TEXAS INSTRUMENTS

Improving Man's Effectiveness Through Electronics

Model 990 Logic State Trace Data Module

Installation And Operation

MANUAL NO. 946241-9701
ORIGINAL ISSUE 1 MAY 1977

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INSERT LATEST CHANGED PAGES DESTROY SUPERSEDED PAGES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 Logic State Trace Data Module Installation and Operation (946241-9701)

Original Issue . . . . . .1 May 1977

Total number of pages in this publication is 56 consisting of the following:

PAGE NO.	CHANGE NO.	PAGE NO.	CHANGE NO.	PAGE NO.	CHANGE NO.
Cover	0				
Effective Pages	0				
iii - vi	0				
1-1 - 1-10	0				
2-1 - 2-10	0				
3-1 - 3-20	0				
4-1 - 4-2	0				
Alphabetical Index Div.	0				
Index-1 - Index-2	0				
User's Response Sheet	0				
Business Reply	0				
Cover Blank	0				
Cover	0				

PREFACE

This manual provides installation procedures, interface requirements, programming information, and operation instructions for the Trace Module of the Texas Instruments AMPL Microprocessor Development Lab. When employed with a host 990 Computer, the Trace Module provides an effective tool for hardware and software development for 9900 series microprocessor prototyping.

The information in this manual is divided into the following sections:

I General Description – Overview description of the Trace Module, including operating characteristics.

II Installation – Instructions for unpacking, inspecting, and installing the Trace Module in a Model 990 Computer chassis.

III Programming – Programming instructions for the Model 990 Computer when utilizing the Trace Module, and a description of how the software controls the operation of the module.

IV Operation – Interface description of the Trace Module and a detailed description of the module's operation.

Additional information related to the Trace Module may be found in the following documents:

Title	Part Number
990 Computer Family Systems Handbook	945250-9701
Model 990 Computer TMS 9900 Microprocessor Assembly Language Programmer's Guide	943441-9701
Model 990/4 Computer System Hardware Reference Manual	945251-9701
Model 990/10 Computer System Hardware Reference Manual	945417-9701
AMPL Microprocessor Prototyping Lab Operation Guide	946244-9701
Model 990 Emulator and Buffer Modules Installation and Operation	946245-9701
Model 990 Logic State Trace Data Module Depot Maintenance	946242-9701

TABLE OF CONTENTS

Paragraph	Title	Page
	SECTION I. GENERAL DESCRIPTION	
1.1	General	1-1
1.2	Purpose of Equipment	1-1
1.3	Trace Module Functional Description	1-4
1.3.1	General	1-4
1.3.2	Trace Module Interfaces	1-4
1.3.3	Target Data Inputs	1-7
1.3.4	Data Latch	1-7
1.3.5	Clock Selection	1-7
1.3.6	Qualifiers	1-7
1.3.7	Trace RAM	1-8
1.3.8	Data Word Recognition (Comparison) Logic	1-8
1.3.9	Event Control Logic	1-8
1.3.10	Event Counter and Delay Counter	1-9
1.3.11	Termination Modes	1-9
1.3.12	Diagnostic Mode	1-10
1.3.13	Multiple Trace Modules	1-10
	SECTION II. INSTALLATION	
2.1	General	2-1
2.2	Unpacking and Inspecting	2-1
2.3	Installation	2-2
2.3.1	Cable Connections	2-2
2.4	Trace Data Probe	2-2
2.5	Checkout	2-7
	SECTION III. PROGRAMMING	
3.1	General	3-1
3.2	Trace Module to CRU Interface	3-1
3.2.1	CRU Output Bit Assignments	3-1
3.2.2	CRU Input Bit Assignments	3-1
3.2.3	Trace Module Interrupts	3-1
3.2.4	Trace Module/Emulator Module Interface	3-1
3.3	Examples of Program Routines	3-1
3.3.1	CRU Bit Definitions	3-9
3.3.2	GSTAT Routine	3-9
3.3.3	SSTAT Routine	3-11
3.3.4	WREG Routine	3-11
3.3.5	RREG Routine	3-14
3.3.6	STRT Routine	3-14
3.3.7	STOP Routine	3-14
3.3.8	RBUF Routine	3-14
3.4	Loading/Reading the Trace RAM	3-14

TABLE OF CONTENTS (Continued)

Paragraph	Title	Page
	SECTION IV. OPERATION	
4.1	General	4-1
4.2	Operating Precautions	4-1
4.3	Controls and Indicators	4-1
4.3.1	*Event* Indication	4-2
4.4	Program Loading	4-2

LIST OF ILLUSTRATIONS

Figure	Title	Page
1-1	Texas Instruments Model 990 Trace Module	1-2
1-2	Typical Development System Configuration	1-3
1-3	Trace Module Functional Block Diagram	1-5
2-1	Standard Computer Chassis Configuration	2-3
2-2	Standalone Trace Module Cabling	2-4
2-3	Trace Module and Emulator Module Interconnecting Cabling	2-5
2-4	Cabling Configuration Using Multiple Trace Modules	2-6
2-5	Trace Data Probe Terminator Box and Leads	2-8
2-6	Data Probe Lead Electrical Equivalent	2-9
3-1	CRU Output Bit Assignments	3-2
3-2	CRU Input Bit Assignments	3-6
3-3	Loading and Reading the Serial Registers	3-10
3-4	GSTAT Sample Routine	3-11
3-5	SSTAT Sample Routine	3-12
3-6	WREG Sample Routine	3-13
3-7	RREG Sample Routine	3-15
3-8	STRT Routine	3-16
3-9	STOP Routine	3-16
3-10	RBUF Sample Routine	3-17
3-11	Loading the Trace RAM	3-18
3-12	Reading the Trace RAM	3-19
4-1	Trace Module LEDs	4-1

LIST OF TABLES

Table	Title	Page
2-1	Trace Module Assembly and Accessories	2-1
3-1	CRU Output Bit Description	3-3
3-2	CRU Input Bit Description	3-7
3-3	Trace/Emulator Control Interface	3-8
3-4	Trace Module CRU Bit Definitions	3-10

SECTION I

GENERAL DESCRIPTION

1.1 GENERAL

This manual provides installation, operating instructions and programming information required to install and check out the Texas Instruments Model 990 Logic-State Trace Data Module (trace module) (figure 1-1), Part Number 949910.

This section contains a physical and functional description of the trace module and describes the use of the trace module within the Texas Instruments AMPL Advanced Microprocessor Prototyping Lab.

1.2 PURPOSE OF EQUIPMENT

The trace module is a programmable general-purpose logic-state recorder with special features to aid in designing and debugging microprocessor prototype systems. The trace module is a full-size circuit board that is installed in a CRU slot in a host 990 computer chassis. The computer's interactive software language and computer console replace the complex controls and indicators typically found on logic-state recorders, thereby increasing its tracing capabilities while reducing operational complexity. The trace module interfaces with the prototype system through either a 20-probe data cable or direct data cable interface with the emulator module (figures 1-2 and 1-3).

The trace module is equipped with a 256 by 20-bit memory which provides temporary storage of selected trace data from the user's target system. When operated synchronously with the emulator module in a prototyping system, a 256-word RAM in the emulator module provides a similar tracing function for the memory addresses associated with the data collected by the trace module.

The trace module incorporates a maskable set of four qualifiers (conditions which must be present before each 20-bit data sample is stored in memory) and programmable logic to permit the user program to define the conditions for halting a trace operation and issuing an interrupt to the host computer (interrupts may also be masked). The module may be halted when the memory is full or permitted to over-record until halted under software control.

The sampling rate of the trace module is also a user-selected feature. For asynchronous tracing operations, the user program specifies the internal 10 MHz clock which produces a sampling period of 100 nanoseconds. For synchronous trace operations, the user program specifies the use of the external clock. In this case, the external clock line is connected to a signal in the user's prototype system which toggles simultaneously with state changes in the sampled data (maximum frequency of 10 MHz).

In order to accommodate tracing of signals which change at a rate faster than the trace module's 100 nanosecond sampling rate, four of the 20 input data lines are connected to special latch circuits. These latches pick up pulses which occur in-between trace module clocks and report each pulse on the next 10 MHz clock. The latches permit recognition of pulse-widths down to approximately 10 nanoseconds.

1.3 TRACE MODULE FUNCTIONAL DESCRIPTION

1.3.1 GENERAL. Figure 1-3 is a simplified functional block diagram of the trace module. The trace module allows the user to monitor the operation of a prototype target system, either directly through a trace data probe or through the emulator module, and to sample and record pertinent data patterns and control signals. The logical state of these data patterns may be stored in a trace RAM (Random Access Memory) that is 256 bits long and 20 bits wide. When a user-specified *Event* occurs in the program, the data sampling process is discontinued and the contents of the trace RAM may be examined to determine the state of the prototype that led up to the *Event*.

1.3.2 TRACE MODULE INTERFACES. The trace module is a double-width circuit board (module) that is installed in any full-width slot of a 990 Computer chassis, or in a CRU expansion chassis. When installed in a computer, the trace module plugs into connectors in a backplane of the chassis.

1.3.2.1 CRU Interface. The trace module/CRU interface to the host 990 Computer consists of 32 bits, and the CRU address is determined by the chassis slot location in which the trace module is installed. Twelve of the 32 CRU interface bits perform control functions and each bit is assigned a unique address. The remaining 20 bits are used for data transfers between the trace module and the CRU. These 32 interface bits, as inputs and outputs of the CRU, are described in detail in Section III of this manual.

1.3.2.2 Data Cable Interfaces. The outer edge of the trace module circuit board has two cable connectors, as shown in figure 1-1, to interface with the emulator module, or to be used directly with general-purpose signal probes. The interface cabling connections for installation in the system are described in detail in Section II of this manual.

The data cable connector (P4) furnishes inputs to the trace module from one of two sources: directly from the user's target system via a data probe, or from the emulator module. The data probe consists of a 6-foot ribbon cable with a terminator box at the end of the cable. Twenty-eight color-coded leads extend from the terminator box (figure 2-5) and are equipped with female pins for insertion on male wire-wrap pins, or IC test clips at the target prototype breadboard. Each data line has a 220-ohm series resistor installed in the terminator box to limit ac loading. Voltages at the trace data probe inputs should be limited between +5.5V and -1.0V. When the trace module and the emulator module are used together in the prototyping system, two short cables (about four inches long) interconnect the two modules; one cable for the prewired data inputs and one cable for commands and controls. The emulator is used if it is desired to trace Memory Address (MA) or Memory Data (MD), which are prewired on the emulator module. The emulator is programmed by the host computer to supply either memory address or memory data signals via the data cable interconnect. When the data cable is connected to the emulator module, the 20 bits of data input are:

D0-D15 – 16 parallel bits of Memory Address (MA) or Memory Data (MD)
D16 – T2EVENT (Address Compare)
D17 – T1DBIN (Data Bus In)
D18 – T1IAQ (Instruction Acquisition)
D19 – Logic 0

