Model 990/10A Computer

Texas Instruments

BUSINESS SYSTEM SERIES

MANUAL REVISION HISTORY

Preface

This manual provides information about the Model 990/10A Computer and is directed to both the installation personnel and the end-user.

Information in this manual is divided into the following sections:

Section

1   General — Contains physical and functional descriptions that acquaint the user with the hardware components and capabilities of the Model 990/10A Computer.

2   Installation — Outlines procedures for unpacking the Model 990/10A Computer from its shipping container and setting the option jumpers and switches.

3   Operation — Describes the indicators on the front panel and PWB and provides general operating procedures.

4   Programming — Presents information for use by programmers on the function of the processor and the asynchronous communication controller.

Appendix

A   Using the Multiprocessing Interface — Contains information on system configuration and programming the multiprocessing interface.

The following documents contain additional information related to the Model 990/10A Computer.

Title	Part Number
Model 990/10A Computer Maintenance Manual — Field Theory and Maintenance	2302634-9701
Model 990/10A Computer Maintenance Manual — Depot Theory and Maintenance	2302635-9701
Model 990 Computer 990/10 and 990/12 Assembly Language Reference Manual	2270509-9701
Model 990 Computer Diagnostics Handbook	945400-9701
Model 990A13 Chassis Maintenance Manual — General Description	2308774-9701
ROM Loader User's Guide	2270534-9701

Contents

2 — Installation

3 — Operation

Appendix

Index

Illustrations

Tables

General Description

1.1 GENERAL

This manual provides unpacking, installation, testing, and operating information for the Texas Instruments Model 990/10A Processor (Figure 1-1). This section contains functional and physical descriptions to acquaint the user with the hardware components and capabilities of the Model 990/10A Processor, hereafter referred to as the 990/10A.

1.2 PURPOSE OF EQUIPMENT

The 990/10A is a single board replacement for the 990/10 processor. The 990/10A combines a 990/10-compatible processor with a maximum of 512 kilobytes of error-correcting memory, an asynchronous communications port, TILINE* interface, communications register unit (CRU) interface, and control logic for operation in a multiprocessor system. When combined in a chassis with appropriate peripherals for input, output, and mass storage, the 990/10A forms part of a stand-alone minicomputer system.

1.3 EQUIPMENT DESCRIPTION

The 990/10A is a full-function processor on a single printed wiring board (PWB). Much of the discrete circuitry of the 990/10 is incorporated into custom large scale integration (LSI) devices in the 990/10A, resulting in lower cost and power consumption with increased reliability. The speed increases resulting from the single-board design of the 990/10A give an average 1.5 times increase in throughput over the 990/10, for most applications. The following paragraphs describe the major components of the 990/10A.

* Trademark of Texas Instruments Incorporated.
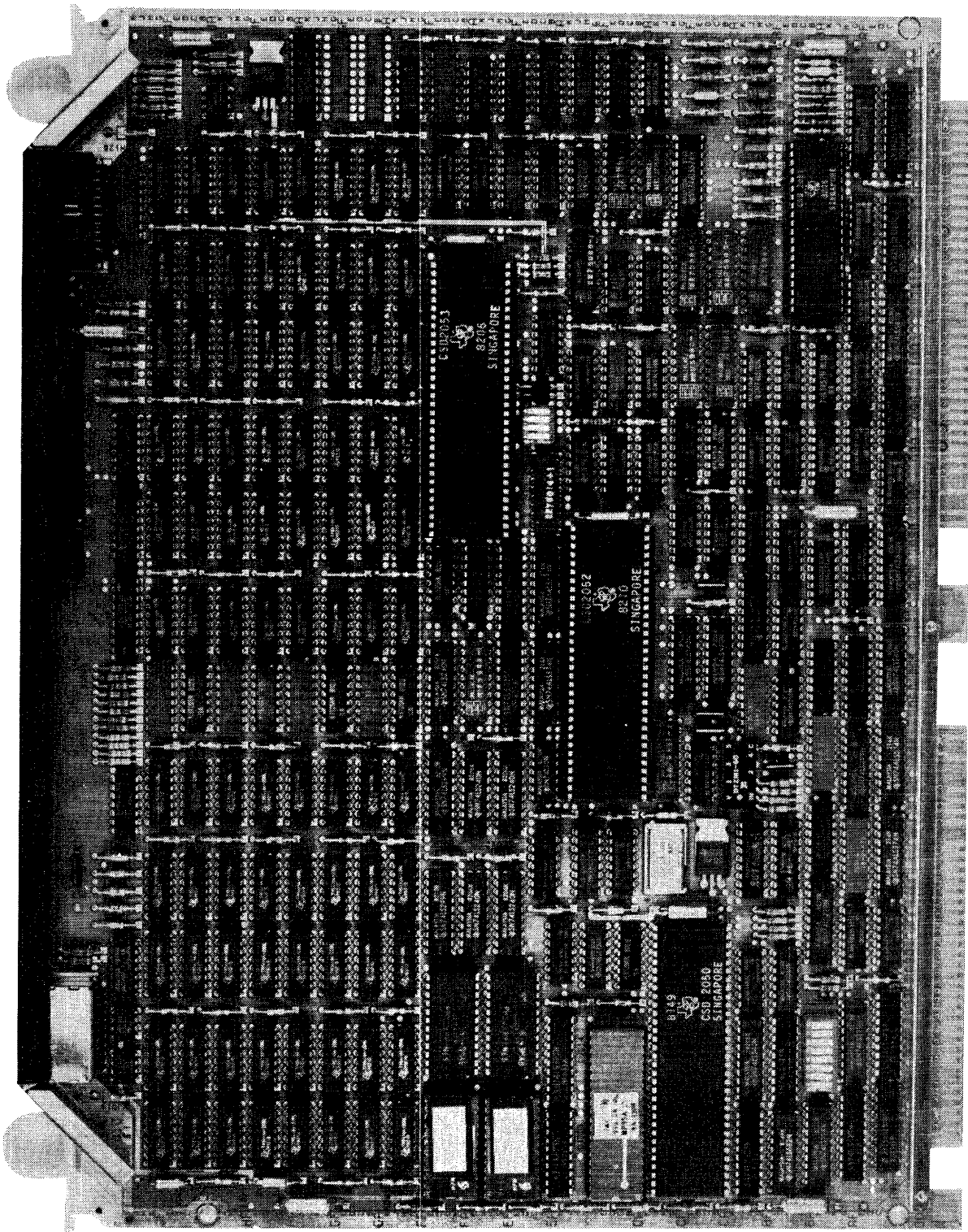
2283230

Figure 1-1.   Model 990/10A Processor Board

1.3.1 Microprocessor

The TMS 99000 microprocessor used on the 990/10A is an upwardly compatible, high performance member of the TMS 9900 family. This microprocessor implements the 990/10 instruction set and an additional five instructions, providing the capability of a full minicomputer on a single chip.

The following characteristics are offered by the microprocessor.

- Sixteen-bit instruction word

- Memory-to-memory architecture

- 77 instructions including signed multiply and divide

- Multiple register files resident in memory

- 16 prioritized hardware interrupts

- 16 software interrupts

- Programmed input/output (I/O) and direct memory accessing (DMA) capability

- I/O operation addressable in bit, byte, or word increments via the CRU

- Multiprocessor system interlock signal

- Privileged instructions

- User and supervisor modes

1.3.2 Memory

The 990/10A can contain a maximum of 512 kilobytes of on-board error-correcting memory. This memory is available in options of 256 kilobytes and 512 kilobytes using two or four rows of 64 kilobit dynamic random-access memory (DRAM) devices. Four pencil switches select the lower bound of on-board memory in 128-kilobyte increments. This memory can be accessed by the microprocessor or by any device on the TILINE bus.

1.3.2.1 Memory Correction and Control. The correction and control chip provides error detection and correction capability for the on-board memory. This chip generates a six-bit checkword with each memory write. On subsequent memory reads, the checkword is used to correct single bit errors and to flag double bit errors.

1.3.2.2 Memory Mapping. The memory mapping chip performs a mapping algorithm to convert 15-bit word level addresses (logical addresses) into 20-bit addresses (physical addresses) for the internal memory bus.

In addition to generating the physical address, the memory mapping chip detects an attempt to access memory addresses larger than the value stored in the highest limit register. Logic external to the memory mapping chip generates an error interrupt, preventing out-of-range addressing. This external logic also detects accesses to peripheral controllers and maps these accesses to the TILINE peripheral control space (TPCS).

1.3.3 Input/Output (I/O)

The following paragraphs describe the interface controllers on the 990/10A that provide input/output capability. These controllers are also used for many of the on-board data and control word transfer operations of the processor. Both TILINE and CRU controllers are provided, as well as one on-board asynchronous communication port.

