow in 3.4.7 and 3.4.8.

3.4.6.1 TRWBLK, Target Memory Block Read/Write. TRWBLK defines two subroutines to transfer a block of data into or out of the target systems memory.

1. TRBLK transfers a block of memory from the target system into the host system (Target Read).
2. TWBLK transfers a block of memory from the host system into the target system (Target Write).

Both TRBLK and TWBLK are called with a branch and link instruction immediately followed by three inline arguments.

```
BL @TRBLK                or    BL @TWBLK
    DATA <T.ADDRS>             DATA <T.ADDRS>
    DATA <H.ADDRS>             DATA <H.ADDRS>
    DATA <W.COUNT>             DATA <W.COUNT>
```

T.ADDRS = First target memory address to read or write

H.ADDRS = First host memory address to write or read

W.COUNT = Number of words to transfer.

```
        IDT     'TRWBLK'
        DEF     TRBLK,TWBLK
        REF     TREAD,TWRIT     Target memory read and write routines
```

Transfer a block from the target system into the host system:

```
TRBLK   EQU     $
        MOV     *R11+,R1        First target memory address to read
        MOV     *R11+,R2        Pointer into the host system
        MOV     *R11+,R3        Number of words to read
        MOV     R11,R4          Save the return address
LP2     BL      @TREAD          Read target memory
*                               *R1, R2, R3, and R4 are not changed by
                                TREAD
        INCT    R1              Next target address
        MOV     R0,*R2+         Store the target memory value
        DEC     R3              More to transfer?
        JNE     LP2
```

Return to Caller:

```
        MOV     R4,R11          Restore the return address
        RT
```

Transfer a block from the host system into the target system:

```
TWBLK   EQU     $
        MOV     *R11+,R4        First target memory address to write
        MOV     *R11+,R2        Pointer to host memory to read
        MOV     *R11+,R3        Number of values to transfer
        MOV     R11,R5          Save the return address

LP3     MOV     *R2+,R0         Value to restore
        MOV     R4,R1           Target address to write into
```

```
        BL      @TWRIT          Write one word of memory
        INCT    R4              Next target address
        DEC     R3              More to transfer?
        JNE     LP3
```

Return to Caller:

```
        MOV     R5,R11          Restore the return address
        RT
        END
```

3.4.6.2 **SAVRST, Save and Restore Trace Memory.** Save and Restore Trace Memory are examples of two subroutines to save (TSAVE) enough of the trace memory to allow tampering with the trace/control memory. The restore routine (TRESTR) restores the trace memory saved. Both TSAVE and TRESTR are called with a branch and link instruction (BL @TSAVE or BL @TRESTR). There are no arguments.

```
        DEF     TSAVE,TRESTR
        REF     ECRU            CRU base address of the emulator
        REF     TRBLK,TWBLK     Target memory read and write routines
```

Emulator Output Bits:

```
E.TM    EQU     21      1       Trace/control memory select
                                0=address FE00(16) to FFFE(16) in the
                                target system
                                1=address FE00(16) to FFFE(16) in trace
                                control memory
```

Emulator Input Bits:

```
E.TM    EQU     21      1       Trace/control memory selected
```

Save the state of the emulator which will be tampered with:

```
TSAVE   EQU     $
        MOV     R11,R4          Save the return address
        MOV     @ECRU,R12       Initialize the CRU base address
```

Save the E.TM selected by the user:

```
        AI      R12,E.TM*2
        STCR    @SAVTM,1
```

Select trace/control memory

```
        SBO     0               SBO E.TM (except R12 is at E.TM)
```

Save $FFF6_{16}$ through $FFFE_{16}$ of the emulator trace control:

```
        BL      @TRBLK
        DATA    >FFF6           First target memory address to save
        DATA    SAVMEM          Pointer to save area in the host memory
        DATA    6               Number of values to save
```

Return to Caller:

```
        MOV     R4,R11          Restore the return address
        RT
```

Restore the part of the emulator module tampered with:

```
TESTR   EQU     $
        MOV     R11,R5          Save the return address
```