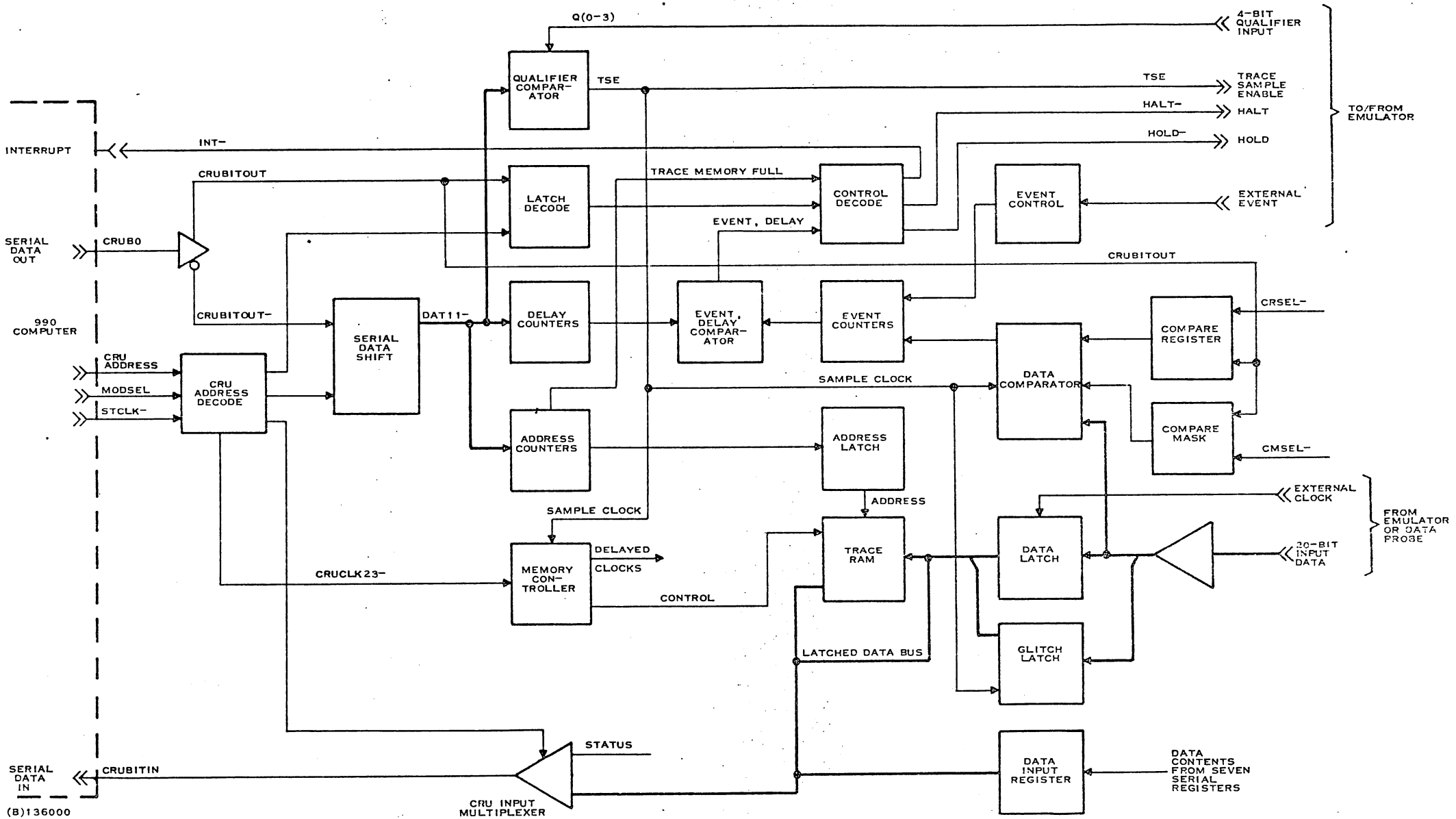


Figure 1-3. Trace Module Functional Block Diagram

1.3.3 TARGET DATA INPUTS. As shown in figure 1-3, 20 data input bits are received from the target system by the trace module via high impedance differential receivers, which minimize loading the target system. Sixteen of the data bits are routed directly to the 256 X 20 trace RAM, and the other four bits are interrogated by a special latch.

1.3.4 DATA LATCH. The purpose of the data latch is to detect narrow pulses that occur between clock pulses, or that change faster than the maximum 10 MHz (100 nanosecond) clock rate at which the trace module operates. The latch is controlled by a latch mode enable signal from the computer CRU which is initiated by the software. When enabled, the latch catches narrow transition spikes, as narrow as approximately 10 nanoseconds, that occur between clock pulses and records this information as one-state transitions or whole samples. The rules for determining what the latched trace will look like for a given input signal are:

1. A constant high input produces a constant high output; a constant low input produces a constant low output.

2. If the input ever goes low while the last trace state was high, then the next trace state will be low.

3. If the input ever goes high while the last trace state was low, then the next trace state will be high.

4. Multiple transitions between any two clocks – the trace will "latch" to the value of the first transition.

1.3.5 CLOCK SELECTION. When the clock changes from low to high, all the data on the 20 data input lines is latched and stored in the trace RAM. There is a choice of two clocks for use by the trace module, internal and external, up to a maximum frequency of 10 MHz. An internal clock may be selected which is derived froma 10 MHz oscillator. The internal clock samples data inputs regularly every 100 nanoseconds, subject to the qualifiers discussed in the next paragraph. The external clock is used to synchronize trace module operations with the target system. When the input data cable is connected to the emulator module, the emulator provides a clock pulse once every reference memory cycle. The emulator clock is used when tracing memory cycles to store Memory Address (MA) or Memory Data (MD), once per memory reference.

1.3.6 QUALIFIERS. Four qualifier bits are input to the comparator functional block to provide clock selectivity. Any or all of these four qualifiers can be masked under software control, and the unmasked qualifiers must equal software programmed states before a trace or sample clock will be accepted. This means that any AND function of any or all of the qualifiers or their complements can be specified as a necessary condition for tracing. The qualifier inputs are part of the data cable so they may be connected to discrete inputs at the target system via the trace data probe, or they may be connected to prewired qualifiers in the emulator module. When the data cable is connected to the emulator module, the qualifiers are prewired as follows:

Q0 – Logic 0 (unused)

Q1 – Instruction Acquisition (IAQ). Indicates that the present memory cycle is an instruction fetch.

Q2 – Data Bus In (DBIN). Distinguishes memory read cycles from memory write cycles. 1 = read; 0 = write.

Q3 – Emulator (Em). Address Compare. The output of the address compare logic of the emulator module. 1 = compare equal; 0 = not equal.

Following is an example of how qualifiers may be used to select only instruction acquisitions from the emulator: when memory reference cycles from the emulator are being traced, the memory cycle clock is received once every memory reference. However, if instructions only are to be traced, then the qualifiers are programmed so that all memory cycles other than instruction acquisition are disregarded, so the qualifiers catch instruction acquisitions only. This means the memory cycle clocks are sifted through, instructions are traced, and the trace RAM is filled with instructions only, rather than all memory data.

1.3.7 TRACE RAM. The trace RAM (Random Access Memory) is a 256 X 20 bit memory having an address counter that decrements each time trace memory is accessed. When the specified qualifiers are true and the selected clock input changes from low to high, all 20 data inputs are sampled and the 20-bit data word is stored in the trace memory. The trace memory is then decremented to the next address in the 256-word memory. As data is being traced, the address counter keeps counting so that new data is continually overwriting old data until conditions are reached as specified by the software program to stop the tracing.

Since the trace memory address (stored in the address counter) is decremented after each cycle, it always points to the oldest entry in a full trace memory. After the trace module has run and stopped, the address counter can be read and written through the computer CRU, and the memory controller can respond to requests from the CRU for trace memory read or write cycles.

When data is traced into address 1 of the trace RAM, the address counter is decremented to 0 and Trace Memory Full (TMF) is generated. The TMF signal can be used to terminate tracing.

1.3.8 DATA WORD RECOGNITION (COMPARISON) LOGIC. The data word recognition logic (comparison logic) consists of two 20-bit programmable registers (compare mask register and compare register) and a data comparator. The compare mask register specifies which of the 20 input data bits are to be ignored, and the compare register specifies the desired state (high or low) of the bits that are unmasked. The data comparator compares the computer-programmed data with the 20-bit input data word upon each qualified clock cycle (sample clock). When the programmed data compares with the input data, a "compare equal" signal, defined as an *Event*, is generated and directed to the event counter. The event counter, together with the delay counter, generates as Event, Delay signal as described in the following paragraphs.

1.3.9 EVENT CONTROL LOGIC. The event control logic dictates how the results of the comparison by the data word recognition logic is to be interpreted by the 16-bit event counter. The event control logic receives an external event input, which may be chosen in lieu of the data recognition logic. When the data input cable is connected to the emulator instead of the data probe, the external event signal is prewired to the breakpoint of the emulator, and the *Event* is defined as an emulator "address compare equal" signal.

Two other bits of event control influence how events are counted by the event counter. The first is useful when an *Event* is present (or absent) for a number of consecutive trace samples. This bit indicates whether to count the *Event* every time it is sampled or only once when the transition is made, i.e., this bit places the event counter in either the "each" or the "edge" mode. The other bit that influences counting determines whether the true or inverted sense of the *Event* is important. A zero in this bit causes the counter to count when *Event* is true, or on its positive-going edge. A logic one in this bit causes the counter to decrement when *Event* is false, or on its negative edge.

To summarize, several options are available for the event control logic to interpret:

- Count *Events* that are high.
- Count *Events* that are low.
- Count every *Event* that occurs.
- Count transitions (edges of *Events*).
- Count *Events* from the word recognizer.
- Count *Events* from the external input.

1.3.10 EVENT COUNTER AND DELAY COUNTER. The event and delay counters work together to form one of the sources for trace termination. The 16-bit event counter counts *Events* as specified by the event control logic and decrements on command. It can be written and read through the CRU. When the last *Event* arrives (i.e., the *Event* signal that decrements the counter to zero), the event counter enables the delay counter to start counting. If the event counter begins at zero, 65,536 *Events* will be counted before the delay counter starts.

The delay counter is an 8-bit programmable counter which counts from 0 to 255. When the event counter has decremented to zero, the delay counter decrements by one every time a trace sample clock occurs. When enabled by the event counter, the delay counter counts trace memory cycles to determine when an attempt should be made to stop tracing. An attempt is made to stop tracing immediately upon arrival of the last *Event*, if the delay counter is set to zero. In this manner, the delay counter allows the user to specify how much of the trace memory contains information before the last *Event* and how much contains information after the last *Event*.

Following is an example of how the event counter and the delay counter work together to stop tracing: when the event counter decrements to 0, the delay counter is enabled to start counting. Since the event counter counts events as defined by the comparison logic, it counts on a coarse scale, such as how many times a particular address was passed. The delay counter, when enabled, is decremented at each qualified clock period. After the event counter decrements to 0, each sample then decrements the delay counter. The event counter continues to count any *Events* that come in until tracing stops. When both the event counter and the delay counter decrements to 0, the *Event*,Delay status bit is set and the program can specify that the trace module stop at that point. This allows the user to position the last *Event* anywhere in the trace RAM. With the delay counter initialized at 0, as soon as the last event comes in, event/delay status is set and tracing halts. This puts the last *Event* as the last item recorded in the trace RAM. On the other hand, if 255 is entered in the delay counter, then the last *Event* could be positioned at the other extreme of memory, and all of the data in memory would be post-*Event* information.

1.3.11 TERMINATION MODES. The decision to terminate tracing can come from two sources:

- Trace Memory Full (TMF) signal from the address counter.
- *Event*,Delay status when both the event counter and the delay counter decrement to zero.

When the control block is set by the software to monitor TMF, the trace module will attempt to terminate when the trace RAM fills. In this case, termination can take place in two ways, both of which are program-selectable. One termination mode applies to cases where the emulator module is not affected. In this mode, the interrupt signal from the trace module to the host computer is not masked (interrupt can be generated), so the trace module can be programmed to interrupt the host and halt itself and all other trace modules in the system (if any) immediately. In the other termination mode, the trace module is programmed to stop the emulator. When the emulator sees HALT, it will proceed to stop at the end of the instruction in progress, so there is a potential for more samples before the trace module really stops. The trace module can also be programmed to stop immediately or to continue tracing until the emulator replies with a Hold signal, which means it has reached a termination place and has stopped.

1.3.12 DIAGNOSTIC MODE. In the diagnostic mode, all of the inputs to the trace module are driven either directly or indirectly by the CRU interface. If the data cable is not connected, the diagnostic mode can be entered by setting the diagnostic bit (bit 21) on the CRU output to logic one. Thereafter, the equivalent of one external clock is generated every time the CRU writes to the diagnostic bit. In the diagnostic mode the external *Event* is driven by the synchronized internal *Event*, which always begins each trace as a logic zero and changes states on sample clocks. The 20 data bits and the qualifier inputs are all driven by the diagnostic data register. Setting the diagnostic bit (bit 21) to zero or issuing an IORESET clears the diagnostic mode.

1.3.13 MULTIPLE TRACE MODULES. As shown in Section II, multiple trace modules can be used in tandem by connecting the control cables together. Simultaneous starting and stopping is made possible by the control signal, Hold. No trace module will begin tracing until Hold is released and all of them will stop when Hold is asserted. When the internal clock is used with multiple trace modules, the module whose internal clock is turned on first will drive the internal clock to all trace modules. This guarantees that all trace modules will sample simultaneously in the internal clock mode.

SECTION II

INSTALLATION

2.1 GENERAL

This section provides instructions for the installation and initial checkout of the trace module, as well as planning requirements and unpacking instructions.

2.2 UNPACKING AND INSPECTING

The trace module is wrapped in bubble-pack and packaged in a rigid cardboard box, except when installed in the 990 Computer. When installed in a computer, the unpacking instructions for the computer chassis assembly apply. Unpack and inspect the trace module as follows:

1. Before opening the cardboard shipping box, inspect it for evidence of damage: crumpled corners, tears, water stains, etc.

2. Open the cardboard box and remove the bubble-pack wrapped circuit boards from the box.

3. Carefully remove bubble-pack wrapping and verify that the part number on the board is correct for the desired assembly.

4. Inspect all printed circuit boards and cables, especially if the shipping containers indicate damage. Look for breaks or cracks in the board, loose or missing components or connectors, broken wires, corrosion, or foreign material lodged between packaged pins that might cause a short circuit.