1.3.3.1 TILINE Control. The TILINE is a high-speed, asynchronous interface bus that provides transfer rates approaching the rates of DMA interface controllers, but does not require the complex controllers for each device. The 990/10A supports all existing TILINE memory and peripheral devices. TILINE devices can access memory contained on the 990/10A board as well as off-board TILINE memory. The TILINE controller consists of a master controller and a slave controller along with synchronization and interface circuits common to both.

1.3.3.2 CRU Interface. The 990/10A provides the CRU interface that is standard on 990 family processors, in a custom integrated circuit. This versatile, command-driven interface can effectively handle a wide range of control and data transfer operations with its extremely flexible format. The CRU can set, reset, or test a field of one to sixteen bits in the CRU array. The CRU operates at a 2.5 megahertz data rate with devices in the main chassis, and at a 1.67 megahertz rate on CRU reads from devices in an expansion chassis. CRU writes to expansion chassis devices occur at 2.5 megahertz. The use of an expansion chassis in the system does not reduce the rate for main chassis communication. In addition to the CRU control function, the CRU controller performs the following functions:

- Bus status decode

- Single instruction execution control

- Run mode logic

- System error latch

- Front panel test logic

- Real-time clock state controller

- Ready generation to the microprocessor

1.3.3.3 Asynchronous Communication Controller. The asynchronous communication controller provides an interface port for one video terminal, printer, or any other EIA compatible device. The asynchronous communication controller uses the TMS 9902 chip to provide the following:

- Interrupt control

- Timing control between the microprocessor and an I/O device

- Asynchronous operation

- Character length of five to eight bits

- Fully programmable data rate

- Interval timer

- Even, odd, or no parity

- 1, 1.5, or 2 stop bits

1.4 SPECIFICATIONS

Table 1-1 lists the physical, electrical, and environmental specifications for the 990/10A board.

Table 1-1.   Physical, Electrical, and Environmental Specifications

Characteristic	Specification
Ambient temperature (operating)	0 to 65 degrees C (32 to 149 degrees F) (Derate 2 degrees C (3.6 degrees F) for every 761 meters (2500 feet) of elevation)
Ambient temperature (storage)	–40 to 70 degrees C (–40 to 158 degrees F)
Ambient humidity (operating)	0 to 95% (noncondensing)
Ambient humidity (storage)	0 to 95% (noncondensing)
Shock	2-inch vertical drop (installed in chassis)
Vibration	1 G (5 to 80 Hz) (installed in chassis) 0.3 G (80 to 500 Hz) (installed in chassis)
Power	30 watts total
Current	
+ 5V main	4.8 A
+ 5V memory with 256K bytes	.94 A (operating) .40 A (standby)
with 512K bytes	1.1 A (operating) .55 A (standby)
+ 12V main	26 mA (EIA drivers)
– 12V main	– 26 mA (EIA drivers)

1.5 FUNCTIONAL DESCRIPTION OF EQUIPMENT

The following paragraphs describe the function of the major components of the 990/10A computer.

1.5.1 Microprocessor

The microprocessor is the principal component of the 990/10A computer. The microprocessor implements the 990/10 instruction set and five additional instructions. These additional instructions are compatible with the 990/12 instruction set. It recognizes interrupts, initiates a context switch in response to interrupts, and stores the present context for a return. The microprocessor provides the interrupt mask function, and implements the nonmaskable interrupt (NMI) function for front panel interrupts. The microprocessor controls all CRU operations, both serial and parallel. The microprocessor also features a multiprocessor interlock signal to facilitate multiprocessor operation, and provides privileged instruction capability.

The microprocessor communicates with the rest of the on-board computer functions by means of a synchronous bus. This bus has no external connection off the 990/10A board, but is synchronized with the TILINE bus by means of the TILINE control circuits. Use of the synchronous bus enhances on-board data transfer operations, but sacrifices performance for TILINE communication. For this reason, on-board memory should be used for maximum performance operation.

1.5.2 Memory Mapping

The map logic is contained in a custom LSI circuit. This chip performs the standard 990 family mapping algorithm to convert 15-bit word level memory addresses from the microprocessor (logical addresses) into 20-bit addresses (physical addresses) for the memory and memory-mapped I/O devices. It is the physical address that is put on the internal memory bus and on the TILINE bus if the TILINE master logic decodes the address as not being resident on the board.

The logical address generated by the microprocessor is expanded into the physical address space by relocating the logical address into blocks of memory in the two-megabyte address space. The displacement added to the logical address is defined by values stored in the mapping registers.

In addition to generating the physical address, the map chip has error logic that detects an attempt to access memory above the largest limit register. Logic external to the chip generates an interrupt and prevents further accesses when an error is detected. The map chip also detects access to the TPCS and maps these accesses to the high order addresses.

1.5.2.1 Map Operation. The map logic consists of three sets of map files. Status register bit 8 selects either map file 0 or map file 1. Map file 2 is selected by the use of a long distance source (LDS) or long distance destination (LDD) instruction. Each map file contains six registers: three 16-bit base registers and three 11-bit limit registers. Mapping is performed by selecting one of the base registers and adding the contents of that register to the high order 11 bits of the logical address. (Note that the low order four bits of the word level logical address are not affected by the map logic.) Within a given map file, the base register to be used is selected by comparing the logical address to the limit registers for that map file. A logical address that is less than or equal to the value of limit register 1 is mapped with base register 1; an address that is greater than the value of limit register 1 but less than or equal to the value of limit register 2 is mapped via base register 2; and a logical address that is greater than the value of limit register 2 but less than or equal to the value of limit register 3 is mapped via base register 3. If the logical address is greater than all three limit registers, an error condition is flagged. The limit registers accept and hold the ones complement of the desired compare data.

1.5.2.2 Map Initialization. On power-up, bit 8 of the status register is set to zero. Mapping is disabled so that the first 31K words of memory are addressed. This memory may or may not be on-board the 990/10A depending upon the setting of the lower-bound address switches.

NOTE

A 990/10A operating in the auxiliary mode will only service certain interrupts as specified in paragraph A.2.2. If these interrupts are to be handled out of local memory, the map chip must be set up to map all interrupts into the on-board address space as determined by the memory bound switches. See Appendix A for additional information on auxiliary mode operation.

NOTE

The "greater than" symbol (>) is used to represent hexadecimal numbers.

1.5.2.3 TILINE Peripheral Control Space (TPCS) Mapping. The TPCS consists of those logical addresses in the range of > F800 through > FBFE, and is reserved for assignment to peripheral device controllers. These addresses are modified before presentation to the TILINE. Five address bits are appended to the left (MSB) side of each address to form a 20-bit TILINE word address. In other words, addresses > F800 through > FBFE are mapped to addresses > FFC00 through > FFDFE. This particular mapping occurs only when map file zero is invoked (Status register bit 8 equals 0) or if mapping is disabled. Figure 1-2 illustrates the TPCS concept.

NOTE

Note from Table 1-2 that DRAM addressed in the 128-kilobyte increment from > F0000 to > FFFFF overlaps the TPCS. The 990/10A includes logic to prevent accessing both memory and control registers when its lower-bound memory address switches and amount of memory jumpers permit addressing memory in this range. However, TILINE expansion memory should be limited so that memory does not respond to TPCS addresses. (In other words, lower-bound address switches on TILINE memory controllers should not be set so that > FFC00 through > FFFFF are valid addresses.)
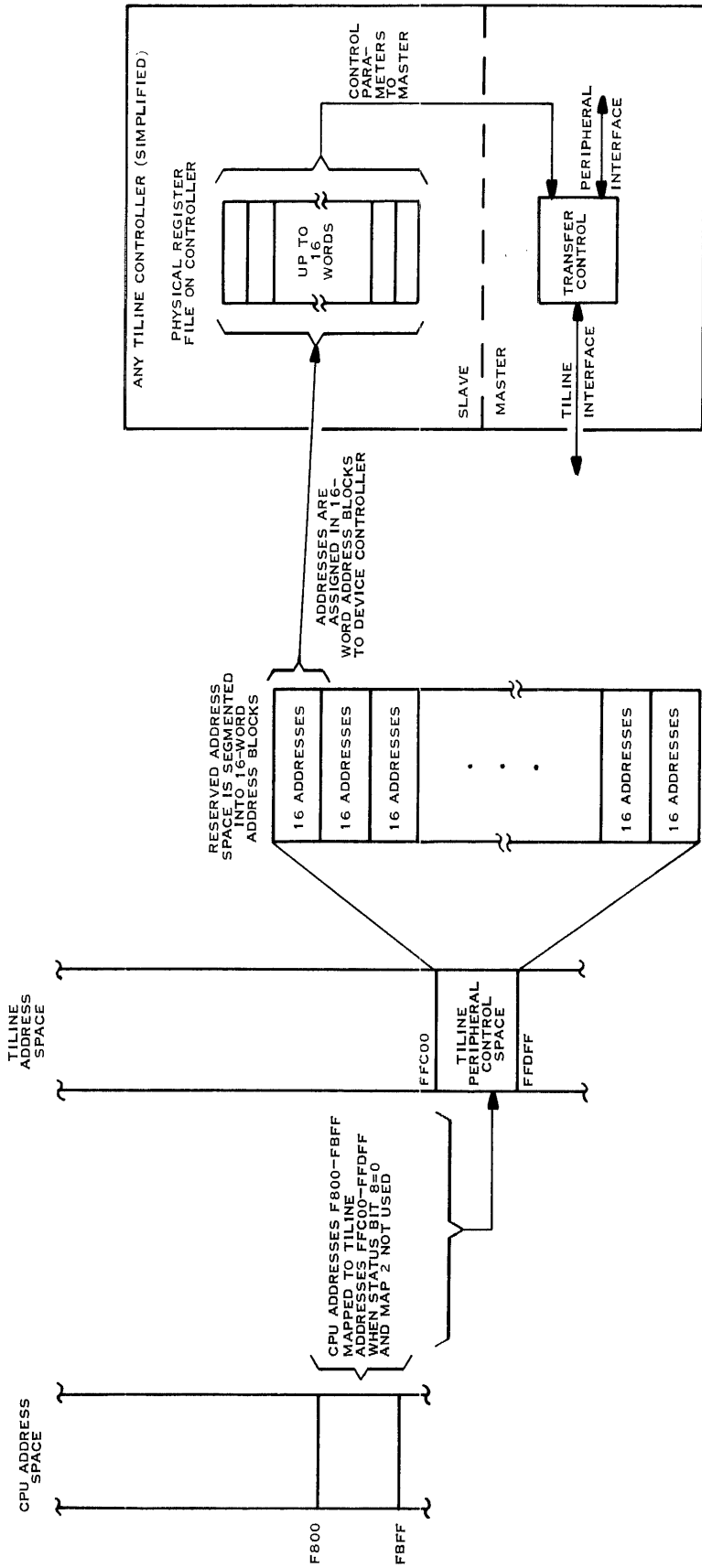
Figure 1-2. TILINE Peripheral Control Space Implementation