Restore $FFF6_{16}$ through $FFFE_{16}$ of the emulator trace/control memory:

```
        BL      @TWBLK
        DATA    >FFF6           First target memory address to restore
        DATA    SAVMEM          Pointer to array of saved memory
        DATA    6               Number of values to restore
```

Restore E.TM selected by the user:

```
        MOV     @ECRU,R12       CRU base address of the emulator
        AI      R12,E.TM*2
        LDCR    @SAVTM,1
```

Return to Caller:

```
        MOV     R5,R11          Restore the return address
        RT
```

```
SAVTM   BSS   2          Storage of E.TM selected
SAVMEM  BSS   6*2
        END
```

3.4.7 GETREG, GET TARGET SYSTEMS CPU REGISTERS. GETREG is a routine to obtain the workspace pointer (WP), program counter (PC), and status register (ST) of the target system.

The target system must be in the hold state. The CPU registers are obtained by forcing a load trap on the target system. The trap deposits the WP, PC, and ST in the workspace of the trap. The WP, PC, and ST are then moved out of the trap workspace into the host system. The trace/control memory used for the trap is preserved by saving and then restoring the six words of memory used. GETREG is called with a branch and link instruction (BL @GETREG). No arguments are expected. The WP, PC, and ST of the target system are stored in the variable TWP, TPC, and TST, respectively. Register 12 is destroyed.

```
        IDT   'GETREG'
        DEF   GETREG,TWP,TPC,TST
        REF   ECRU              Emulator CRU base address
        REF   TSAVE,TRESTR      Emulator state save and restore routines
        REF   EBRK,TRUN         Set a breakpoint, run the target system
        REF   TRBLK,TWBLK       Target memory read and write routines
```

Emulator Output Bits:

```
E.T     EQU   17      TRAP      0=Force a load trap
```

```
GETREG
        MOV   R11,R7            Save the return address
        BL    @TSAVE            Prepare for using the trace/control
                                memory
```

Put into Control Memory

	(Address)	(Contents)
	>FFFA	>10FF (JMP $)
	>FFFC	>FFDA (Workspace pointer)
	>FFFE	>FFFA (PC pointer)

```
        BL    @TWBLK            Set up for the load trap
        DATA  >FFFA             Target address to write into
        DATA  LOADPG            Host address to read from
        DATA  3                 Number of words to transfer
```

```
            LI      RO,>FFFA
            LI      R1,3
            BL      @EBRK       Setup for a breakpoint when PC = FFFA16
            SBZ     E.T         Force a load trap when hold is released
            BL      @TRUN       Perform the load trap and breakpoint
```

The target system WP, PC, and ST registers are now in the trace/control memory at locations $FFF4_{16}$, $FFF6_{16}$, and $FFF8_{16}$

```
            BL      @TRBLK      Read the WP, PC, and ST registers
            DATA    >FFF4       Target address to read from
            DATA    TWP         Host address to write into
            DATA    3           Number of words to transfer
            BL      @TRESTR     Restore the emulator to its original state
Return to Caller:
            MOV     R7,R11      Restore the return address
            RT
TWP         BSS     2           Target system workspace pointer
TPC         BSS     2           Target system program counter
TST         BSS     2           Target system status register
LOADPB      EQU     $
            JMP     $           Just spin after load trap
            DATA    >FFDA       Load trap workspace address
            DATA    >FFFA       Load trap program counter
            END
```

3.4.8 SETREG, SET TARGET CPU REGISTERS. SETREG is a routine to set the workspace pointer, program counter, and status register of the target system.

The target system has been load-trapped by the GETREG routine (for example). The desired WP, PC and ST registers are placed back into the workspace used for the load trap. Performing a RTWP (return from trap) will then move the target CPU registers back into the microprocessor. The trace/control memory used for the trap is preserved by saving and then restoring the six

words of memory used. SETREG is called with a branch and link instruction (BL @SETREG). No arguments are expected. The desired WP, PC, and ST of the target system must have been stored in the variables TWP, TPC, and TST, respectively.

```
        IDT     'SETREG'
        DEF     SETREG
        REF     ECRU            Emulator CRU base address
        REF     TSAVE,TRESTR    Emulator save and restore routines
        REF     EBRK,TRUN       Set a breakpoint. Run the target system
        REF     TWRIT,TWBLK     Target memory write routines
        REF     TWP,TPC,TST     New target CPU register values
SETREG
        MOV     R11,R7          Save the return address
        BL      @TSAVE          Prepare to use the trace/control memory
```