5. Inventory the parts received to verify they coincide with the shipping list. The number and type of assemblies required are listed in table 2-1.

Table 2-1. Trace Module Assembly and Accessories

Trace Module Kit, Part Number 949947

949910	Trace Module
949935	Emulator/Trace Data Cable
949936	Emulator/Trace Control Cable

Trace Data Probe Assembly, Part Number 944915

2.3 INSTALLATION

The trace module circuit board requires a full-width chassis slot in the CRU section of a 990 Computer, or in a CRU expansion chassis, as described in the hardware manuals listed in the Preface. The selected slot is significant because the location determines the base CRU address as well as the priority interrupt level in the scheme of the computer system. Standard computer chassis configurations are shown in figure 2-1.

After the permanent chassis slot location has been determined, install the trace module in the following manner:

1. Ensure that chassis power is off.
2. Insert the circuit board, component side up, into the selected slot until the board slides into the card guides on either side of the slot.
3. Push the board straight in until the edge connector engages the chassis backpanel. Verify that the guide slots in the circuit board mate properly with the alignment comb in the chassis backpanel.
4. Lock the ejector tabs in place.

2.3.1 CABLE CONNECTIONS. When the trace module circuit board has been installed in its chassis slot, install the cables for the particular configuration in which the trace module is to be used. A standalone trace module cabling configuration is shown in figure 2-2; a trace module interconnected with an emulator module is shown in figure 2-3.

CAUTION

The emulator/trace control cable connector is a right-angle ribbon cable connector. Be sure the connector block arrows are aligned when inserting the plug.

The two cable installation configurations are:

1. Direct data tracing at the target system breadboard which uses the trace data probe.
2. Interface with the emulator module which requires the emulator/trace data cable and control cable to be interconnected between the two modules.

NOTE

As many as four trace modules can be used to increase the trace width. Multiple trace modules can be connected together by daisy-chaining the control cables and connecting the final cable in the series to the emulator module. See figure 2-4.

2.4 TRACE DATA PROBE

CAUTION

To prevent damaging receivers on the trace module, voltages applied to the trace data probe inputs should be limited to between +5.5V and -1.0V.

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	8K TILINE MEMORY EXPANSION	N/A	0280	8K TILINE MEMORY EXPANSION	N/A
4	0260	24K TILINE MEM-TORY EXPANSION	N/A	0240	24K TILINE MEM-ORY 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	4	0140	EMULATOR	4
9	0120	TRACE MODULE	4	0100	TRACE MODULE	4
10	00E0	913A VDT	3	00C0	913A VDT	3
11	00A0	FLOPPY DISC CONTROLLER	7	0080	FLOPPY DISC CONTROLLER	7
12	0060	LINE PRINTER (OPTIONAL)	4	0040	CARD READER (OPTIONAL)	4
13	0020	PROM PROGRAMMER (OPTIONAL)	N/A	0000	733 OR 743 ASR/KSR (OPTIONAL)	6

990 10 13-SLOT 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	990/4/AU	N/A	N/A	990/4/AU	N/A
2	02E0	990/4 MEMORY EXPANSION	N/A	02C0	990/4 MEMORY EXPANSION	N/A
3	02A0	990/4 MEMORY EXPANSION	N/A	0280	990/4 MEMORY EXPANSION	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	4	0140	EMULATOR	4
9	0120	TRACE MODULE	4	0100	TRACE MODULE	4
10	00E0	913A VDT	3	00C0	913A VDT	3
11	00A0	FLOPPY DISC CONTROLLER	7	0080	FLOPPY DISC CONTROLLER	7
12	0060	LINE PRINTER (OPTIONAL)	4	0040	CARD READER (OPTIONAL)	4
13	C020	PROM PROGRAMMER	N/A	0000	733 ASR/KSR (OPTIONAL)	6