1.5.2.4   Scratch Pad RAM.   During self-test, the 990/10A requires scratch pad memory area for use as workspace registers, temporary storage, etc. Therefore, a 128 x 16 random-access memory (RAM) is implemented which can be accessed during self-test only. To use the scratch pad RAM, a latch must be set by writing a one to bit two of the diagnostic register (CRU address > FAFE). After that, any subsequent access to logical memory address > FA00 through > FAFF is directed to the scratch pad RAM. The scratch pad RAM is disabled by writing a zero to bit two. The scratch pad RAM is not accessible from the TILINE.

1.5.2.5   Loader/Self-Test ROM Mapping.   The loader/self-test read-only memory (ROM) contains a total of eight kilobytes of loader/self-test code. The 990/10A has the capability to directly access one kilobyte of ROM code only. Therefore, the 990/10A uses an addressing scheme that is compatible with the 990/12 to allow the access of an additional seven kilobytes of loader/self-test code. This is accomplished by dividing the loader/self-test ROM into eight pages, with each page containing one kilobyte. From Table 1-2, note that one kilobyte is the total allowed by the address allocation. The page that is being addressed is then determined by three latched bits written into a register at memory address >FAFE, not by the address field. This address normally is within the TPCS and initiates a TILINE cycle. Therefore, the ROM page address field is qualified to be accessible only when the scratch pad RAM is enabled. The ROM page address field then appears as the last word of the scratch pad RAM memory space (>FA00 – >FAFE). On power-up, loader self-test page zero is selected. The page selected is changed by setting the scratch pad RAM enable bit at *CRU address* >FAFE bit 2 and then writing to bits 13, 14, and 15 of *memory address* >FAFE. The transition from page to page is based on the principle that the microprocessor prefetches the next instruction before accomplishing the destination write from the present instruction. An example instruction sequence follows:

```
LI      R5,@NPEP        LOAD NEW PAGE ENTRY POINT
INC     R4              INCREMENT PAGE COUNTER
MOV     R4,>FAFE        PERFORM THE PAGE SWAP AFTER PREFETCH
B       *R5             BRANCH TO ENTRY POINT
```

1.5.3   Memory Correction and Control
The correction and control chip is a custom LSI integrated circuit that performs the error detection and correction algorithm employed on other 990 memory products. This chip generates a six-bit checkword with each memory write. On subsequent memory reads, the checkword is used to correct single-bit errors and to flag double-bit errors. The error condition of all logic ones or all logic zeros from memory is detected, but other errors of three or more bits may not be detected. An on-chip register that responds to a TPCS address permits control of the error-correcting function for diagnostic purposes. The error correction can be disabled to certain rows of memory chips, or the checkword can be interchanged with the six most significant bits (MSBs) of the data word to verify the cause of a failing condition. Status information about the test results, memory size, memory address, and chip-in-error can be read through two 16-bit registers (the memory error log) in the TPCS whose address is selected by pencil switches on the 990/10A board.

The correction and control chip also includes two addressable registers for controlling interrupts among processors in a multiprocessor system whose addresses are also switch selectable.

1.5.4 Memory Allocation

Table 1-2 illustrates the allocation of the 990/10A memory.

Table 1-2. 990/10A Typical Memory Allocation

Description	Logical Address (Hexadecimal)	Physical Address (Hexadecimal)	Memory Content
Interrupt vectors	0000 0002 0004 : : 003C 003E	00000 00001 00002 : : 0001E 0001F	WP level 0 interrupt PC level 0 interrupt WP level 1 interrupt : : WP level 15 interrupt PC level 15 interrupt
XOP software trap vectors	0040 0042 : : 007C 007E	00020 00021 : : 0003E 0003F	WP XOP 0 PC XOP 0 : : WP XOP 15 PC XOP 15
	0080 : : F7FE	00040 : : 07BFF	62K* bytes of general purpose on-board memory
General purpose memory	F800 : : FFFE	07C00 : : 07FFF	2K* bytes of on-board RAM not available to processor through map file 0
		08000 : FFBFF	Expansion memory
TILINE peripheral control space	F800 : : FBFE	FFC00 : : FFDFF	TILINE peripheral control registers or scratch pad memory (see text)
Loader and self-test programs	FC00 : : FFFA	N/A : : N/A	Loader/self-test ROM (accessible by the processor only)
Load vector	FFFC FFFE	N/A N/A	WP load function PC load function

Note:

* K equals 1,024 bytes.

1.5.5 Interrupts

The 990/10A uses 16 priority-vectored interrupt levels. The priority ranking system assigns numbers to the levels from 0 (highest priority) to 15 (lowest priority) in order to resolve interrupt conflicts. The interrupts are also maskable via bits 12–15 of the status register. Upon detection of a pending interrupt, the processor compares the interrupt code with the interrupt mask. If the level of the interrupt is less than or equal to the mask level (higher or equal priority), the processor recognizes the interrupt and initiates a context switch after completion of the currently executing instruction. The processor fetches the new context workspace pointer and program counter from the appropriate interrupt vector location and stores the previous workspace pointer, program counter, and status register contents in workspace registers 13 through 15 of the new workspace. The processor then loads the interrupt mask with a value that is one less than the value of the interrupt being processed (except for a level 0 interrupt which loads 0 into the mask). This allows only interrupts of a higher priority to interrupt the service routine. The processor also inhibits additional interrupts until the first instruction of the service routine has been processed. All interrupts are tested during prefetch of the next instruction. Interrupts are sampled beginning at the first state of the prefetch cycle until one state prior to the end of the destination write cycle.

In addition to the interrupts already discussed, there are two interrupts that are not affected by the interrupt mask. One is referred to as the NMI. This is a level –1 interrupt that vectors through locations > FFFC and > FFFE. This is normally used for front panel control. The other is an illegal operation code interrupt that is normally a level 2 interrupt; but, on the 990/10A, it cannot be masked out via the status register. This allows software emulation of instructions not defined on the 990/10A microprocessor.

1.5.5.1 Level 0 Interrupt (Power Restored).

The highest priority level is reserved for the power restored interrupt. When power is restored, TILINE power reset (TLPRES–) from the power supply remains true long enough for the power supply to stabilize, and then goes false. TLPRES– true (RESET) forces the microprocessor to cease instruction execution. When TLPRES– goes false after being true, the microprocessor initiates a level 0 interrupt trap (power restored). This level 0 interrupt trap never completes on the 990/10A.

When the microprocessor recognizes RESET (TLPRES– true), it sends the RESET bus status code until the level 0 trap is initiated. The single instruction execute controller in the CRU chip recognizes this RESET bus status code and forces an NMI. The result is that the level 0 interrupt context switch completes (first instruction fetched), but this first instruction is not executed because an NMI is present and the NMI interrupt trap is initiated.