Move the desired CPU register back into the trap workspace.

```
        BL      @TWBLK          Restore the target system CPU registers
        DATA    >FFF4
        DATA    TWP
        DATA    3
```

Change the contents of $FFFA_{16}$ to an RTWP instruction.

```
        LI      R0,>0380        RTWP opcode
        LI      R1,>FFFA
        BL      @TWRIT
        LI      R0,>FFFA
        LI      R1,3
        BL      @EBRK           Set up for a breakpoint when PC = FFFA16
        BL      @TRUN           Perform the RTWP, but breakpoint imme-
                                diately.
```

The target system WP, PC, and ST registers have now been set into the microprocessor.

```
        BL      @TRESTR         Restore the emulator to its original state.
```

Return to Caller:

```
        MOV     R7,R11          Restore the return address

        RT
        END
```

SECTION IV

OPERATION

4.1 GENERAL PROCEDURES

The general operating procedures for the emulator and buffer modules are equivalent to program execution on the host sytem. That is, the emulator is connected to the target system, the program is loaded through the 733 data terminal from cassette or floppy disk storage, and the program is executed by means of the host 990 Computer control.

Once the target system program and hardware have been debugged and proved, the software can be read out of the target system and stored by the host on diskette or cassette. The PROM program utility (TXPROM, reference TXDS Utilities Manual in Preface), or other available PROM program may then be employed to create custom ROMs for the target system from commercially available PROMs or EPROMs.

4.2 TMS 9900 BUFFER/TARGET MICROPROCESSOR INTERFACE REQUIREMENTS

The target microprocessor to TMS 9900 buffer module electrical ratings and operational and timing requirements are defined in tables 4-1 through 4-5 and figures 4-1 and 4-2.

Table 4-1. Maximum Operational Ratings

(ABSOLUTE MAXIMUM RATINGS OVER OPERATING FREE-AIR TEMPERATURE RANGE, UNLESS OTHERWISE NOTED)*

Supply voltage, V_{CC} (see Note 1)	−0.3 to 20V**
Supply voltage, V_{DD} (see Note 1)	−0.3 to 20V**
Supply voltage, V_{SS} (see Note 1)	−0.3 to 20V**
All input voltages (see Note 1) (except clocks)	−0.3 to 5.5V
Clock input voltage	−0.3 to 20V
Output voltage (with respect to V_{SS})	−0 V to 5V
Operating free-air temperature range	0°C to 50° C
Storage temperature range	−55°C to 150° C

*Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the "Recommended Operating Conditions" section of this specification is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTE 1: Under absolute maximum ratings, voltage values are with respect to the most negative supply, V_{BB} (substrate), unless otherwise noted. Throughout the remainder of this section, voltage values are with respect to V_{SS}.

**All power supplied by emulator module

Table 4-2. TMS 9900 Recommended Operating Conditions

	MIN	NOM	MAX	UNIT
Supply voltage, V_{BB}*	−5.25	−5	−4.75	V
Supply voltage, V_{CC}*	4.75	5	5.25	V
Supply voltage, V_{DD}*	11.4	12	12.6	V
Supply voltage, V_{SS}*		0		V
High-level input voltage, V_{IH} (all inputs except clocks)	2.0	2.4		V
High-level clock input voltage, $V_{IH}(\phi)$	10.0	12.0		V
Low-level input voltage, V_{IL} (all inputs except clocks)			0.4	V
Low-level clock input voltage, $V_{IL}(\phi)$		0.3	0.6	V
Operating free-air temperature, T_A	0		50	°C

*All power supplied by emulator module.