(A)136001

990/4 13-SLOT CHASSIS

Figure 2-1. Standard Computer Chassis Configuration

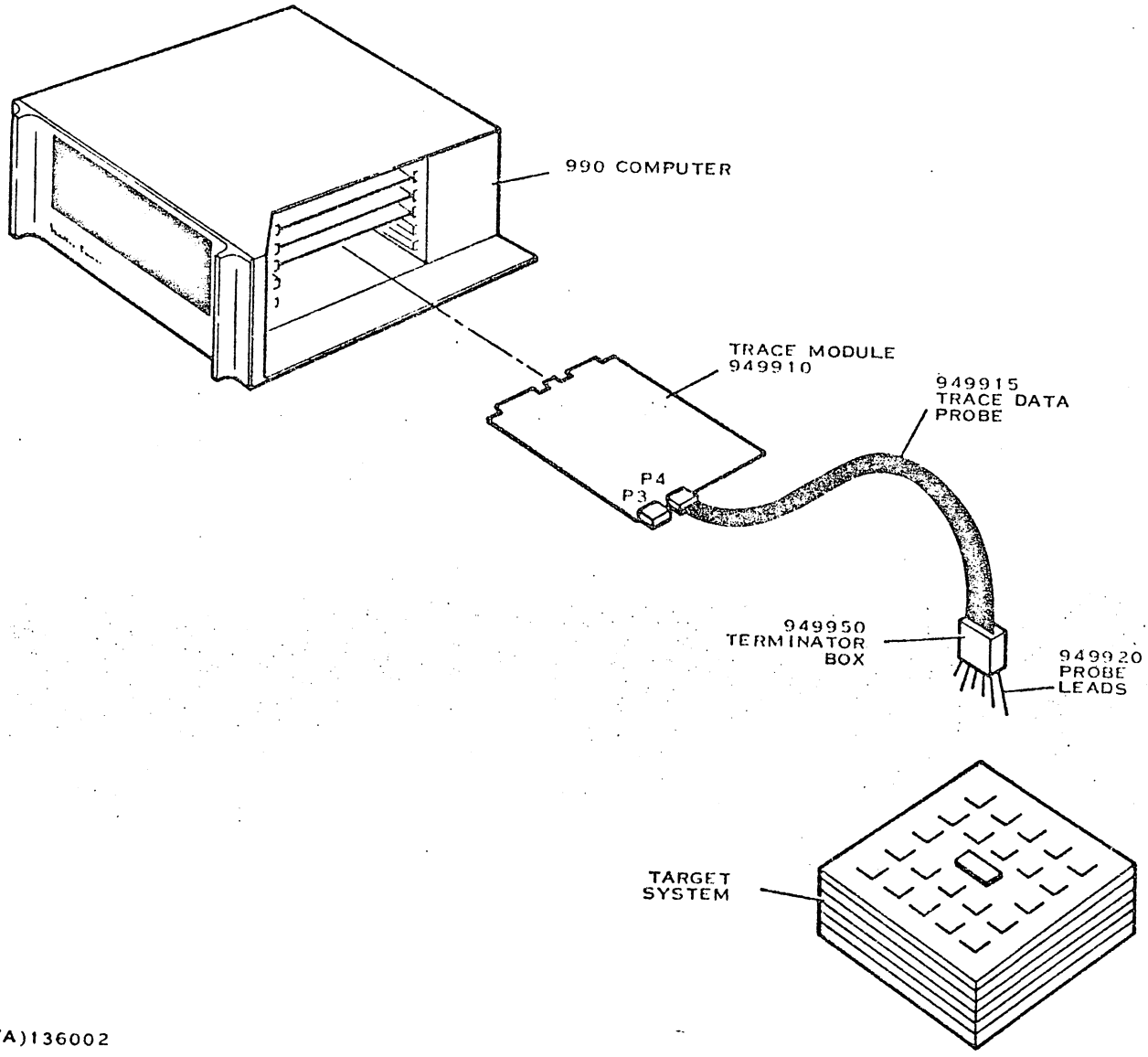


Figure 2-2. Standalone Trace Module Cabling

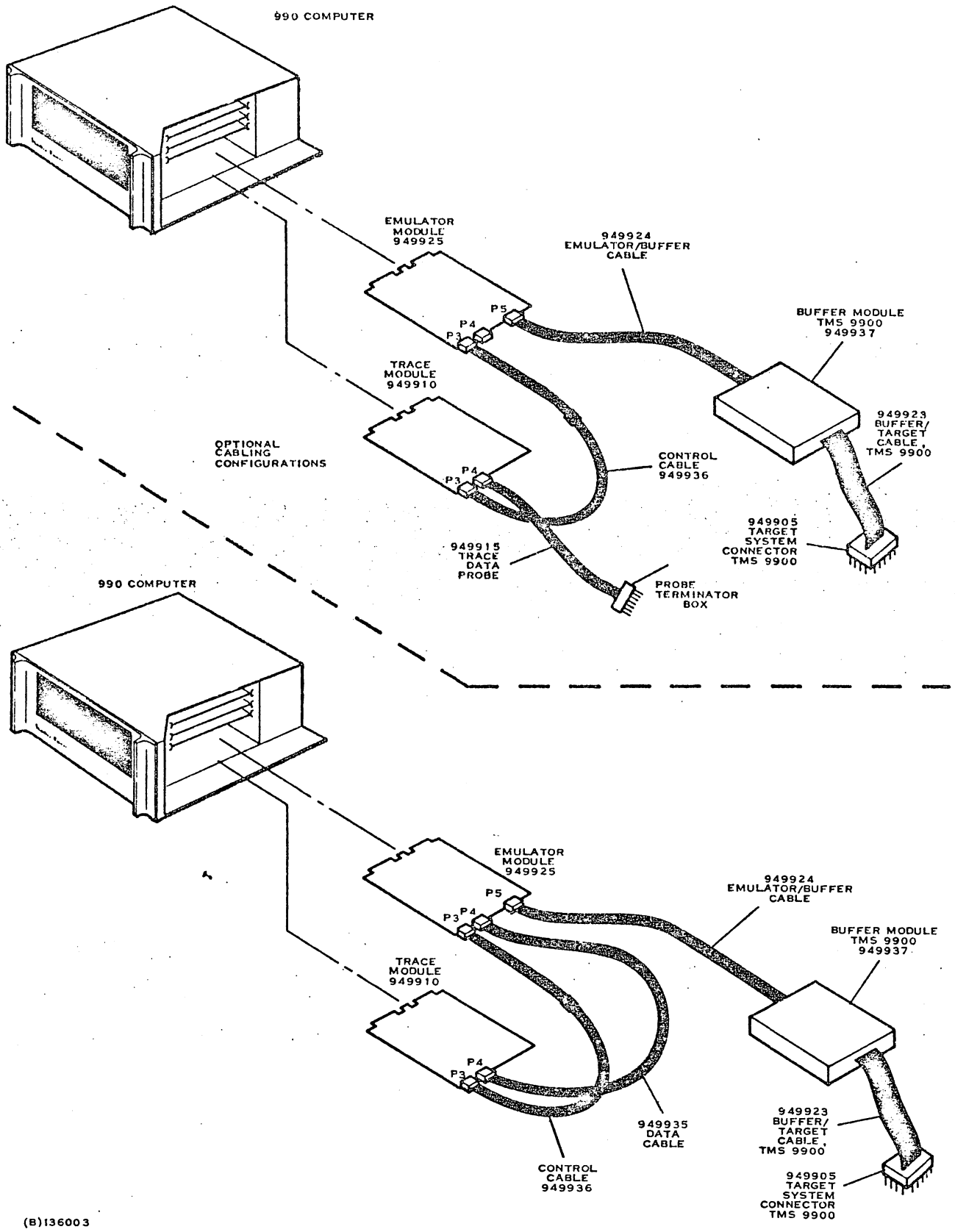


Figure 2-3. Trace Module and Emulator Module Interconnecting Cabling

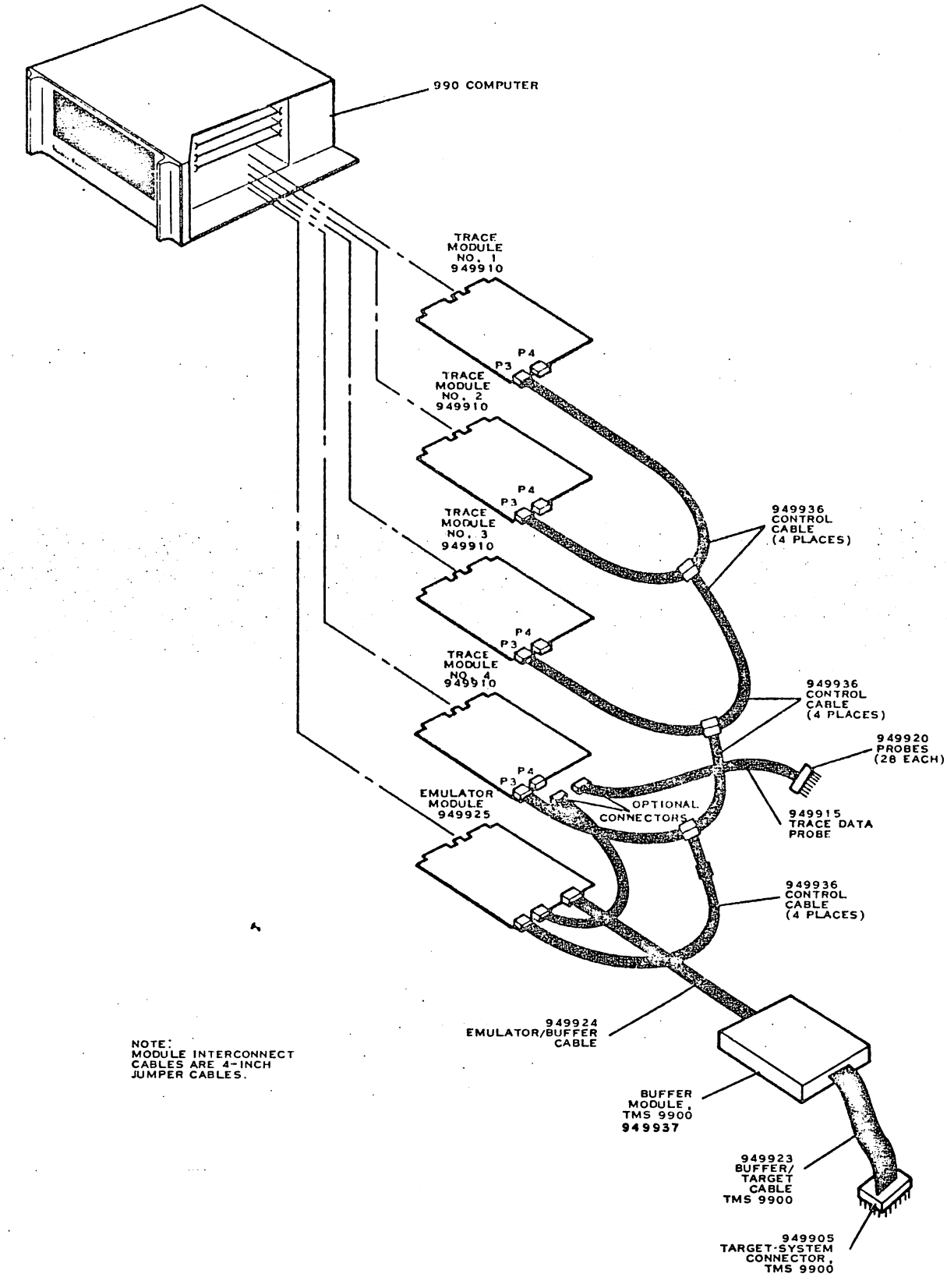


Figure 2-4. Cabling Configuration Using Multiple Trace Modules

The trace data probe consists of a 6-foot ribbon cable, a terminator box, and 28 replaceable leads installed in the terminator box. The leads are color-coded as shown in figure 2-5. In series with each lead is a 220-ohm resistor installed in the terminator box as shown in figure 2-6. The purpose of the resistor is to limit ac loading.

In a standard configuration, each data lead is equipped with female connectors for insertion on male wire-wrap pins at the target system breadboard. The 28 leads are:

20 data leads

4 qualifiers

2 grounds

1 clock

1 event

If a probe lead is damaged and needs replacing, spare leads are available in the Trace Probe Accessory Kit (949959-1). To replace a lead, refer to figure 2-7 and perform the following steps:

1. Loosen four screws in the terminator box and separate the cover halves.
2. Remove the damaged lead by sliding the female connector from its pin inside the terminator box.
3. Slide the new lead onto the pin and place the lead in the proper cutout in the box. Check the remaining leads to verify each is in its correct position.
4. Close the box and tighten the screws.

2.5 CHECKOUT

Power supplied to the trace module from the 990 Computer chassis is as follows:

Volts	Amperes	Tolerance
+5.0	5.0 (typical)	±3% regulation, 50 mV ripple

The host computer can employ a variety of peripheral devices and several combinations may be used for checkout and operation of the trace module. Installation, operation, and programming for the computer and the peripherals are contained in the respective installation and operation manuals listed in the Preface.

CAUTION

Computer power should always be turned off when installing the trace module in the system. The host system should be powered up before the target system is powered up.

A diagnostic program is used for the trace module checkout. Refer to the *Model 990 Computer Diagnostics Handbook* listed in the Preface to operate the diagnostic program.

General operation procedures are given in Section IV.

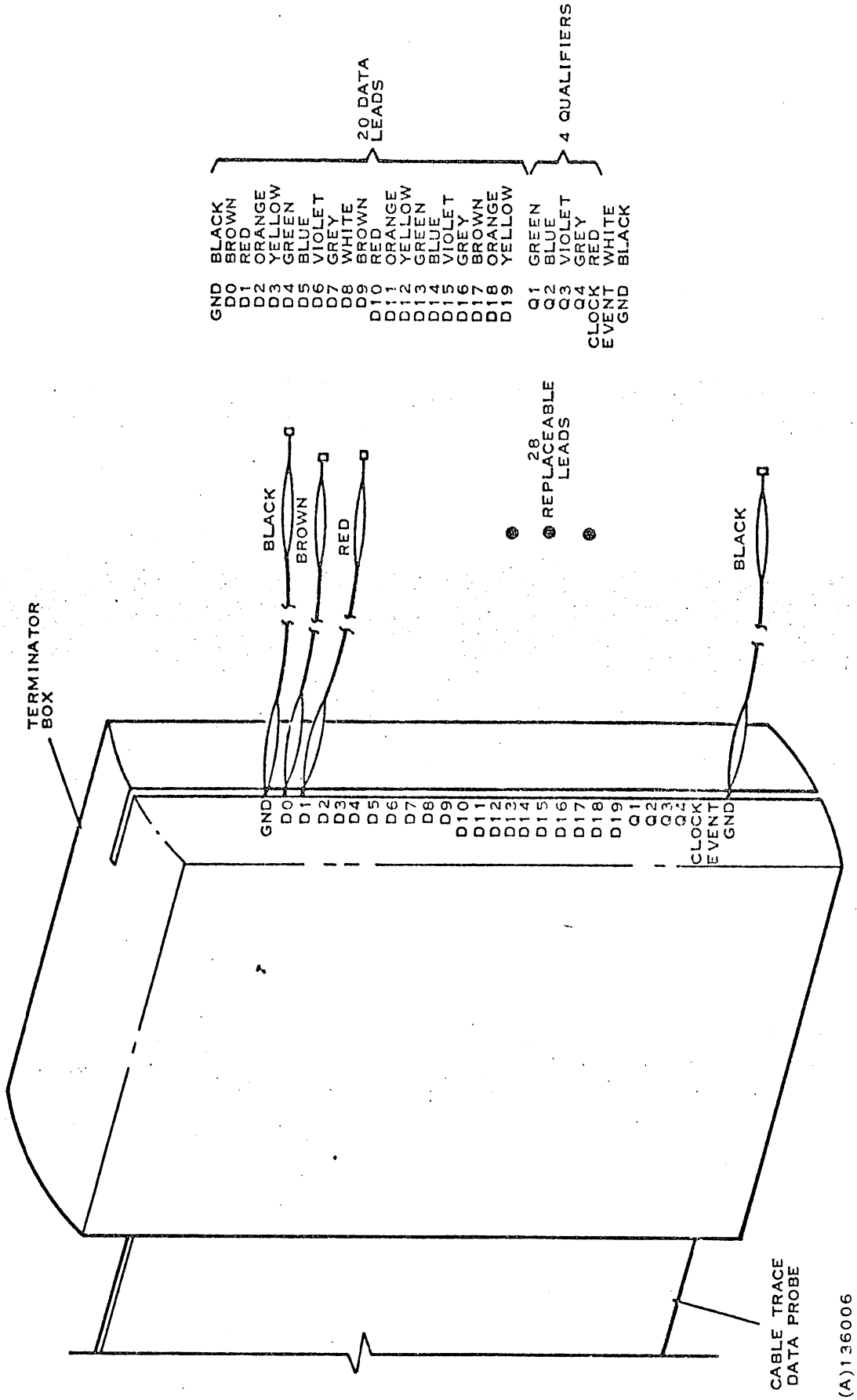


Figure 2-5. Trace Data Probe Terminator Box and Leads

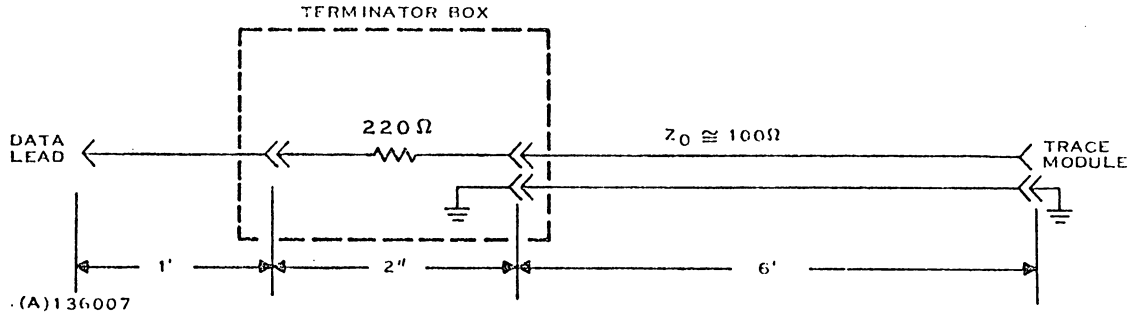


Figure 2-6. Data Probe Lead Electrical Equivalent

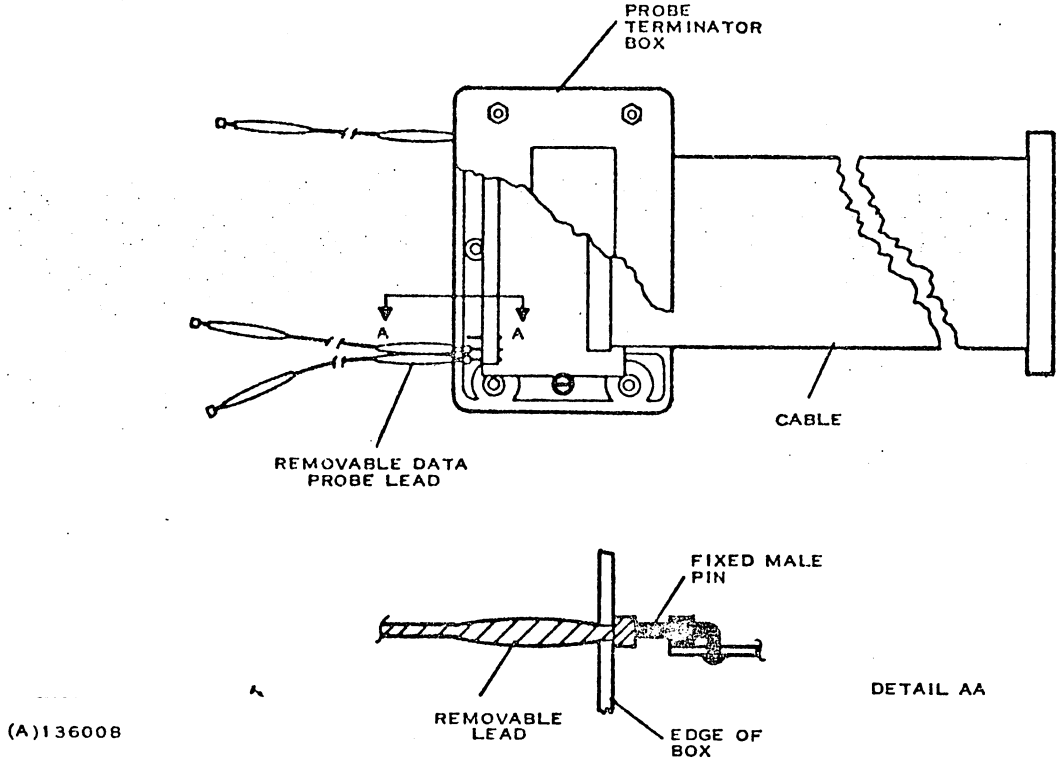


Figure 2-7. Replaceable Probe Leads

SECTION III

PROGRAMMING

3.1 GENERAL

This section describes the trace module interfaces and provides information for use in designing service routines which may be required by specific applications. The information is directed to programmers familiar with the 990 Computer, so only basic programming requirements are given. For additional software information, refer to the publications listed in the Preface.