The NMI handler in ROM (see paragraph 1.5.5.8) then runs self-test because the NMI occurred due to power restored (or power-up). Following a successful completion of self-test, under the right circumstances, a level 0 interrupt context switch then occurs. See the power-up NMI paragraph in this section for a detailed explanation of what happens on power-up.

1.5.5.2 Level 1 Interrupt (Power Failure Imminent).

When the chassis power supply senses a loss of power, a level 1 interrupt is generated. The hardware has several milliseconds of processing time from the time the interrupt signal is asserted until a master clear (TLPRES–) is issued.

1.5.5.3 Level 2 Interrupt (Error Conditions). A level 2 interrupt can be caused by error conditions external to the microprocessor as well as by internal microprocessor chip errors. The external error conditions are memory data error, memory map error, and TILINE time-out. The internal microprocessor chip level 2 interrupts include arithmetic overflow interrupt and the privileged opcode interrupt as well as the macroinstruction decode trap.

The macroinstruction decode signal is used to indicate an illegal opcode in the 990/10A. Upon detection of an undefined opcode, the microprocessor chip traps (subroutine call) using the workspace pointer (WP) and program counter (PC) stored in locations > 0008 and > 000A. These locations contain the level 2 interrupt vectors.

The 990/10A implementation differs from the 990/10 with regard to level 2 interrupts. The macroinstruction decode trap is not maskable with the interrupt mask logic and therefore overrides any other interrupt. Also, the trap does not change the interrupt mask.

A flag indicating which condition caused the level 2 interrupt is stored in the error status register.

Error Status Register. The error status register stores error conditions and provides a composite output to the level 2 interrupt of the microprocessor. The error status register is read through the CRU at CRU base address > 1FC0. Table 1-3 gives the bit definitions for the error status register.

Table 1-3. Error Status Register CRU Bit Definitions

	CRU BASE 1FC0 (HEX)	
ERROR CONDITION	INPUT BIT	OUTPUT BIT
TILINE TIMEOUT	15	15
PRIVILEGE VIOLATION	14	14
ILLEGAL OPERATION CODE	13	13
MEMORY ERROR	12	12
MAPPING ERROR	11	11
ARITHMETIC OVERFLOW	4	4
RESERVED	3	
RESERVED	2	
ID (990/10A = 1)	1	
ID (990/10-990/12 INDICATOR)	0	

```
        BIT1  BIT0
         0     0    990/10
         1     0    990/10A

         0     1    990/12
```

1.5.5.4   Level 3 Host Processor Interrupt.   Logic is included in the 990/10A design to allow multiple processor boards to operate together in a single chassis with the processor in slot 1 acting as a host processor. This requires attention and acknowledge interrupts between the host processor and the auxiliary processors. The host processor can interrupt the auxiliary through a TPCS register. The auxiliary can acknowledge that interrupt by generating an interrupt back to the host. In a similar fashion, the auxiliary can generate an attention interrupt to the host and the host can acknowledge by setting a bit that interrupts the auxiliary. Either of these interrupts can be masked on the remote processor through the TPCS registers.

1.5.5.5   Level 5 or 15 Interrupt (Real-Time Clock).   A line frequency synchronized oscillator on the power supply is an input to the 990/10A processor. On every cycle of the oscillator (either 8.33 or 10.0 milliseconds depending on the line frequency) the real-time clock interrupt signal is generated. This signal can be connected by a jumper on the processor board to interrupt level 5 or level 15, or it can be disconnected. The clock on (CKON) and clock off (CKOF) instructions enable or disable the real-time clock interrupt independently of the interrupt mask.

1.5.5.6   Level 8 (EIA Port Interrupt).   The interrupt from the EIA port on the processor board is jumpered to level 8. This interrupt is enabled or disabled by setting a specified CRU output bit and is disconnected by removing a jumper on the processor board (see Section 2 for jumper descriptions).

1.5.5.7   External Interrupts.   Interrupt requests from other boards can be wired by backpanel jumpers to any of the 13 interrupt request lines (levels 3 through 15). These lines form 13 separate wired-OR interrupt buses. Each interrupt request signal must be an active low signal driven by an open collector TTL gate. The request signal must remain active until it is reset by software communication.

When operated in a slot other than slot 1, only the internally generated interrupts can, in general, be received by the processor. There is one exception that might be exploited for some applications. That exception occurs because the backpanel connector P2 pin 66 is normally an output for TILINE or CRU devices to send interrupts to the interrupt jumper plug. This pin, on a 990/10A is actually an input to interrupt level 14. Therefore it is possible, by modifying the backpanel jumper plug, to direct one external interrupt into the 990/10A at level 14 while operating as an auxiliary processor.

When in slot 1, all interrupts are received by the processor. Those interrupts that are allotted to internal processor board functions share a particular wired-OR interrupt level with the external interrupts unless disconnected by removing the jumper.

1.5.5.8   Nonmaskable Interrupt (NMI).   The function referred to as the nonmaskable interrupt is equivalent to the load function in other 990 family microprocessors. NMI cannot be disabled by the interrupt mask logic and is therefore always executed as soon as the current instruction is completed. NMI causes execution to begin with the WP at location > FFFC and the PC at location > FFFE. NMI is activated by the restart signal. This signal can result from the execution of an LREX instruction, from the HALT switch on the front panel, or from power-up.

When an NMI occurs, the NMI handler in the loader/self-test ROM determines the sequence that follows the NMI. This sequence depends on whether the central processing unit (CPU) is a host or auxiliary, a front panel is present, and the NMI occurred as a result of power-up. A flowchart of the NMI handler is shown in Figure 1-3.
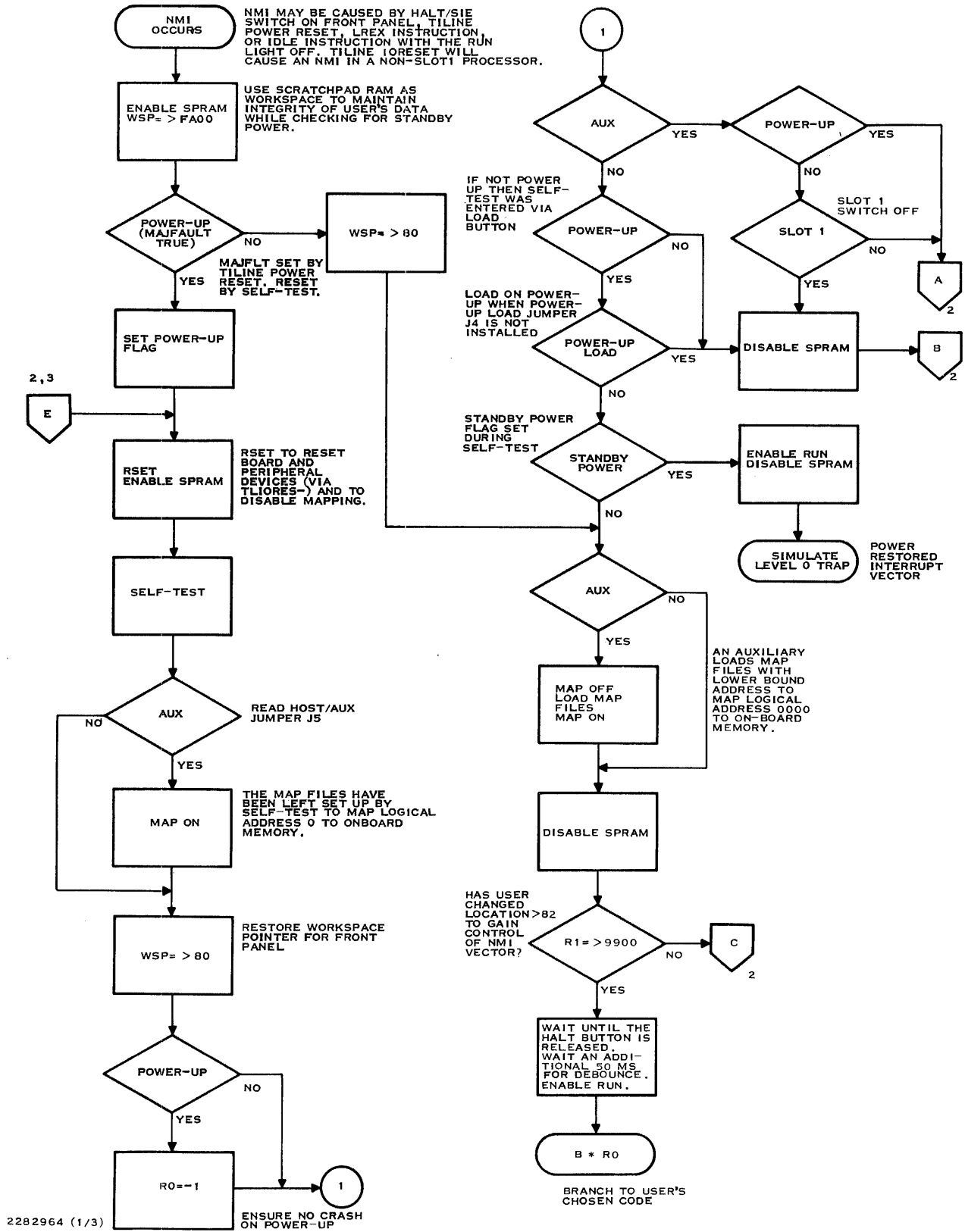
Figure 1-3.   Flowchart of Nonmaskable Interrupt and Front Panel Code (Sheet 1 of 3)