Table 4-3. TMS 9900 Electrical Requirements Over Recommended Operating Conditions (Unless Otherwise Noted)

	Parameter		Test Conditions	Min	Typ†	Max	Unit
I_I	Input current	Data bus during DBIN	$V_I = V_{SS}$ to V_{CC}		±75	100	μA
		$\overline{WE}$, $\overline{MEMEN}$, DBIN during HOLDA	$V_I = V_{SS}$ to V_{CC}		±75	100	
		Clock	V_I = -1 V to 13.6 V		±75	100	
		Any other inputs	$V_I = V_{SS}$ to V_{CC}			1	mA
V_{OH}	High-level output voltage		I_O = –0.4 mA	2.4	3.4		V
V_{OL}	Low-level output voltage		I_O = 3.2 mA		0.4	0.5	V
I_{BB}	Supply current from V_{BB}*				0		mA
I_{CC}	Supply current from V_{CC}*				0		mA
I_{DD}	Supply current from V_{DD}*				0		mA
C_i	Input capacitance (any inputs except clock and data bus)		f = 1 MHz, unmeasured pins at V_{SS}		30	40	pF
$C_i(\phi 1)$	Clock 1 input capacitance		f = 1 MHz, unmeasured pins at V_{SS}		100	150	pF
$C_i(\phi 2)$	Clock 2 input capacitance		f = 1 MHz, unmeasured pins at V_{SS}		200	250	pF
$C_i(\phi 3)$	Clock 3 input capacitance		f = 1 MHz, unmeasured pins at V_{SS}		100	150	pF
$C_i(\phi 4)$	Clock 4 input capacitance		f = 1 MHz, unmeasured pins at V_{SS}		100	150	pF
C_{DB}	Data bus capacitance		f = 1 MHz, unmeasured pins at V_{SS}		40	50	pF

†All typical values are at T_A = 25°C and nominal voltages.
*All power supplied by emulator module

Table 4-4. TMS 9900 Microprocessor Timing Interface Requirements (Over Full Range of Recommended Operating Conditions)

	Parameter	Min	Nom	Max	Unit
$t_{c(\phi)}$	Clock cycle time	.310	0.333	0.5	μs
$t_{r(\phi)}$	Clock rise time	10	12		ns
$t_{f(\phi)}$	Clock fall time	10	12		ns
$t_{w(\phi)}$	Pulse width, any clock high	40	45	100	ns
$t_{d(\phi 1L\text{-}\phi 2H)}$	Delay time, clock 1 low to clock 2 high (time between clock pulses)	0	5		ns
$t_{d(\phi 2L\text{-}\phi 3H)}$	Delay time, clock 2 low to clock 3 high (time between clock pulses)	0	5		ns
$t_{d(\phi 3L\text{-}\phi 4H)}$	Delay time, clock 3 low to clock 4 high (time between clock pulses)	0	5		ns
$t_{d(\phi 4L\text{-}\phi 1H)}$	Delay time, clock 4 low to clock 1 high (time between clock pulses)	0	5		ns
$t_{d(\phi 1H\text{-}\phi 2H)}$	Delay time, clock 1 high to clock 2 high (time between leading edges)	70	80		ns
$t_{d(\phi 2H\text{-}\phi 3H)}$	Delay time, clock 2 high to clock 3 high (time between leading edges)	70	80		ns
$t_{d(\phi 3H\text{-}\phi 4H)}$	Delay time, clock 3 high to clock 4 high (time between leading edges)	70	80		ns
$t_{d(\phi 4H\text{-}\phi 1H)}$	Delay time, clock 4 high to clock 1 high (time between leading edges)	70	80		ns
t_{su}	Data or control setup time before clock 1	40	50		ns
t_h	Data hold time after clock 1	10	20		ns

See figures 4-1 and 4-2.

Table 4-5. TMS 9900 Microprocessor Switching Interface Characteristics (Over Full Range of Recommended Operating Conditions)

	Parameter	Test Conditions	Min	Typ	Max	Unit
t_{PLH} or t_{PHL}	Propagation delay time, clocks to outputs	C_L = 200 pF	25		40	ns
t_{PLN} or t_{PHL} (MEMEN-)			40			ns

See figure 4-2.

4.3 TMS 9980A BUFFER ELECTRICAL CHARACTERISTICS

The characteristics of the TMS 9980A buffer input and output signals are given in tables 4-6 and 4-7. The input load or output driver is given for all buffer/target system lines. Table 4-8 gives information on the external clock source requirements as well as setup and hold time alterations imposed by the buffer on input timing. Finally, table 4-9 shows the propagation delays associated with buffer outputs.