3.2 TRACE MODULE TO CRU INTERFACE

The trace module interface to the computer CRU consists of 20 data bits and 12 control bits for a total of 32 bits. Each of the 12 control bits is assigned a unique bit address; the remaining 20 bits are used for data transfer. The base address of the trace module/CRU interface is determined by the physical slot location of the trace module in the computer chassis (see figure 2-1). All input and output bits are cleared to zero after a power-up or RSET (Reset) instruction.

3.2.1 CRU OUTPUT BIT ASSIGNMENTS. The addressable output from the CRU to the trace module consists of 20 data bits and 12 control bits, as shown in figure 3-1. These bit assignments are described in table 3-1.

3.2.2 CRU INPUT BIT ASSIGNMENTS. The CRU input from the trace module consists of 20 data bits and 12 status bits, as shown in figure 3-2 and described in table 3-2.

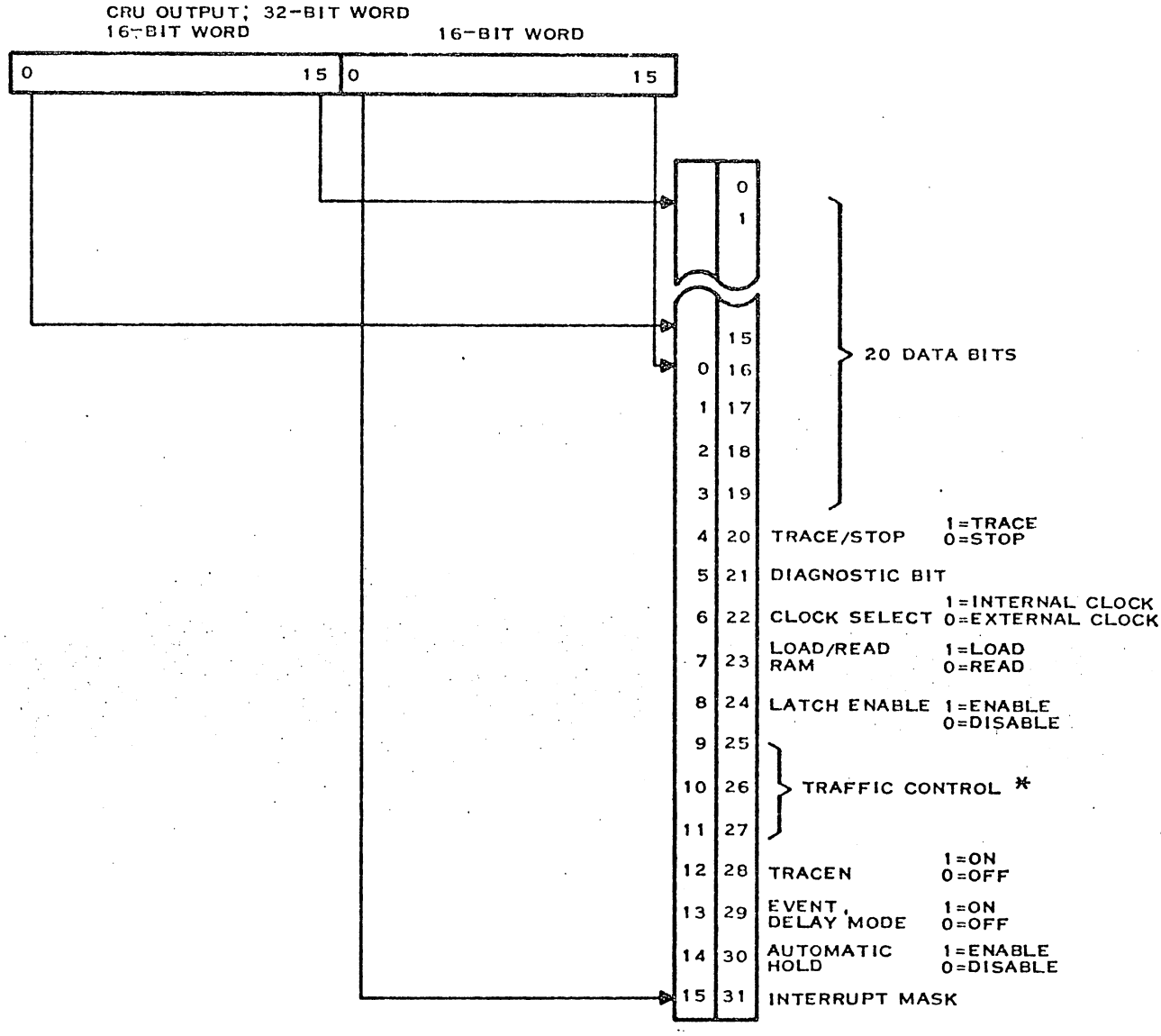
3.2.3 TRACE MODULE INTERRUPTS. The stimulus for a general interrupt is generated when CRU output bit 31 (Interrupt Mask) is logic zero and CRU input bit 31 (Halt/Interrupt Asserted/Pending) is logic one. As described for input and output bits 28 and 29, the two causes of an interrupt are Trace Memory Full (TMF) and Event,Delay. Since both of these signals are cleared by addressing output bit 20 (Trace/Stop), the interrupt can be cleared by setting bit 20 to either logic one or zero.

3.2.4 TRACE MODULE/EMULATOR MODULE INTERFACE. The control signal interchange between the trace module and emulator are listed in table 3-3.

3.3 EXAMPLES OF PROGRAM ROUTINES

The following paragraphs present some example standalone routines for the trace module. These routines involve eight programmable registers that are accessed through the computer CRU.

The computer CRU input to the trace module provides three bits (25, 26, 27) that are identified as Traffic Control. The logic state of these three bits steer the 20 output data bits of the CRU (bits 0-19) into one of eight programmable registers in the trace module. The various states of these three bits, and the data fields they load into the eight registers, are diagrammed in table 3-1.



TRAFFIC CONTROL *

27	26	25	
0	0	0	DATA INPUT REGISTER
0	0	1	QUALIFIER MASK AND REGISTER
0	1	0	COMPARE REGISTER
0	1	1	COMPARE MASK
1	0	0	EVENT CONTROL
1	0	1	DELAY COUNTER
1	1	0	ADDRESS POINTER
1	1	1	DIAGNOSTIC DATA REGISTER

(A)136009

Figure 3-1. CRU Output Bit Assignments

Table 3-1. CRU Output Bit Description

Bit	Name	Description
0-19	20 data bits	The 20 data output bits are used on the trace module as a shift register. All data outputs must be a total of 20 bits in length, with the LSB output first. Bits 25-27 determine into which register the data will actually be stored.
20	Trace/Stop	Setting this bit to logic one enables tracing. Tracing will not start until the emulator module releases HOLD–. Tracing stops when either the emulator or another trace module enters Hold. Input bits 28, 29 and 31 and the Interrupt/Hold signal are cleared by writing to bit 20.
21	Diagnostic Bit	The first time this bit is set to logic one with the data cable disconnected, the trace module enters the Diagnostic Mode. Thereafter, each time this bit is addressed, a diagnostic clock is generated. Setting this bit to zero causes the trace module to exit the diagnostic mode.
22	Clock Select	Setting this bit to logic one selects the internal clock; setting it to logic zero selects the external clock.
23	Load/Read RAM	This bit is used in conjunction with bits 0-19 and bits 25-27 for the computer to access the trace RAM. When a zero is written to this bit, the data in the trace RAM stored at the address currently indicated by the address counter is transferred to the CRU input multiplexer. When a one is written to bit 23, the data in the data input register is written into the trace RAM at the address indicated by the address counter. Both loading and reading the trace RAM decrement the address counter after the data transfer.
24	Latch Enable	This bit controls the special latch on data input bits D0-D3. When this bit is logic one, the latch is enabled. When zero, the latch is disabled.
25-27	Traffic Control	The state of these three bits steer the CRU data bits 0-19 into one of eight registers in the trace module. When these 20 data bits are written to a register, the previous contents of that register are moved to the CRU data input register. This implies that the contents of any register can be read (destructively) by writing 20 data bits to the register and then reading the CRU data input. The traffic control address of all the registers involved and the bit assignments of each register are shown on the following page:

Table 3-1. CRU Output Bit Description (Continued)

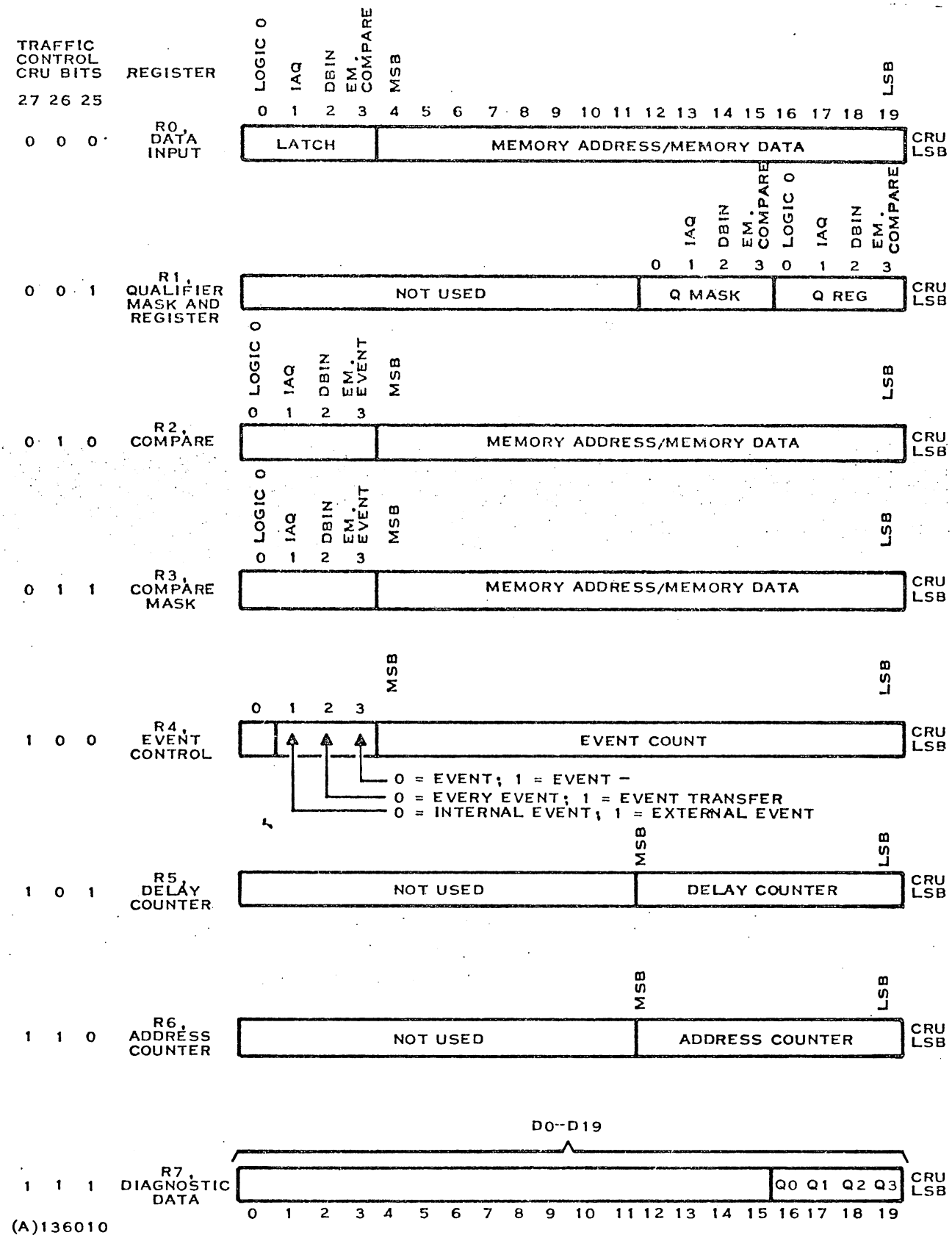


Table 3-1. CRU Output Bit Description (Continued)

Bit	Name	Description
28	Trace N	Logic one in this bit enables the Trace Memory Full (TMF) logic to attempt to terminate tracing, when data is traced into address 1 of the trace RAM. If tracing begins with N as the contents of the address pointer, then TMF appears after N trace samples. If output bit 28 is logic zero, TMF is reported at input bit 28, and no other action takes place.
29	Event, Delay Mode	If this bit is logic one, trace termination is attempted when the delay counter underflows. (The delay counter will not decrement until the event counter has also underflowed.) If this bit is logic zero, the only effect of the delay counter underflow is to set input bit 29, and no other action takes place.
30	Automatic Hold Enable	Logic one in this bit enables the trace module to automatically *hold* itself and all other trace modules if a termination condition is met (see output bits 28, 29). If this bit is logic zero, the trace module cannot stop itself; and it can be stopped only by the host, the emulator module, or another trace module entering HOLD.
31	Interrupt Mask	If this bit is logic one when a termination condition is reached (see bits 28, 29), no interrupt is generated but halt information is sent to the emulator. If this bit is logic zero, the stimulus for a general interrupt is generated when a termination condition is met.