2302633-9701
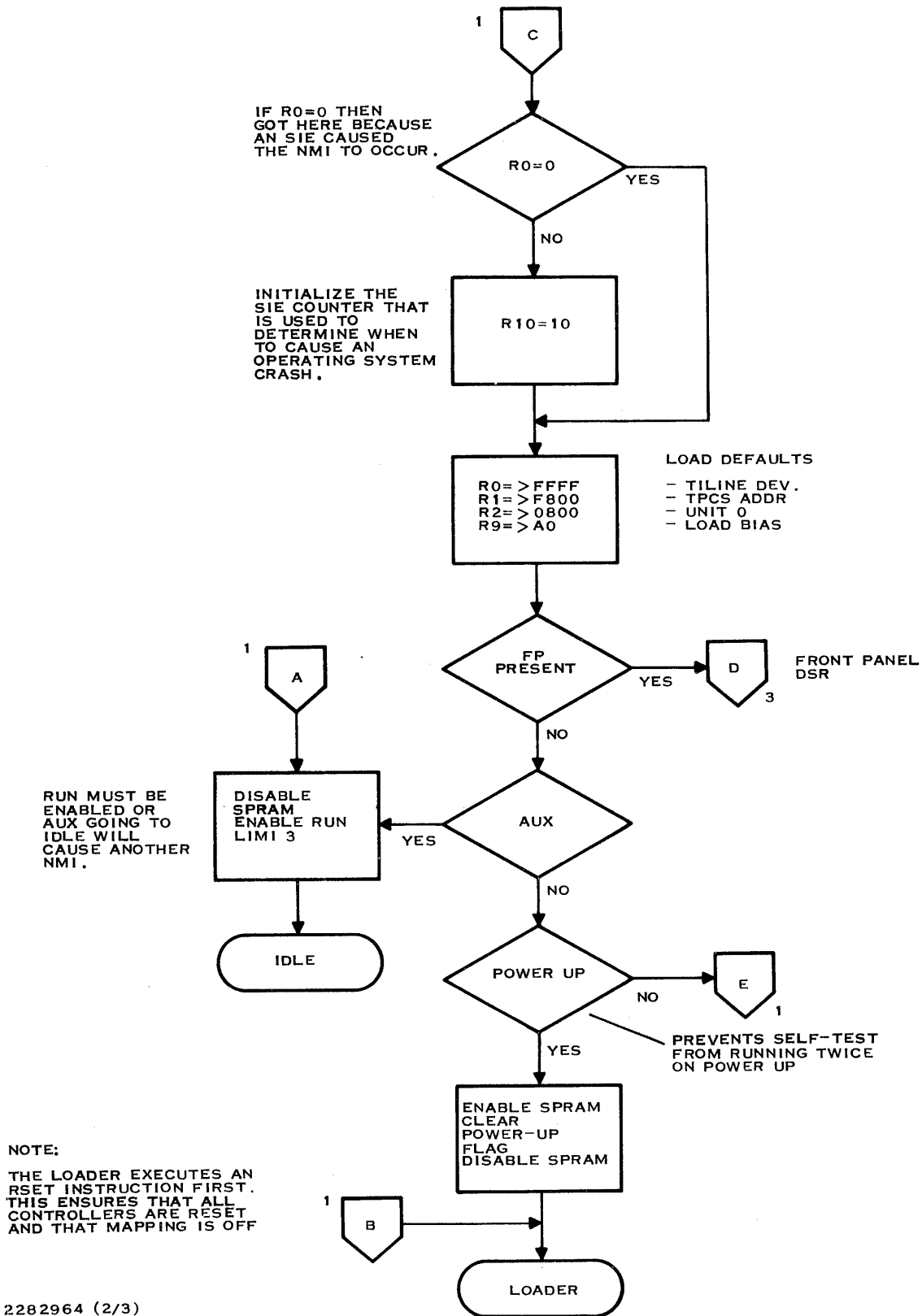
Figure 1-3. Flowchart of Nonmaskable Interrupt and Front Panel Code (Sheet 2 of 3)
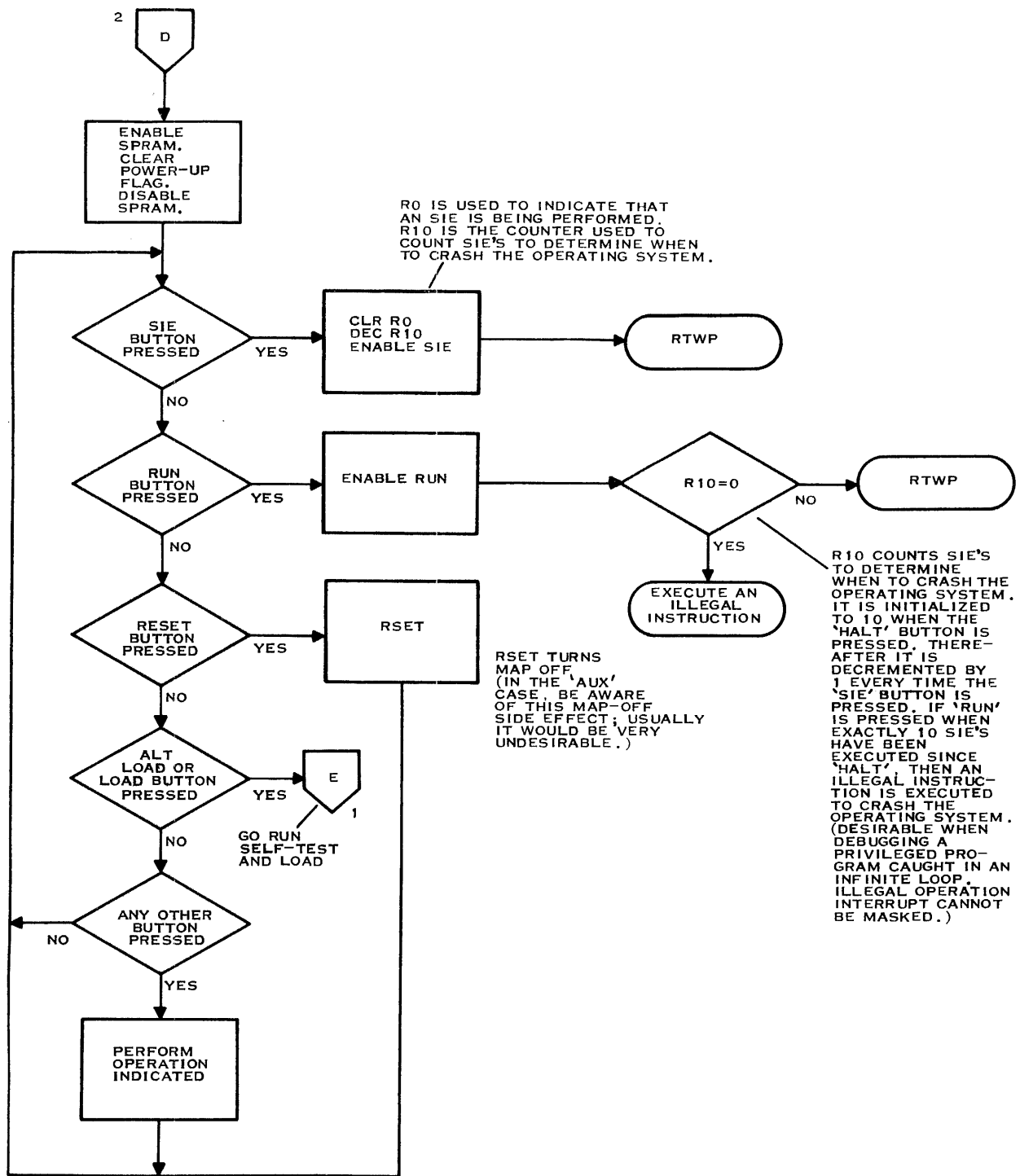
Figure 1-3.   Flowchart of Nonmaskable Interrupt and Front Panel Code (Sheet 3 of 3)

Power-up NMI. Upon power-up, self-test is always run first. After successful completion of self-test, the power-up load jumper is examined. If the jumper is not installed, the loader is executed. If the jumper is installed, the host/auxiliary jumper is examined. If the CPU is a host and has battery backup, a level 0 interrupt trap (power restored) is taken. If the CPU is a host but has no battery backup, a load is performed unless a front panel is present. If a front panel is present, front panel code is executed.