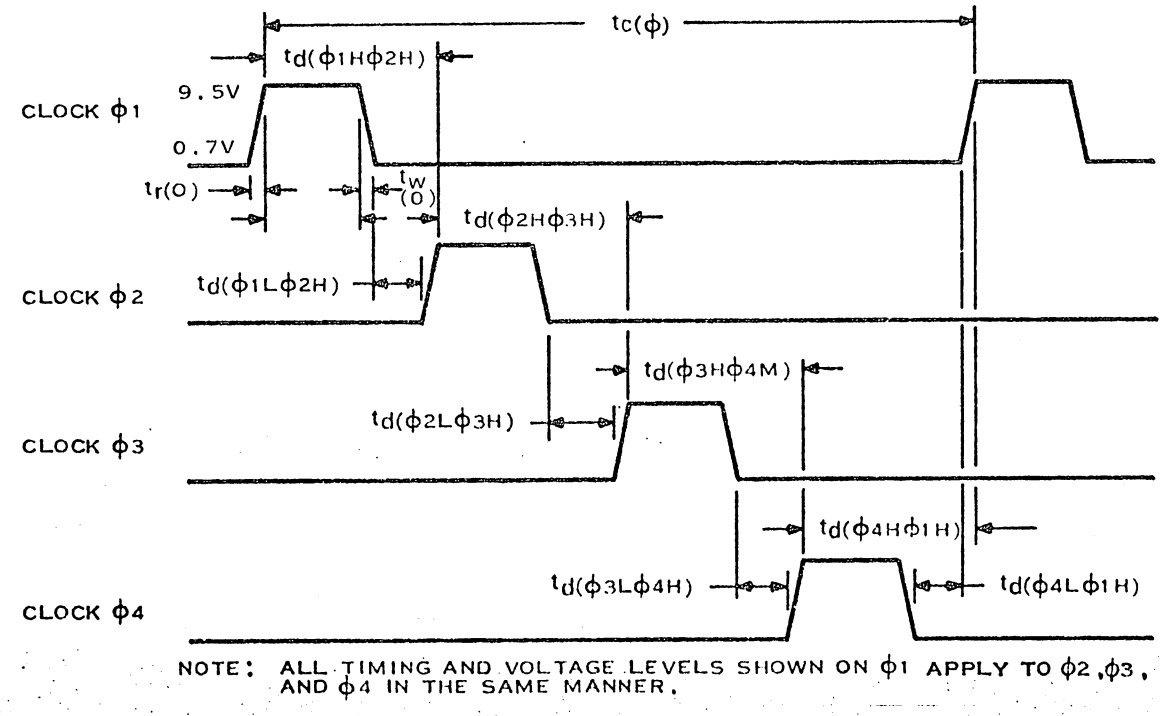


Figure 4-1. TMS 9900 Microprocessor Clock Timing

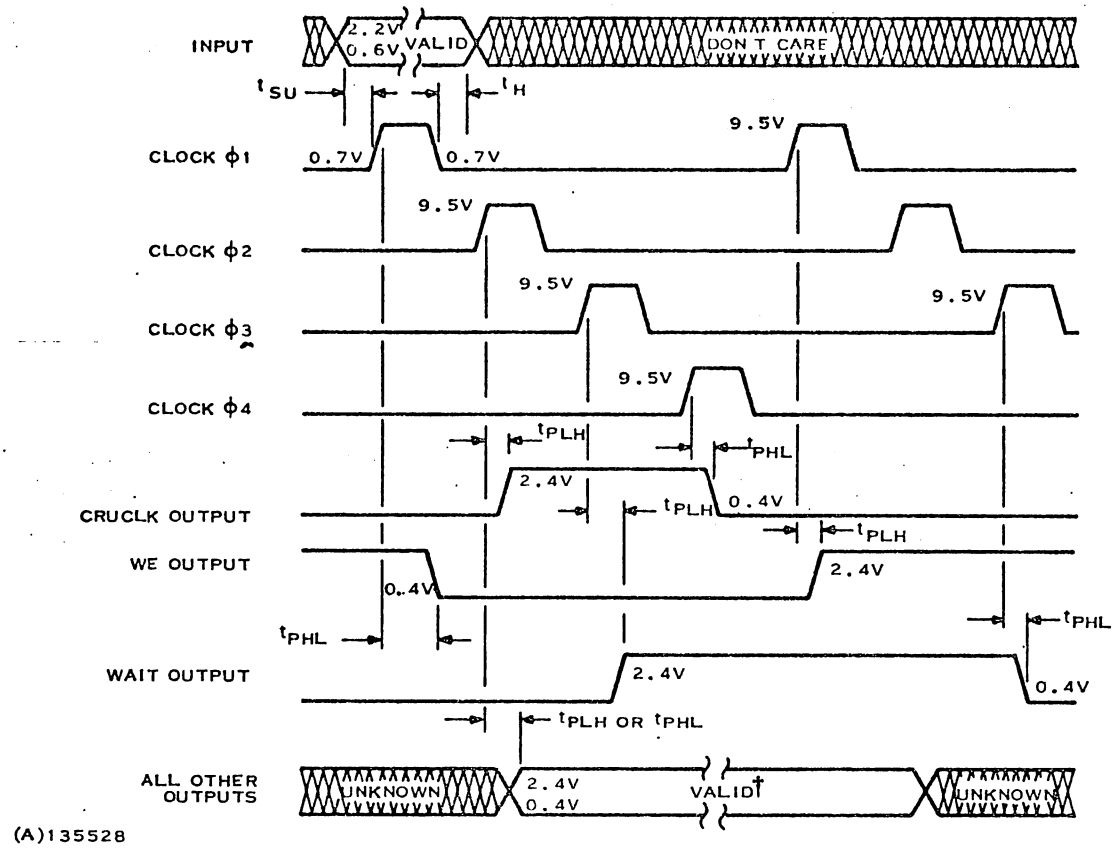


Figure 4-2. TMS 9900 Microprocessor Signal Timing

Table 4-6. TMS 9980A Buffer Input Signal Loads

Input	Load
D0 - D7	74LS243
CKIN	TMS 9980A + 74LS04
CRUIN	74LS32
INT0-2	74LS32
HOLD–	74LS04
READY	74LS125

Table 4-7. TMS 9980A Buffer Output Drivers

Output	Type
A0 - A13/CRUOUT	74LS244
D0 - D7	74LS243
CKOUT (03)	74LS244
DBIN	74LS241
MEMEN–	74S241
WE–	74S241
CRUCLK	74LS244
HOLDA	74LS08
IAQ	74LS244

Table 4-8. TMS 9980A Timing Requirements

Parameter		Min	Nom	Max	Units
Clock frequency, text		6		10	MHZ
Clock cycle time t_c (0)		.4		.666	μsec
External source rise/fall time Tr/Tf			10		nsec
External source high level pulse width T_{wh}		40			nsec
External source low level pulse width T_{wl}		40			nsec
External source high level V_h		2.2			V
External source low level V_l				0.8	V