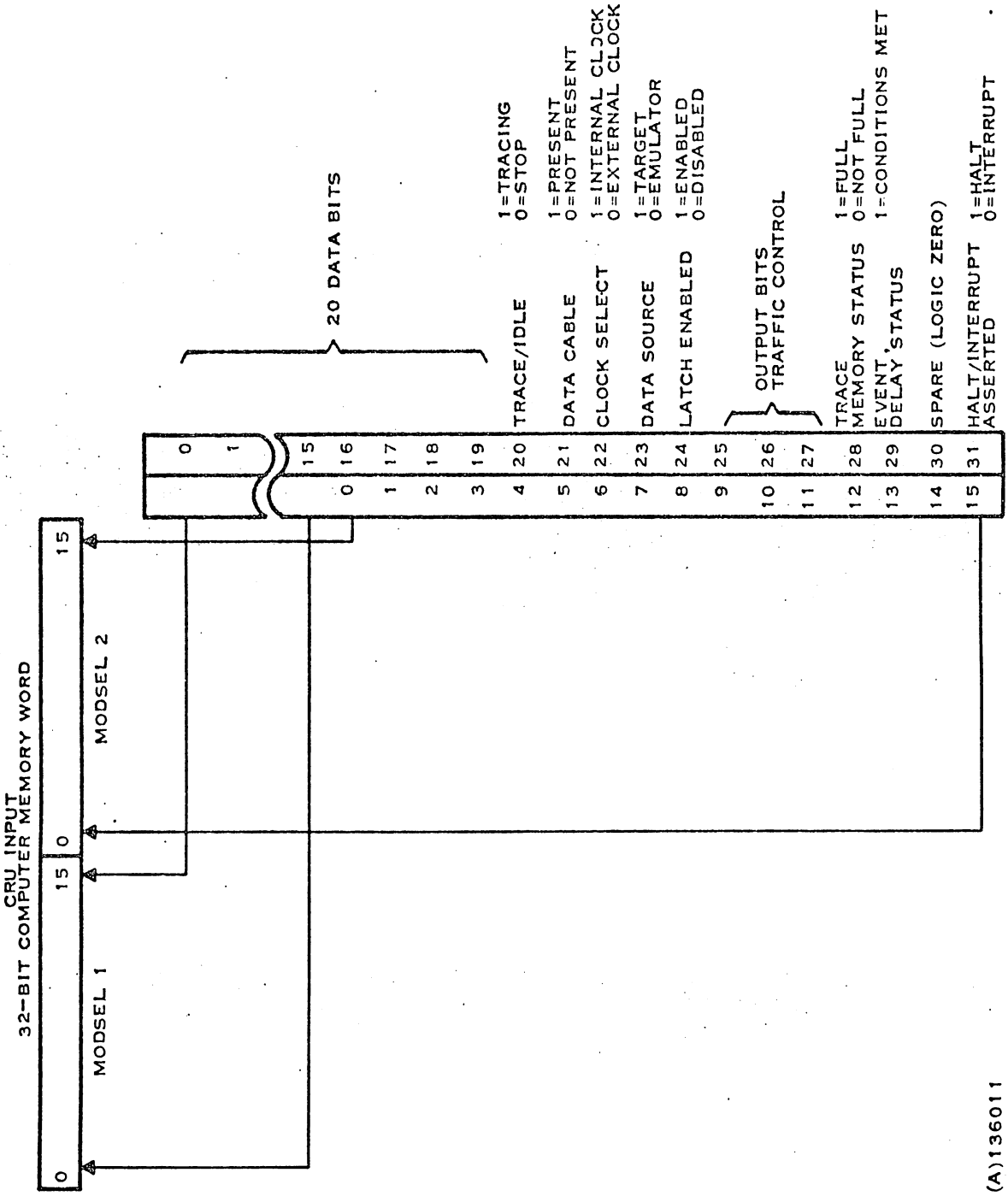


Figure 3-2. CRU Input Bit Assignments

Table 3-2. CRU Input Bit Description

Bits	Name	Description
0-19	20 Data Bits	CRU input data consisting of 20 bits. This data is determined by CRU output bits 23 and 25-27.
20	Trace/Idle	This bit is logic one as long as the trace module is tracing. When tracing terminates, the bit resets.
21	Data Cable	When the data cable is connected to the trace module this bit is logic one. A logic zero indicates that no cable is connected.
22	Clock Select	This bit is logic one if, and only if, the internal clock has been selected.
23	Data Source	This bit is logic one if the trace data probe cable is connected to the trace module. If the data cable is connected to the emulator module, or if no cable is present, this bit is logic zero.
24	Latch Enabled	This bit is logic one if the latch is enabled.
25-27	Traffic Control	These bits reflect the status of the corresponding CRU output bits 25-27.
28	Trace Memory Status	This bit is logic one when data has been traced into address 1 of the trace RAM. It is cleared by addressing CRU output bit 20.
29	Event, Delay Status	This bit is logic one when the Event, Delay condition is satisfied by the event and delay counters. It is cleared by addressing CRU output bit 20.
30	Spare	Logic zero.
31	Halt/Interrupt/ Asserted	This bit is logic one when the trace module has reached a terminating condition and has generated either an interrupt or a Halt. This bit is actually the combinational function of CRU output bits 28 and 29, and CRU inputs bits 28 and 29. IN31 = OUT29 • IN28 + OUT29 • IN29.

Table 3-3. Trace/Emulator Control Interface

Signal	Type Drive	Sender	Receiver	Connection	Description
HALT–	1	Trace	Emulator	P3-2	Command to emulator to halt emulation as soon as possible. The emulator cannot stop until the end of the current instruction.
HOLD–	2	Emulator Trace	Trace	P3-4	Tracing will not start until Hold is released and all trace modules will stop immediately when Hold is asserted.
TSE	1	Trace	Emulator	P3-6	Trace Sample Enable. Combinational result of the trace module Qualifier inputs and the Qualifier mask and compare registers. Can be used by emulator to qualify its trace.
ICLKSEL	3	Trace	Trace	P3-10	Control signal driven low by the first trace module to be placed in Internal Clock mode.
ICLKBUSS	4	Trace	Trace	P3-8	10 MHz clock driven by the trace module which controls ICLKSEL.

The eight programmable registers are:

R0 Data Input

R1 Qualifier and Mask

R2 Compare

R3 Compare Mask

R4 Event Count and Event Control

R5 Delay Counter

R6 Address Pointer

R7 Diagnostic Mode

Since these registers are loaded serially, the CRU output bits do not have unique addresses. Data integrity is maintained by following two rules when loading a serial register:

1. Output the least significant bit first.

2. Always transfer 20 data bits. For an example, refer to the WREG routine.

When a new, 20-bit data value is loaded into any one of the registers R1 through R7, the previous contents of that register are moved to the data input register (R0). The data field of the CRU input always addresses Register R0. R0 can be read at any time, without regard to bit order or transfer length. This means that the contents or status of any register may be read at any time by loading data into the register and then reading register R0. The procedure for loading and reading the serial registers is shown in figure 3-3.

3.3.1 CRU BIT DEFINITIONS. The names and definitions assigned to the CRU bits shown in table 3-4 make the sample routines in the following paragraphs easier to read and understand.

3.3.1.1 CRU Base Address. The following sample programs use a data word containing the CRU base address of the trace module. This base address is determined by the slot in which the trace module is connected. For example, when the trace module is installed in slot 6 of a 13-slot chassis:

```
CRUBAS   DATA >1C0
```

3.3.2 GSTAT ROUTINE. The simplest and most basic control routine merely reads the status of the trace module and returns the condition bits in register R0. GSTAT (Get Status) routine returns in register R0 the current status of the trace module in the form of bit flags directly corresponding to the readable bits in the trace module CRU interface. The value in R0 can be tested for the presence of each bit, as shown in figure 3-4.

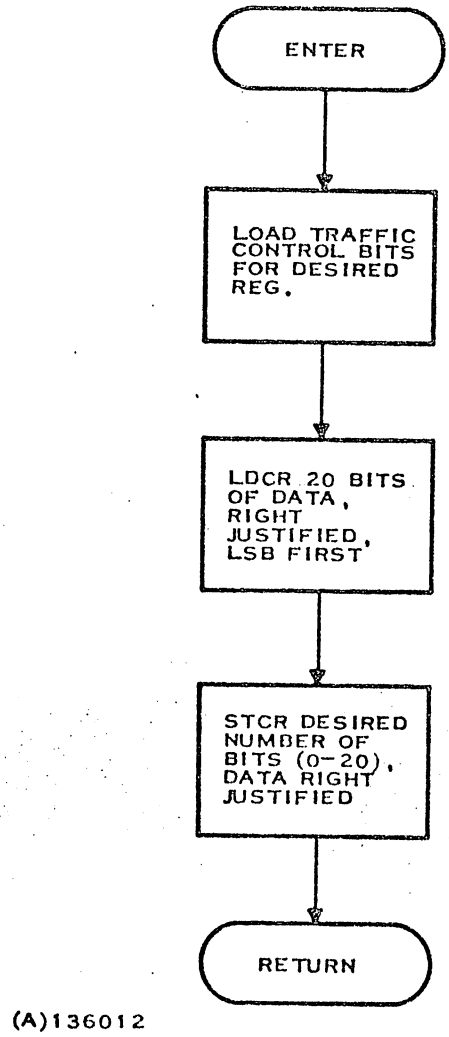


Figure 3-3. Loading and Reading the Serial Registers

Table 3-4. Trace Module CRU Bit Definitions

	* BIT *NAME		BIT VALUE	READ/ WRITE	USE
	*				
0001	T. RUN	EQU	1	R/W	SET/CLR TO START/STOP.
0002	T. CAB	EQU	2	R	SET IF DATA CABLE PRESENT.
0004	T. CLK	EQU	4	R/W	SET/CLR FOR INT/EXTERNAL CLOCK.
0008	T. DAT	EQU	8	R	SET/CLR IF PROBES/EMULATOR DATA.
0008	T. RWB	EQU	8	W	SET/CLR TO WRITE/READ BUFFER.
0010	T. LAT	EQU	16	R/W	SET/CLR TO EN/DISABLE LATCH.
00E0	T. IND	EQU	224	R/W	REGISTER INDEX.
0100	T. FUL	EQU	256	R/W	SET/CLR FOR BUFFER OVERFLOW.
0200	T. EVT	EQU	512	R/W	SET/CLR FOR EVENT SATISFACTION.
0400	T. HLT	EQU	1024	W	SET/CLR TO HALT ON BREAK COND.
0800	T. INT	EQU	2048	R/W	SET/CLR FOR HALT EMU/INTR. HOST.

```
        * INTERFACE:
        *       BL    @GSTAT
        *             ON ENTRY...
        *                   NOTHING IS ASSUMED.
        *             ON EXIT...
        *                   R0...CONTAINS CURRENT STATUS.
        *                   R12..POINTS TO THE FIRST STATUS BIT OF
        *                         THE TRACE MODULE CRU INTERFACE.
        *
        GSTAT
        *
        * SET R12 TO POINT TO THE FIRST STATUS BIT IN THE TRACE
        * MODULE CRU INTERFACE, I.E.  20 BITS ABOVE THE BASE CRU
        * ADDRESS.  NOTE THAT THE OFFSET 20 MUST BE MULTIPLIED BY
        * TWO FOR THE 'STCR' INSTRUCTION TO WORK PROPERLY.
        *
C320            MOV   @CRUBAS,R12
0000'
022C            AI    R12,20*2
0028
        *
        * READ THE TWELVE STATUS BITS INTO R0, RIGHT-JUSTIFIED
        * IN THE 16-BIT WORD.  CLEAR UPPER FOUR BITS TO RETURN
        * ONLY INTERESTING BITS IN THE LOWER TWELVE.
        *
04C0            CLR   R0
3700            STCR  R0,12
045B            RT
```

Figure 3-4. GSTAT Sample Routine

3.3.3 SSTAT ROUTINE. The SSTAT (Set Status) routine corresponds closely to the GSTAT routine. This routine partially sets the status of the trace module with six of the twelve status bits. These bits are concerned primarily with the breakpoint conditions which the trace module will recognize and the actions the trace module takes when a breakpoint condition occurs. Also, two of the bits select the sampling clock and the data latch. The SSTAT routine example is shown in figure 3-5. The SSTAT routine sets the clock select bit, the latch enable bit, and the break condition and break action bits of the trace module. These constitute a partial state of the trace module. This routine does not start or stop the trace module.