If the CPU is an auxiliary, map 0 is set up to map logical address 0 to the on-board memory lower-bound address and mapping is enabled. The CPU then idles after executing a LIMI 3 to enable the multiprocessor interrupt.

Nonpower-up NMI. An NMI occurring from some condition other than power-up always results in front panel code execution if a front panel is present. If no front panel is present, then a load is performed if the CPU is a host, or a LIMI 3, IDLE if the CPU is an auxiliary. The front panel code operates the same for both hosts and auxiliaries.

1.5.6 Communication Register Unit (CRU) Controller

The 990/10A is compatible with all standard CRU interfaces, but allows 32,768 output bits and 32,768 input bits, instead of the 4096 allowed by previous processors. In addition, multiple bit transfers are made in parallel mode on the 990/10A board. This provides high-speed transfers to the address mapping chip and diagnostic hardware. Parallel transfers off the processor board are not supported. Only the 12 low order CRU bits are available off the 990/10A board, and CRU addresses for off-board functions are compatible with all existing 990 family hardware and software.

NOTE

The 990/10A uses the three MSBs of the CRU address field for internal addressing and any attempt to issue a CRU access to an address with any of the three MSBs of register 12 set to a logic one may cause erroneous results.

Both input and output bits can be addressed individually or in fields of from 1 to 16 bits. The processor employs three dedicated I/O pins (CRUBITIN, CRUBITOUT, and STORECLK–) and 12 bits (CRUBIT4 through CRUBIT15) of the address bus at the interface to the CRU system. The processor instructions that drive the CRU interface can set, reset, or test any bit in the CRU array or move data between memory and CRU data fields.

The processor performs three single-bit CRU functions: test bit (TB), set bit to one (SBO), and set bit to zero (SBZ). To identify the bit to be operated on, the processor develops a CRU-bit address and places it on the address bus, CRUBIT4 through CRUBIT15.

For the two output operations, SBO and SBZ, the processor generates a STORECLK– pulse that indicates to the CRU device that the operation is an output operation, and places bit 7 of the instruction word on the CRUBITOUT line to accomplish the specified operation (bit 7 is a one for SBO and a zero for SBZ). The TB instruction is an input operation that transfers the addressed CRU bit from the CRUBITIN input line to bit 2 (equal bit) of the status register.

The processor develops a CRU-bit address for the single-bit operations from the CRU base address contained in workspace register 12 (R12) and the signed displacement contained in bits 8 through 15 of the instruction. The displacement allows two's complement addressing from base minus 128 bits through base plus 127 bits. The base address from R12 is added to the signed displacement specified in the instruction, and the result is loaded onto the address bus. Figure 1-4 illustrates the development of a single-bit CRU address.

When the processor is in slot 1, CRU write operations are executed in two clock cycles whereas CRU read cycles are executed in either two or three cycles. If the address to be read is in the expansion chassis or the expanded CRU address space (above > 1FFE), the read operation takes three cycles. If the address lies in the main chassis, it takes two cycles. If the 990/10A is not in slot 1, all CRU cycles are three cycles. The lower rate for expansion chassis operations allows for signal propagation in the interconnecting cable.
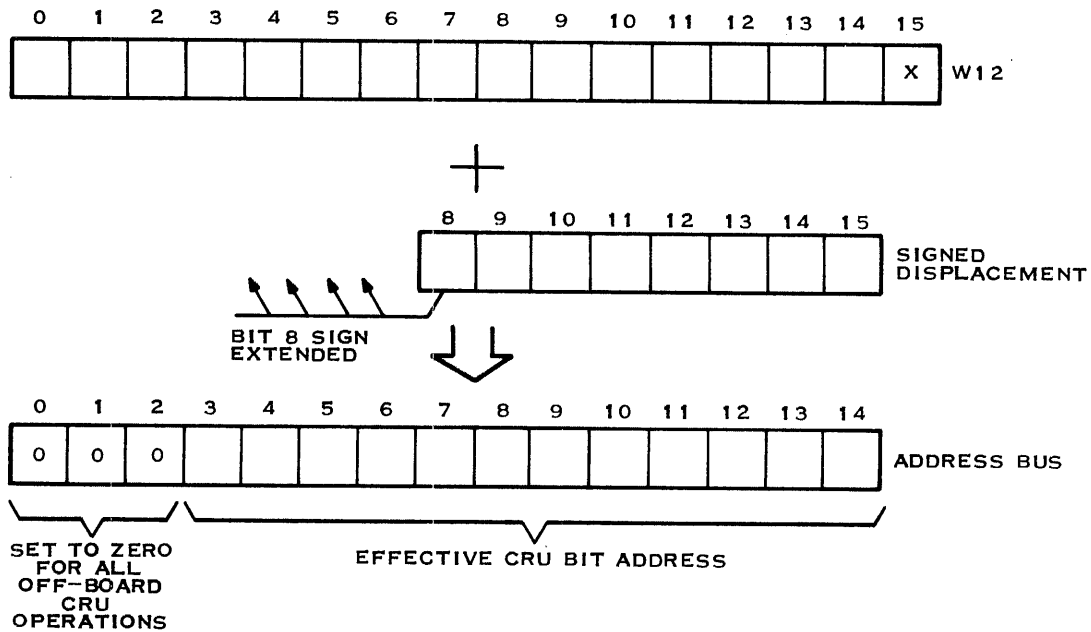
2283231

Figure 1-4.   Single-Bit CRU Address Development

1.5.6.1 Parallel CRU Operations. The 990/10A has the ability to perform multiple bit CRU operations in a parallel mode. When the MSB of the 15-bit CRU address used with a LDCR or STCR instruction is set to one, the CRU transfer takes place in two or three clock cycles with the data transferred in parallel via the on-board data bus. This mechanism is used in the 990/10A to provide high-speed transfers to the map chip and other diagnostic hardware. No provision is made to support parallel CRU transfers off the processor board.

1.5.6.2 CRU Addressing. Table 1-4 shows a map of the CRU addresses assigned for the 990/10A. In this table, the addresses are referenced by the R12 value required to address them.

Table 1-4. CRU Address Map

Bit Serial Address Space
(MSB = 0)

CRU Base	Contents
0000–02FE	Main chassis addresses
0400–06FE	Expansion chassis #1
0800–0AFE	Expansion chassis #2
0C00–0EFE	Expansion chassis #3
1000–12FE	Expansion chassis #4
1400–16FE	Expansion chassis #5
1700–173E	Asynchronous EIA port
1740	Output: data terminal ready
1742	Input: ring indicator
	Output: analog loopback
174C	Interrupt enable EIA port
174E	EIA port comm enable
1800–1AFE	Expansion chassis #6

——— Write operations to addresses above 1C00 are privileged ———

CRU Base	Contents
1C00–1EFE	Expansion chassis #7
1EC4–1ECE*	External instructions
1FA6	Map enable
1FBF	CRU chip loopback mode
1FC0–1FDE	Error status register
1FE0–1FFE	Front panel
7800–781E	Diagnostic loopback

Bit Parallel Address Space
(MSB = 1)

CRU Base	Contents
9F80–9FFF	Map chip registers
FAFE	Input: self-test register
	Output: diagnostic control

Note:

* Overlaps chassis 7 addresses.

1.5.7 Backpanel Interface

All connections to the computer chassis and peripheral devices (other than the front panel and EIA port) are through the backpanel connector. Table 1-5 and Table 1-6 list the pins and definitions for the backpanel connector.

Table 1-5. Backpanel Connector P1

Pin Number	Slot 1 Signal	In/Out	Not Slot 1 Signal	In/Out	Comments
1, 2	GND	—	GND	—	
3, 4	+5 MAIN	I	+5 MAIN	I	
5, 6	NO CONNECT	—	NO CONNECT	—	(+12V MEMORY)
7, 8	+5 MEMORY	I	+5 MEMORY	I	MAIN AFTER SLOT 7
9, 10	NO CONNECT	—	NO CONNECT	—	(-5V MEMORY)
11	TLREAD	I/O	TLREAD	I/O	
12	GND	—	GND	—	
13	TLPRES-	I	TLPRES-	I	
14	TLIORES-	O	TLIORES-	I	470 OHM PULL-UP IN SLOT 1
15	GND	—	GND	—	
16	TLPFWP-	I	TLPFWP-	I	
17	GND	—	GND	—	
18	CRUBITOUT	O	OPEN(CRUBITOUT)*	—	
19	GND	—	GND	—	
20	TLTM-	I/O	TLTM-	I/O	
21	GND	—	GND	—	
22	STORECLK-	O	OPEN(STORECLK-)*	—	
23	MODSEL0-	O	OPEN*	—	
24	GND	—	GND	—	
25	TLGO-	I/O	TLGO-	I/O	
26	GND	—	GND	—	
27	TLDAT12-	I/O	TLDAT12-	I/O	
28	TLDAT13-	I/O	TLDAT13-	I/O	
29	120HZ	I	120HZ	I	
30	TLDAT14-	I/O	TLDAT14-	I/O	
31	TLDAT15-	I/O	TLDAT15-	I/O	
32	CRUBIT13	O	OPEN(CRUBIT13)*	—	
33	NO CONNECT	—	NO CONNECT	—	IAQ- ON /10
34	CRUBIT15	O	OPEN(CRUBIT15)*	—	
35	MODSEL1-	O	OPEN*	—	
36	CRUBIT12	O	OPEN(CRUBIT12)*	—	
37	MODSEL2-	O	OPEN*	—	
38	CRUBIT14	O	OPEN(CRUBIT14)*	—	
39, 40	NO CONNECT	—	NO CONNECT	—	(+12V MAIN)
41, 42	NO CONNECT	—	NO CONNECT	—	(-12V MAIN)
43	MODSEL3-	O	OPEN*	—	
44	MODSEL4-	O	OPEN*	—	
45	MODSEL5-	O	OPEN*	—	
46	MODSEL6-	O	OPEN*	—	
47	MODSEL7-	O	OPEN*	—	