Setup Time t_{su}	HOLD–		+33	+53	ns
	READY		+12	+21	ns
	D0 - D7		+15	+21	ns
	CRUIN		+17	+25	ns
	IC (0-2)		+32	+48	ns
Setup Time t_h	HOLD–	-53	-33	+0	ns
	READY	-21	-12	+0	ns
	D0 - D7	-21	-15	+0	ns
	CRUIN	-25	-17	+0	ns
	IC (0-2)	-48	-32	+0	ns

+ indicates a quantity to be added to the corresponding TMS 9980A parameter.

- indicates a quantity to be subtracted from the corresponding TMS 9980A parameter.

Table 4-9. TMS 9980A Buffer Switching Characteristics

Parameter	From	To	Test Condition	Min.	Typ.*	Max.*
t_{PLH} or t_{PHL}	03	Address Bus (A0 - 13)			+15	+21
"	"	Data Bus D0 - D7			+15	+21
"	"	Write Enable (WE)			+9	+12
"	"	Data Bus In (DBIN)			+31	+45
"	"	Memory Enable (MEMEN)			+26	+38
"	"	CRU Clock (CRUCLK)			+15	+21
"	"	CRU Data Out (CRUOUT)			+15	+21
"	"	Hold Acknowledge (HOLDA)			+25	+41
"	"	Instruction Acquisition (IAQ)			+15	+21

+ Indicates a quantity to be added to the corresponding TMS 9980A parameter.