3.3.4 WREG ROUTINE. The remainder of the trace module status is set with the registers stored in the trace module. These registers are accessed via the index field of the CRU interface and the 20 data bits in the CRU interface. Each register is 20 bits wide, although not every bit of each register is significant to the trace module. The WREG routine writes a 20-bit value into a specified register in the trace module. There are six normally used trace module registers, indexed one through six. The WREG routine example is shown in figure 3-6. WREG takes a 20-bit value from R0 and R1 and writes this value into the register specified by the value in R2.

Also, the diagnostic register, index 7, and the data input register, index 0, may be written into.

```
                * INTERFACE:
                *       BL    @SSTAT
                *             ON ENTRY...
                *                   R0...CONTAINS THE VALUES OF THE BITS TO
                *                       BE SET/CLEARED.
                *             ON EXIT...
                *                   R0...DESTROYED.
                *                   R1...DESTROYED.
                *                   R3...DESTROYED.
                *                   R12..POINTS TO THE FIRST STATUS BIT OF
                *                       THE TRACE MODULE CRU INTERFACE.
                *             ROUTINES CALLED...
                *                   GSTAT
                *
0010            SSTAT
                *
                * SAVE THE LINK REGISTER IN R3, BECAUSE THIS ROUTINE WILL
                * CALL GSTAT, AND THE LINK REGISTER WILL BE DESTROYED BY
                * THE CALL.  SAVE R0 IN R1 BECAUSE GSTAT WILL RETURN THE
                * TRACE MODULE STATUS IN R0, DESTROYING THE VALUE PASSED TO
                * THIS ROUTINE.
                *
0010 C0CB               MOV   R11,R3
0012 C040               MOV   R0,R1
                *
                * CLEAR THE BITS IN THE CURRENT STATUS WHICH CAN BE SET OR
                * CLEARED BY SSTAT.  CLEAR THE BITS IN THE VALUE PASSED
                * WHICH CANNOT BE USED TO SET STATUS BITS IN SSTAT.  THEN
                * 'OR' THE TWO VALUES TOGETHER TO CREATE THE NEW STATUS.
                *
0014 0240               ANDI  R0,>00EB
0016 00EB
0018 0241               ANDI  R1,>0F14
001A 0F14
001C E001               SOC   R1,R0
                *
                * WRITE THE NEW STATUS ONTO THE CRU INTERFACE.  R12 POINTS
                * TO THE FIRST STATUS BIT BECAUSE GSTAT LEFT IT SO.
                *
001E 3300               LDCR  R0,12
                *
                * RETURN TO THE CALLER, USING RETURN ADDRESS SAVED IN R3
                * ABOVE.
                *
0020 0453               B     *R3
```

Figure 3-5. SSTAT Sample Routine

```
*  INTERFACE:
*        BL    @WREG
*              ON ENTRY...
*                    R0...CONTAINS THE LOW-ORDER SIXTEEN BITS
*                         OF THE VALUE TO BE WRITTEN INTO THE
*                         REGISTER.
*                    R1...CONTAINS, RIGHT-JUSTIFIED IN THE
*                         HIGH-ORDER BYTE, THE HIGH-ORDER FOUR
*                         BITS OF THE VALUE TO BE WRITTEN INTO
*                         THE REGISTER.
*                    R2...CONTAINS, RIGHT-JUSTIFIED IN THE
*                         HIGH-ORDER BYTE, THE INDEX OF THE
*                         REGISTER INTO WHICH THE VALUE IS TO
*                         BE WRITTEN.
*              ON EXIT...
*                    R0...AS ON ENTRY.
*                    R1...AS ON ENTRY.
*                    R2...AS ON ENTRY.
*                    R12..POINTS TO THE BASE ADDRESS OF THE
*                         TRACE MODULE CRU INTERFACE.
*
WREG
*
* SET R12 TO POINT TO THE INDEX FIELD OF THE TRACE MODULE
* CRU INTERFACE.  NOTE THAT THE BIT DISPLACEMENT, 25, MUST
* BE MULTIPLIED BY TWO FOR THE 'LDCR' INSTRUCTION TO WORK
* PROPERLY.
*
C320          MOV   @CRUBAS,R12
0000'
022C          AI    R12,25*2
0032
*
* SET THE INDEX FIELD TO THE REGISTER INDEX PASSED IN THE
* HIGH-BYTE OF R2.  THEN THAT REGISTER WILL BE ACCESSIBLE
* THROUGH THE DATA FIELD OF THE CRU INTERFACE.
*
30C2          LDCR  R2,3
*
* RESET R12 TO POINT TO THE BEGINNING OF THE TRACE MODULE
* CRU INTERFACE, WHERE THE 20-BIT DATA FIELD BEGINS.  THEN
* WRITE THE 20-BIT VALUE PASSED IN R0 AND R1 INTO THE DATA
* FIELD, WHENCE IT WILL BE TRANSFERRED INTO THE REGISTER
* INDEXED BY R2.  THE ORDER OF THE 'LDCR' INSTRUCTIONS MUST
* BE AS GIVEN.
*
022C          AI    R12,-25*2
FFCE
3000          LDCR  R0,0
3101          LDCR  R1,4
045B          RT
```

Figure 3-6. WREG Sample Routine

3.3.5 RREG ROUTINE. RREG is a routine to read the contents of a trace module register. The 20-bit value read from the register is returned to the caller, although not all 20 bits may be meaningful. The RREG routine uses the WREG routine, because a register must have a value written into it before the original contents can be read. After a write into a register, the previous value of that register can be read from the data field of the CRU interface. After the value has been read, the original value must be restored to the register because the write operation destroyed the original value. As shown in figure 3-7, RREG reads the 20-bit value of the trace module register that is indexed by R2 and places the value into R0 and R1. Since the reading process is destructive, RREG writes the value back into the register to maintain the original content.

3.3.6 STRT ROUTINE. When the trace module status has been set, the trace module may be started, and this is accomplished by the STRT routine. When the trace module is started, it will break when the proper break condition occurs. The status bits set by SSTAT will determine whether the trace module:

- Halts on a break or continues.
- Halts the emulator or interrupts the host computer.

The procedure given in figure 3-8 enables the trace module to begin tracing as soon as the clock signals and qualifier signals attain the required values for a sample to be taken.

3.3.7 STOP ROUTINE. If the trace module does not stop itself, the program must be able to do so. The routine shown in figure 3-9 disables the trace module from tracing.

3.3.8 RBUF ROUTINE.

NOTE

See paragraph 3.4 for a description of loading and reading the trace RAM.

After the trace module has run and has stopped, the traced data may be read from the trace RAM buffer. The RBUF routine, shown in figure 3-10, reads a specified 20-bit word of the trace buffer and returns the value in R0 and R1. The index is specified in R2 and must lie between 0 and 255.

3.4 LOADING/READING THE TRACE RAM

Figures 3-11 and 3-12 illustrate reading and loading the trace RAM. The trace RAM deviates from the normal load/read procedures. To load the trace RAM, the Traffic Control bits (CRU bits 25-27) should be set so that data can be placed directly into the CRU data input register. When the data is ready, writing a logic one to bit 23 writes the data into the RAM at the current address and increments the address.

To read the trace RAM, once the address is correct, bit 23 should be set to 0. Each time a 0 is written to bit 23, 20 data bits are transferred to the CRU data input register and the trace RAM address is decremented.

```
*  INTERFACE:
*       BL    @RREG
*             ON ENTRY...
*                  R2...CONTAINS, RIGHT-JUSTIFIED IN THE
*                       HIGH-ORDER BYTE, THE INDEX OF THE
*                       REGISTER TO BE READ.  THERE ARE
*                       SIX REGISTERS, NUMBERED 1...6.
*             ON EXIT...
*                  R0...CONTAINS THE LOW-ORDER SIXTEEN BITS
*                       OF THE 20-BIT VALUE READ FROM THE
*                       INDEXED REGISTER.
*                  R1...CONTAINS, RIGHT-JUSTIFIED IN THE
*                       HIGH-ORDER BYTE, THE HIGH-ORDER FOUR
*                       BITS OF THE 20-BIT VALUE READ FROM
*                       THE INDEXED REGISTER.
*                  R2...AS ON ENTRY.
*                  R3...DESTROYED.
*                  R12..POINTS TO THE 17'TH LEAST SIGNIFICANT
*                       BIT OF THE DATA FIELD OF THE TRACE
*                       MODULE CRU INTERFACE.
*             ROUTINES CALLED...
*                  WREG
*
RREG
*
* SAVE THE LINK REGISTER IN R3 BECAUSE RREG CALLS WREG, AND
* THE CALL WILL DESTROY THE LINK REGISTER.  THEN CALL WREG
* TO WRITE SOMETHING (THE CURRENT CONTENTS OF R0 AND R1)
* INTO THE INDEXED REGISTER, FORCING THE PREVIOUS VALUE OF
* THE REGISTER INTO THE DATA FIELD OF THE CRU INTERFACE.
*
CBCB          MOV   R11,R3
06A0          BL    @WREG
0022'
*
* NOW READ THE PREVIOUS VALUE OF THE INDEXED REGISTER FROM
* THE DATA FIELD, INTO R0 AND R1 (HIGH BYTE).  R12 IS
* POINTING TO THE DATA FIELD OF THE CRU INTERFACE BECAUSE
* WREG LEFT IT SO.
*
3400          STCR  R0,0
022C          AI    R12,16*2
0020
3501          STCR  R1,4
*
* NOW WRITE THE VALUE BACK INTO THE INDEXED REGISTER WITH
* WREG.  R2 STILL CONTAINS THE INDEX BECAUSE WREG DOES NOT
* DESTROY IT.
06A0          BL    @WREG
0022'
*
* RETURN THE VALUE IN R0 AND R1 TO THE CALLER.  THE VALUE
* IS STILL IN R0 AND R1 BECAUSE WREG DOES NOT DESTROY IT.
* THE RETURN ADDRESS IS IN R3, WHERE IT WAS SAVED ABOVE.
*
0453          B     *R3
```

Figure 3-7. RREG Sample Routine

```
        * ROUTINE:  STRT

        * ABSTRACT: STRT ENABLES THE TRACE MODULE TO BEGIN TRACING
        *       AS SOON AS CLOCK SIGNALS AND QUALIFIER SIGNALS
        *       ATTAIN REQUIRED VALUES FOR A SAMPLE TO BE TAKEN.
        * INTERFACE:
        *       BL    @STRT
        *             ON ENTRY...
        *                   NOTHING IS ASSUMED.
        *             ON EXIT...
        *                   R12..POINTS TO THE BASE ADDRESS OF THE
        *                        TRACE MODULE CRU INTERFACE.
        *
        STRT
        *
        * SET R12 TO POINT TO THE BASE ADDRESS OF THE TRACE MODULE
        * CRU INTERFACE.  THEN THE 'SB' INSTRUCTION CAN BE USED TO
        * SET THE RUN BIT.
        *
C320            MOV   @CRUBAS,R12
0000'
1D01            SBO   T.RUN
045B            RT
```

Figure 3-8. STRT Routine

```
        * ROUTINE:  STOP
        * ABSTRACT: STOP DISABLES THE TRACE MODULE FROM TRACING.
        * INTERFACE:
        *       BL    @STOP
        *             ON ENTRY...
        *                   NOTHING IS ASSUMED.
        *             ON EXIT...
        *                   R12..POINTS TO THE BASE ADDRESS OF THE
        *                        TRACE MODULE CRU INTERFACE.
        STOP
        *
        * SET R12 TO POINT TO THE BASE ADDRESS OF THE TRACE MODULE
        * CRU INTERFACE.  THEN THE 'SBZ' INSTRUCTION CAN BE USED
        * TO CLEAR THE RUN BIT.
        *
C320            MOV   @CRUBAS,R12
0000'
1E01            SBZ   T.RUN
045B            RT
```