Table 1-5. Backpanel Connector P1 (Continued)

Pin Number	Slot 1 Signal	In/Out	Not Slot 1 Signal	In/Out	Comments
48	MODSEL8–	O	OPEN(MODSELB–)*	—	
49	MODSEL9–	O	OPEN*	—	CBDAT(0)—
50	CRUBIT7	O	OPEN(CRUBIT7)*	—	
51	MODSEL10–	O	OPEN*	—	CBDAT(1)—
52	CRUBIT6	O	OPEN(CRUBIT6)*	—	
53	MODSEL11–	O	OPEN*	—	CBDAT(2)—
54	CRUBIT5	O	OPEN(CRUBIT5)*	—	
55	TLMER–	I/O	TLMER–	I/O	
56	CRUBIT4	O	OPEN(CRUBIT4)*	—	
57	GND	—	GND	—	
58	TLAV	I	TLAV	I	
59	GND	—	GND	—	
60	CRUBITIN	I	CRUBITIN	—	470 OHM PULL-UP IN SLOT 1
61	MODSEL12–	O	OPEN*	—	CBDAT(3)–
62	CRUBIT8	O	OPEN(CRUBIT8)*	—	
63	TLWAIT–	I	TLWAIT–	I	
64	CRUBIT9	O	OPEN(CRUBIT9)*	—	
65	NO CONNECT	—	NO CONNECT	—	
66	INTP1A–	O	INTP1A–	O	HOST INTERRUPT
67	MODSEL13–	O	OPEN*	—	CBDAT(4)–
68	CRUBIT10	O	OPEN(CRUBIT10)*	—	
69	MODSEL14–	O	OPEN*	—	CBDAT(5)–
70	CRUBIT11	O	OPEN(CRUBIT11)*	—	
71	TLAK–	I/O	TLAK–	I/O	
72	GND	—	GND	—	
73	NO CONNECT	—	NO CONNECT	—	TLCACHEN ON /12
74	CBTEST	—	CBTEST	—	GND IN CHASSIS
75	NO CONNECT	—	NO CONNECT	—	
76	MODSEL15–	O	OPEN*	—	
77, 78	+5 MAIN	I	+5 MAIN	I	
79 ,80	GND	—	GND	—	

Note:

* Outputs tristate, inputs disabled by slot 1 switch.

Table 1-6. Backpanel Connector P2

Pin Number	Slot 1 Signal	In/Out	Not Slot 1 Signal	In/Out	Comments
1, 2	GND	—	GND	—	
3, 4	+5 MAIN	I	+5 MAIN	I	
5	TLAG(OUT)	O	TLAG(OUT)	O	
6	TLAG(IN)	I	TLAG(IN)	I	
7	GND	—	GND	—	
8	TLADR14-	I/O	TLADR14-	I/O	
9	TLADR15-	I/O	TLADR15-	I/O	
10	TLADR10-	I/O	TLADR10-	I/O	
11	TLADR12-	I/O	TLADR12-	I/O	
12	TLADR11-	I/O	TLADR11-	I/O	
13	MODSEL23-	O	OPEN(TLPRES-)*	—	
14	NO CONNECT	—	N.C.(TLIORES-)	—	
15	TLADR13-	I/O	TLADR13-	I/O	
16	MODSEL22-	O	OPEN(TLPFWP-)*	—	
17	TLADR08-	I/O	TLADR08-	I/O	
18	MODSEL21-	O	OPEN(CRUBITOUT)*	—	
19	TLADR09-	I/O	TLADR09-	I/O	
20	TLDAT11-	I/O	TLDAT11-	I/O	
21	TLDAT08-	I/O	TLDAT08-	I/O	
22	MODSEL20-	O	OPEN(STORECLK-)*	—	
23	TLDAT10-	I/O	TLDAT10-	I/O	
24	INT3-	I	OPEN(GND)*	—	470 OHM PULL-UP
25	TLADR18-	I/O	TLADR18-	I/O	
26	TLHOLD-	I/O	TLHOLD-	I/O	
27	TLADR17-	I/O	TLADR17-	I/O	
28	RESTART-	I	RESTART-(OPEN)	—	
29	TLADR16-	I/O	TLADR16-	I/O	
30	GND	—	GND	—	
31	TLADR19-	I/O	TLADR19-	I/O	
32	MODSEL19-	O	OPEN(CRUBIT13)*	—	
33	TLDAT09-	I/O	TLDAT09-	I/O	
34	MODSEL18-	O	OPEN(CRUBIT15)*	—	
35	TLDAT02-	I/O	TLDAT02-	I/O	
36	MODSEL17-	O	OPEN(CRUBIT12)*	—	
37	TLDAT03-	I/O	TLDAT03-	I/O	
38	MODSEL16-	O	OPEN(CRUBIT14)*	—	
39, 40	+12 MAIN	I	+12 MAIN	I	
41, 42	-12 MAIN	I	-12 MAIN	I	
43	TLDAT06-	I/O	TLDAT06-	I/O	
44	TLADR01-	I/O	TLADR01-	I/O	
45	TLDAT07-	I/O	TLDAT07-	I/O	
46	INT4-	I/O	OPEN(MODSELB-)*	—	470 OHM PULL-UP
47	TLADR06-	I/O	TLADR06-	I/O	
48	INT5-	I	OPEN(MODSELA-)*	—	470 OHM PULL-UP
49	TLADR07-	I/O	TLADR07-	I/O	
50	INT6-	I	OPEN*	—	470 OHM PULL-UP
51	TLADR02-	I/O	TLADR02-	I/O	

Table 1-6. Backpanel Connector P2 (Continued)

Pin Number	Slot 1 Signal	In/Out	Not Slot 1 Signal	In/Out	Comments
52	INT7–	I	OPEN*	—	470 OHM PULL-UP
53	TLADR03–	I/O	TLADR03–	I/O	
54	INT8–	I	INT8–	I	470 OHM PULL-UP
55	TLADR00–	I/O	TLADR00–	I/O	
56	INT9–	I	INT9–	I	470 OHM PULL-UP
57	TLADR04–	I/O	TLADR04–	I/O	
58	INT10–	I	OPEN(GND)*	—	470 OHM PULL-UP
59	TLADR05–	I/O	TLADR05–	I/O	
60	NO CONNECT	—	N.C.(CRUBITIN)	—	
61	TLDAT04–	I/O	TLDAT04–	I/O	
62	INT11–	I	INT11–	I	470 OHM PULL-UP
63	TLDAT05–	I/O	TLDAT05–	I/O	
64	INT12–	I	INT12–	I	470 OHM PULL-UP
65	INT13–	I	INT13–	I	470 OHM PULL-UP
66	INT14–	I	INT14–	I	470 OHM PULL-UP
67	TLDAT00–	I/O	TLDAT00–	I/O	
68	INT15–	I	INT15–	I	470 OHM PULL-UP
69	TLDAT01–	I/O	TLDAT01–	I/O	
70	NO CONNECT	—	NO CONNECT	—	
71, 72	NO CONNECT	—	NO CONNECT	—	(–5V MEMORY)
73, 74	+5 MEMORY	I	+5 MEMORY	I	MAIN ABOVE SL7
75, 76	NO CONNECT	—	NO CONNECT	—	(+12V MEMORY)
77, 78	+5 MAIN	I	+5 MAIN	I	
79, 80	GND	—	GND	—	

Note:

* Outputs tristate, inputs disabled by slot 1 switch.

1.5.8 Front Panel Interface

The 990/10A processor board supports a front panel through a 20-pin top edge connector. This interface is compatible with standard 990 front panels. The front panel provides switches and indicators to permit operator control and observation of computer operation. The front panel interface is a CRU device, with 16 directly addressable input bits and 16 directly addressable output bits.