* Unit times are in nanoseconds

4.3.1 TMS 9980A BUFFER SWITCHING CHARACTERISTICS. The external signal timing diagram for the TMS 9980A buffer is shown in figure 4-3. The timing of all the inputs and outputs is controlled by the internal 4-phase clock; thus, all timings are based on the width of one phase of the internal clock. This is $1/f_{(CKIN)}$, whether driven or from a crystal. This is also one-fourth frequency of the system, $\frac{1}{4}f$ (system). In the tabular part of figure 4-3, where the parameters and test conditions are shown, the phase time is denoted t_w. All external signals are with reference to $\phi 3$.

4.4 PRECAUTIONS

The following caution notes apply to the interface between emulator and buffer modules to host computer and target systems.

CAUTION

1. Computer power must be "off" before inserting or removing the emulator module, or before mating the emulator to buffer or emulator to trace interface cables.

2. Host computer power must be "on" before the target system power is turned "on". Normally the microprocessor replacement plug may be inserted or withdrawn from the target system while computer power is "on"; however, caution notes 3 through 5 apply to this microprocessor plug interconnection.

3. If host system power is "on", do not permit the microprocessor replacement connector pins to contact any conductive, grounded, or energized surfaces.

4. Target system power must be "off" before unplugging the target system microprocessor chip or connecting the buffer interface.

5. Make certain that the microprocessor replacement plug, from the buffer module, is properly polarized and aligned with the target microprocessor socket upon insertion. The cutoff index corner of the buffer microprocessor replacement plug must be aligned with pin 1 of the target microprocessor socket.

6. When they are used, target system clocks must be properly adjusted for reliable operation. Excessive over voltage may damage the microprocessor in the buffer module.

4.5 CONTROLS AND INDICATORS

The following paragraphs describe the emulator and buffer controls and indicators.

4.5.1 CLOCK SELECT SWITCH. The buffer module has a screwdriver-actuated clock select switch accessible through a hole in the top of the module enclosure. The clock select switch is positioned to either INTERNAL or TARGET clock position to employ either the internal clock or the target clock.

NOTE

Computer reset and restart may be required if the clock selection is changed after program loading.

4.5.2 HOLD ACKNOWLEDGE INDICATOR. The emulator has a red LED indicator between the buffer cable interface connectors and the trace module interface connectors (figure 1-3). This indicator is illuminated whenever the buffer microprocessor is in a hold state.

4.6 PROGRAM LOADING

The programmer can use a 733 ASR Data Terminal (with cassette storage and loading) or a floppy disk or other program I/O device, as dictated by the peripherals employed in the user's system. Software routines are normally supplied on either cassette or diskette media. Refer to the 733 or Floppy Disk Installation and Operation manuals for the applicable peripheral device.

4.7 SPECIAL MAINTENANCE

Since the emulator module plugs directly into the 990 Computer chassis, general maintenance of this card is related to the computer maintenance provisions. The emulator circuit board has no special maintenance provisions. Depot level maintenance for the buffer module and emulator module are covered in their respective maintenance manuals (see Preface).

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
$t_r(\phi 3)$	RISE TIME OF $\phi 3$	C_L = 200pf	3	5	10	nS
$t_f(\phi 3)$	FALL TIME OF $\phi 3$		5	7.5	15	nS
$t_W(\phi 3)$	PULSE WIDTH OF $\phi 3$		t_W-15	t_W-10	t_W+10	nS
t_{SU}	DATA OR CONTROL SETUP TIME *		t_W-30			nS
t_h	DATA HOLD TIME *		$2t_{tW}$+10			nS
$t_{PHL(We)}$	PROPAGATION DELAY TIME WE HIGH TO LOW		t_W-10	t_W	t_W+20	nS
$t_{PLH(We)}$	PROPAGATION DELAY TIME WE LOW TO HIGH		t_W	t_W+10	t_W+30	nS
$t_{PHL(CRUCLK)}$	PROPAGATION DELAY TIME CRUCLK HIGH TO LOW		-20	-10	+10	nS
$t_{PLH(CRUCLK)}$	PROPAGATION DELAY TIME CRUCLK LOW TO HIGH		$2t_W$-10	$2t_W$	$2t_W$+20	nS
t_{OV}	DELAY TIME FROM OUTPUT VALID TO $\phi 3$ LOW		t_W-50	t_W-30		nS
t_{OX}	DELAY TIME FROM OUTPUT INVALID TO $\phi 3$ LOW			t_W-20	t_W	nS