Figure 3-9. STOP Routine

```
*  INTERFACE.
*       BL    @RBUF
*             ON ENTRY...
*                  R2...CONTAINS, RIGHT-JUSTIFIED IN THE
*                       HIGH-ORDER BYTE, THE INDEX OF THE
*                       WORD OF TRACE BUFFER TO BE READ.
*             ON EXIT...
*                  R0...CONTAINS THE LOW-ORDER SIXTEEN BITS
*                       OF THE INDEXED WORD OF TRACE BUFFER.
*                  R1...CONTAINS, RIGHT-JUSTIFIED IN THE
*                       HIGH-ORDER BYTE, THE HIGH-ORDER FOUR
*                       BITS OF THE INDEXED WORD OF TRACE
*                       BUFFER.
*                  R2...DESTROYED
*                  R3...DESTROYED.
*                  R12..POINTS TO THE 17'TH LEAST SIGNIFICANT
*                       BIT OF THE DATA FIELD OF THE TRACE
*                       MODULE CRU INTERFACE.
*             ROUTINES CALLED...
*                  WREG
*
RBUF
*
* SAVE THE LINK REGISTER IN R3 BECAUSE THIS ROUTINE CALLS
* WREG AND THE LINK REGISTER WILL BE DESTROYED BY THE CALL.
*
C0CB          MOV   R11,R3
*
* WRITE THE INDEX SPECIFIED IN R2 INTO THE TRACE MODULE
* REGISTER NUMBER SIX, THE BUFFER POINTER, USING WREG.
* PUT THE INDEX INTO R0, AND PUT 6 INTO THE HIGH BYTE
* OF R2 FOR WREG.  THE HIGH FOUR BITS DON'T MATTER, SO
* LEAVE R1 AS IT IS FOR WREG.
*
C002          MOV   R2,R0
0202          LI    R2,>0600
0600
06A0          BL    @WREG
0022'
*
* NOW THAT THE BUFFER POINTER IS SET TO THE INDEXED WORD
* OF TRACE BUFFER, PERFORM THE READ FROM THE BUFFER INTO
* THE DATA FIELD OF THE CRU INTERFACE BY CLEARING THE WRITE
* BIT.  R12 IS POINTING TO THE BASE ADDRESS OF THE CRU
* INTERFACE BECAUSE WREG LEFT IT SO.
*
1E08          SBZ   T.RWB
*
* READ THE 20-BIT VALUE FROM THE DATA FIELD INTO R0 AND
* R1 (HIGH BYTE).
*
3400          STCR  R0,0
022C          AI    R12,16*2
0020
3501          STCR  R1,4
*
* RETURN TO THE CALLER USING THE RETURN ADDRESS SAVED IN
* R3 ABOVE.
*
0453          B     *R3
```

Figure 3-10. RBUF Sample Routine

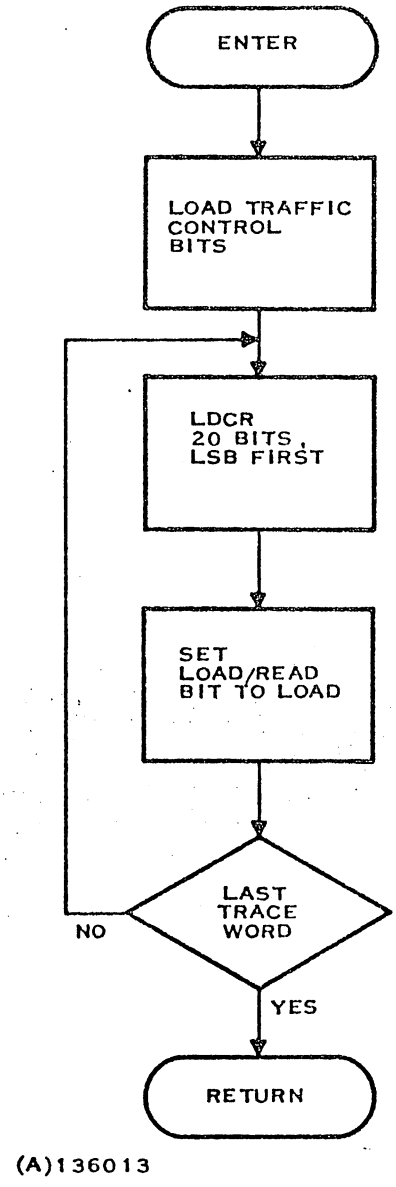


Figure 3-11. Loading the Trace RAM

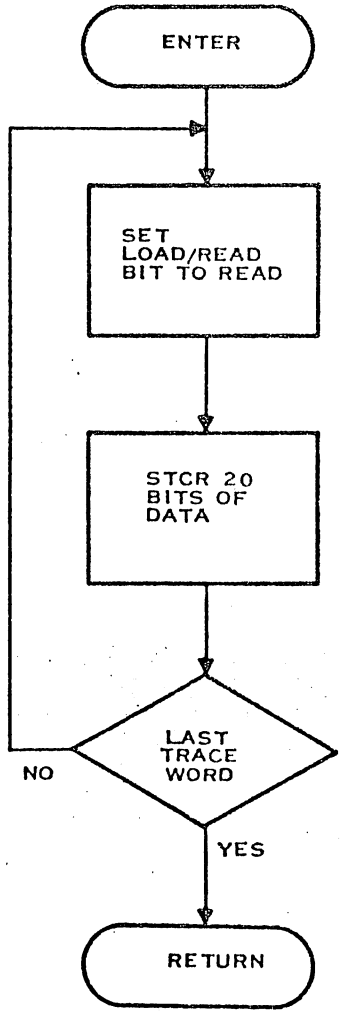


Figure 3-12. Reading the Trace RAM

SECTION IV

OPERATION

4.1 GENERAL

The operating procedure for the trace module is integral with the basic prototyping system. This means that the prototyping system is connected to the target. The program is loaded via the 733 Data Terminal from cassette or floppy disc storage and then executed by means of the 990 Computer control. When tracing is turned on by the computer and the emulator module leaves the Hold mode, the trace module begins tracing.

4.2 OPERATING PRECAUTIONS

The following precautions should be observed when the trace module is operating in a prototyping system:

1. Be sure computer power is "off' before inserting or removing the trace module from the computer chassis.

2. Host computer power must be "on" before the target system power is turned "on".

4.3 CONTROLS AND INDICATORS

The trace module has four, edge-mounted LED indicators, as shown in figure 4-1, to continuously indicate operating status. They are as follows:

EVENT,DELAY – The LED nearest the right edge of the board indicates the status of the event and delay counters. When both the event counter and the delay counter have decremented to zero, satisfying the event/delay conditions, CRU input bit 29 becomes a logic one and the EVENT,DELAY indicator lights. Bit 29 is cleared to logic zero and the indicator extinguished when Trace/Stop bit (CRU output bit 20) is addressed.

TMF – The TMF (Trace Memory Full) LED lights when tracing is active and data has been traced into address 1 of the trace RAM. At the same time the TMF indicator lights, the trace memory status bit (CRU input bit 28) becomes logic one. Bit 28 is cleared to logic zero and the TMF indicator extinguished when the Trace/Stop bit (CRU output bit 20) addressed.

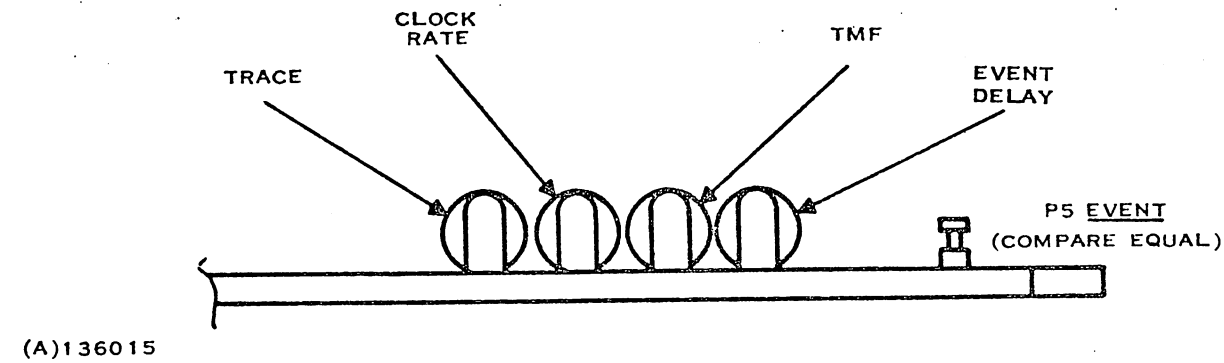


Figure 4-1. Trace Module LEDs

CLK RATE – The clock rate (or sample clock rate) LED indicates by its intensity the frequency of the clock when tracing is taking place. In other words, the faster the sampling frequency, the brighter the light.

TRACE – The LED nearest the center of the board is the trace indicator that lights when the trace module is recording or looking for sample clocks. The TRACE LED works in conjunction with the Trace/Idle CRU input bit 20; i.e., when bit 20 is logic one and the trace module is tracing, the TRACE LED is lit. When the trace module is in the idle mode and bit 20 is logic zero, the LED is extinguished.

4.3.1 *EVENT* INDICATION. At the top right-hand edge (as viewed in figure 4-1) of the trace module circuit board is located a test point, labeled P5, for the purpose of monitoring the output of the data word recognition logic. A scope test probe may be connected to P5 to determine the results of the *Event*, or the output of the trace module word recognizer logic. When the comparison between the data input word and the programmed word is true, the *Event* Indication at P5 is low. When the comparison is false (the data input word and the programmed word do not compare), P5 is high. This output is a TTL level.

4.4 PROGRAM LOADING

The basic prototyping system can use a 733 ASR/KSR Data Terminal, with cassette storage and loading, or it can use some other program I/O device such as an FD800 Dual Floppy Disc for loading and executing programs, as dictated by the peripherals employed in the user's system. For information on loading and executing software programs, refer to the applicable manuals listed in the Preface.

ALPHABETICAL INDEX

Address, CRU 2.3, F2-1

Base Address, CRU 2.3, 3.3.1.1, F2-1
Bit Definitions, CRU 3.3.1, T3-4
Block Diagram, System F1-3

Cable Connections 2.3.1, F2-2, F2-3
Checkout, Initial 2.4
CLK RATE LED 4.3, F4-1
Clock 1.2, 1.3.5
Configuration, System F1-2
Connections, Cable 2.3.1, F2-2, F2-3
Controls 4.3
Counter:
  Delay 1.3.10
  Event 1.3.9
CRU:
  Base Address 2.3, 3.3.1.1, F2-1
  Bit Definitions 3.3.1, T3-4
  Interface 1.3.2.1
    Input Bits 3.2.2, F3-2, T3-2
  Output Bits 3.2.1, F3-1, T3-1

Data:
  Cable Interface 1.3.2.2, F2-5
  Latch 1.3.4
  Probe 1.2
Delay Counter 1.3.10
Diagnostic Mode 1.3.12

Emulator Module 1.2
Electrical Probe Lead 2.4, F2-6
Event:
  Counter 1.3.9
  Test Point 4.3.1
EVENT,DELAY LED 4.3, F4-1

Functional Description 1.3.1, F1-3

GSTAT Routine 3.3.2, F3-4

Indicators 4.3
Initial Checkout 2.4
Input Bits, CRU 3.2.2, F3-2, T3-2
Inspection 2.2
Installation 2.3, F2-1
Interface:
  CRU 1.3.2.1
  Data Cable 1.3.2.2
  Signals 3.2.3, T3-3
  Trace Module/CRU 3.2
Interrupt, Trace Module 3.2.4

Latch, Data 1.3.4
Leads, Probe F2-5
LEDs, Trace Module 4.3, F4-1
Loading, Program 4.4
Loading/Reading:
  Serial Registers F3-3
  Trace RAM 3.4, F3-11, F3-12
Logic State Recorder 1.2

Mode:
  Diagnostic 1.3.12
  Termination 1.3.11
Module, Emulator 1.2
Multiple Trace Modules 1.3.13, F2-4

Operating Procedure 4.1
Output Bits, CRU 3.2.1, F3-1, T3-1

Part Numbers T2-1
Probe, Data 1.2
Probe Leads F2-5
  Electrical 2.4, F2-6
  Replacement 2.4, F2-7
Procedure, Operating 4.1
Program Loading 4.4
Programming Registers 3.3

Qualifiers 1.2, 1.3.6

Rate, Sampling 1.2
RBUF Routine 3.3.8, F3-10
Recorder, Logic State 1.2
Registers, Programming 3.3
Replacing Probe Leads 2.4, F2-7
Routines:
  GSTAT 3.3.2, F3-4
  RBUF 3.3.8, F3-10
  RREG 3.3.5, F3-7
  STOP 3.3.7, F3-9
  SSTAT 3.3.3, F3-5

Sampling Routine 1.2
Selection, Clock 1.3.5
Serial Registers, Loading and Reading F3-3
Signals, Interface 3.2.3, T3-3
SSTAT Routine 3.3.3, F3-5
STOP Routine 3.3.7, F3-9
STRT Routine 3.3.6, F3-8
System:
  Block Diagram F1-3
  Configuration 1.2, F1-2

Target Data Inputs 1.3.3, F1-3
Termination Modes 1.3.11
Test Point, Event 4.3.1
TMF LED 4.3, F4-1
TRACE LED 4.3, F4-1
Trace/Emulator Cabling F2-2
Trace Module:
  Cabling F2-2
  CRU Interface 3.2
  Interrupt 3.2.4
Trace RAM 1.3.7
  Flowcharts F3-11, F3-12

Unpacking 2.2

WREG Routine 3.3.4, F3-6