* ALL INPUTS EXCEPT IC0,IC2 MUST BE SYNCHRONIZED TO MEET THESE REQUIREMENTS. IC0,IC2 MAY CHANGE ASYNCHRONOUSLY.

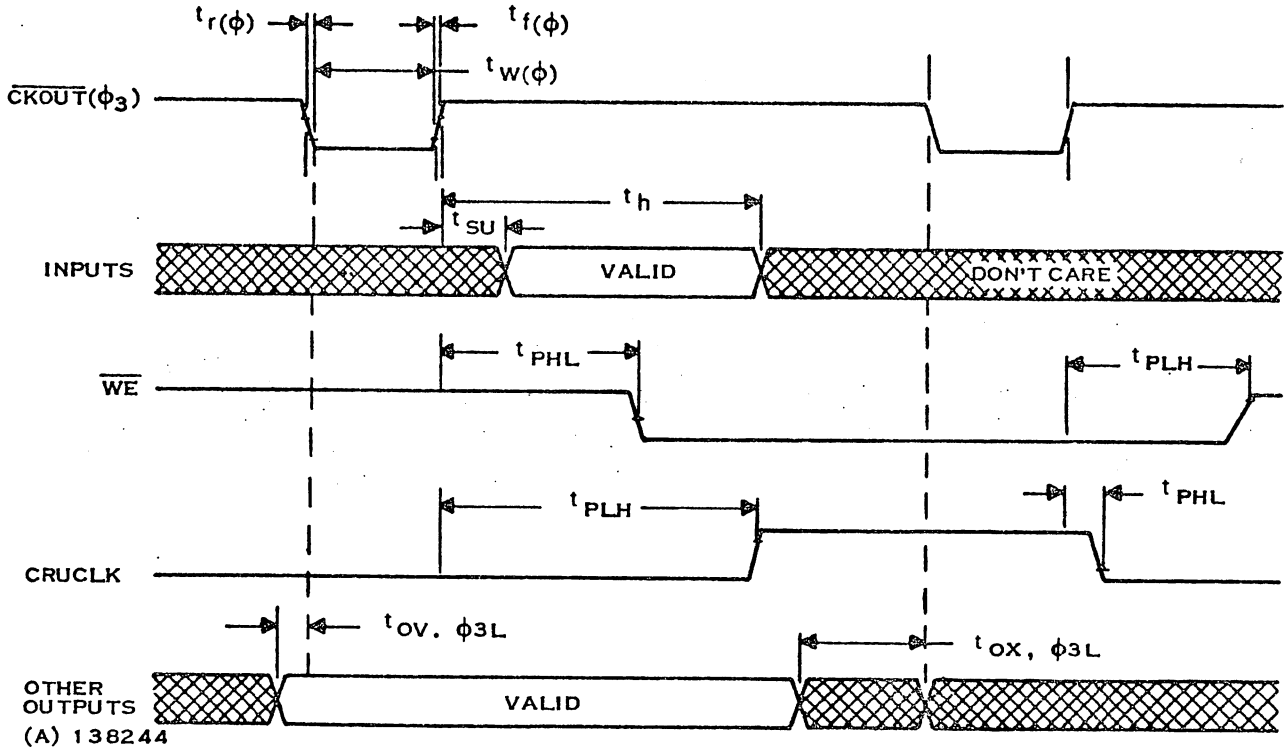


Figure 4-3. TMS 9980A External Signal Timing Diagram

ALPHABETICAL INDEX