The front panel operates in either the RUN mode or the HALT mode. When operating in the RUN mode, all switches are disabled except for the HALT/SIE switch. The indicators can still be used to display data by issuing LDCR instructions. When a programmer panel is connected, and the front panel is in the HALT mode, pressing the HALT/SIE switch causes a vector to the front panel code in the ROM loader. This allows for manual data entry by the operator.

Table 1-7 lists the signals on the front panel connector.

Table 1-7. Front Panel Connector

Signature	Direction	Pin Number
CRUBITIN	In	7
CRUBITOUT	Out	15
STORECLK–	Out	2
CRUBIT12	Out	19
CRUBIT13	Out	20
MODSEL–	Out	14
CRUBIT14	Out	18
CRUBIT15	Out	16
IDLELED–	Out	11
RESTART–	In	5
RUN–	Out	12
FAULTLED–	Out	6
POWERLED–	Out	13
Ground	—	1
Ground	—	3
Ground	—	4
Ground	—	17
+5 v main	Out	9
+5 v main	Out	10

1.5.9 Asynchronous Communication Port

The asynchronous communication port uses a TMS 9902 asynchronous communication controller to provide an on-board interface for one EIA RS-232C or RS-423 compatible device. Table 1-8 lists the connector pins on the asynchronous communication port connector.

NOTE

While data carrier detect (CCITT 109.0 and EIA CF) is not available as a software input, a jumper on the 990/10A permits using data carrier detect instead of data set ready as an input to the TMS 9902. For some modems, changing this jumper from the normal position of data set ready to data carrier detect will give a better indication of when the modem is available.

Table 1-8. Asynchronous Communication Port Connector

Signature	CCITT	EIA	Pin Number	I/O	Description
XMTD	103.0	BA	1	O	Transmitted data
RCVD	104.0	BB	2	I	Received data
RTS	105.0	CA	3	O	Request to send
CTS	106.0	CB	4	I	Clear to send
DSR	107.0	CC	5	I	Data set ready
DCD	109.0	CF	6	I	Data carrier detect
No connect			7-11		
DTR	108.2	CD	12	O	Data terminal ready
RIND	125.0	CE	13	I	Ring indicator
No connect			14		
ALB			15	O	Analog loopback
GND	102.0	AB	16	—	Signal ground
No connect			17-18		

1.5.10 TILINE Interface

The 990/10A processor supports all existing TILINE memory and peripherals on the TILINE bus. In addition, TILINE-connected devices can access the memory on the processor board. In this respect the logic on the 990/10A performs the same function as a TILINE coupler; it gives priority to a TILINE master accessing the 990/10A's on-board memory at the same time the 990/10A microprocessor is attempting to access it.

1.5.10.1 TILINE Master. The 990/10A functions as a TILINE master when it addresses memory address locations that are not implemented on the processor board. These locations can be TILINE memory modules, TILINE peripheral control space locations on peripheral controllers, or memory locations contained on other boards. Accesses via TILINE are synchronized to the microprocessor clock. If the memory access is decoded to be a TILINE memory or TILINE peripheral device, the 990/10A relinquishes access to the TILINE at the end of the normal TILINE cycle. (In other words, the 990/10A does not do burst transfers on the TILINE.) The 990/10A requests access to the TILINE only when an access is not on-board. This dramatically cuts down TILINE activity. Overhead due to the 990/10A on a TILINE master cycle is typically 800 nanoseconds. If the access is to the TPCS or if the 990/10A is not in slot 1, additional overhead is incurred.

1.5.10.2 TILINE Slave. The 990/10A functions as a slave when its on-board memory or its memory error log is addressed by an external TILINE master. Any memory access in progress by the microprocessor is allowed to complete (maximum latency of three clock cycles); then, the external master has priority access to the internal memory bus. The 990/10A slave interface monitors the TILINE GO signal from the backpanel. If the backpanel address lies within the range of the 990/10A on-board memory, the address, data, and read/write lines are latched on the 990/10A. The slave controller then accesses the on-board memory or memory error log. If TILINE GO should cease during the cycle, TILINE terminate is not asserted to end the cycle. An access to the 990/10A's memory by an external TILINE master typically requires 1.4 microseconds before TILINE terminate is asserted.

1.5.10.3 TILINE Hold. If TILINE hold is asserted during a slave cycle, the TILINE port is given exclusive access to the internal bus for the duration of TILINE hold. TILINE hold is normally used as an interlock procedure to ensure that no other master changes the contents of an address between the read and write of the instruction asserting TILINE hold. The absolute value (ABS) instruction is the only 990/10A instruction that asserts TILINE hold. If the 990/10A executes an ABS instruction on an external memory address, TILINE hold is generated on the backpanel. If an ABS instruction is directed to an onboard address, TILINE hold is not generated. The slave controller still prevents an external master from accessing the internal bus on the 990/10A until the final write of the ABS instruction is complete.

1.5.10.4 TILINE Wait. TILINE wait is generated by TILINE couplers to resolve conflicts when two masters, each having access to the TILINE in their respective chassis, attempt to access slaves in the opposite chassis. The TILINE coupler resolves this deadlock by giving priority to the master designated by the cable vector and sending TILINE wait to force the lower priority master to relinquish access to the TILINE. TILINE wait causes the 990/10A master controller to relinquish control of the TILINE if it has access. If the master that caused the TILINE wait is addressing any of the onboard functions of the 990/10A, then the 990/10A slave controller grants that master access to the internal bus. This applies only to conflicts between masters in separate chassis joined by TILINE couplers. For multiple 990/10As in a single chassis, only one master has access to the TILINE and deadlocks over contention for the local memory bus are resolved by the addressed 990/10A slave controller.

2

Installation

2.1 GENERAL

This section provides information and procedures for unpacking the 990/10A from its shipping container, and installing it in a Model 990 System chassis. When the 990/10A is shipped in a chassis as part of a complete system, refer to the unpacking procedures for the chassis. All switches and jumpers described in later paragraphs are set for normal operation when the 990/10A is shipped as part of a system.

This section covers installation details of the 990/10A board only. Refer to the manual furnished with the computer system chassis for a discussion of interrupt and TILINE connections, and for logic board and interface cable installation procedures. Instructions are provided in this section for setting the option switches and jumpers on this board. The procedures assume that the user has a fundamental knowledge of basic hand tools and cabling techniques, but they do not require a detailed understanding of computer hardware or software.

2.2 UNPACKING/PACKING 990/10A BOARD

Upon receipt of the container, inspect it to ensure that no damage has occurred. After completion of the preliminary inspection, perform the following steps to remove the board from its container and ready the computer for operation.

NOTE

Do not discard any packing materials until unpacking, inspection, and inventory are complete.

1. Remove the top cushion pad.

2. Remove the envelope that contains shipping information and an inventory of equipment shipped.

3. Pack all shipping materials into the original shipping container and store the container for use in reshipment of the unit.

4. Inspect the 990/10A board and components for signs of damage that may have occurred during shipment. If damage has occurred, notify the carrier immediately.

To repack the unit, reverse the above procedure using the original packing material.

2.3 990/10A INSTALLATION PROCEDURES

The following paragraphs describe the preparation and installation of the 990/10A computer board.

WARNING

Ensure that the chassis ac power cord is disconnected from power while performing the installation procedures. Failure to observe this precaution could result in severe electrical shock.

2.3.1 Setting Option Switches and Jumpers on the 990/10A Board

Two dual in-line pack (DIP) switches and five jumpers are provided on the 990/10A board for option setting. Figure 2-1 shows the locations of these switches and jumpers.

The four position DIP switch controls the lower bound address of the on-board memory and is set simply by coding in the binary equivalent of the most significant hexadecimal digit of the desired beginning TILINE address (see Table 1-1). For a typical single processor system, all switches should be off.

The eight position switch contains the slot 1 switch in the leftmost position (SW1) and sets the address of the memory error log in the remaining seven positions (SW2 through SW8). The slot 1 switch disables certain slot 1 functions for operating the 990/10A as an auxiliary processor and should be off for normal slot 1 operation. Table 2-1 describes setting the memory control switches.

Table 2-1.   Memory Control TPCS Addresses

Logical Address (Hexadecimal)	Physical Address (Hexadecimal)	Sw2	Sw3	Sw4	Sw5	Sw6	Sw7	Sw8
F800-06	FFC00-03	Off	Off	Off	Off	Off	Off	Off
F808-0E	FFC04-07	Off	Off	Off	Off	Off	Off	On
FB00-06[1]	FFD80-83	On	On	Off	Off	Off	Off	Off
:	:	:	:	:	:	:	:	:
FBF0-F6	FFDF8-FB	On	On	On	On	On	On	Off
FBFC-FE[2]	FFDFC-FF	On	On	On	On	On	On	On

Notes:

[1] Suggested default.

[2] This setting causes self-test to fail with a front panel code of >0030.

Figure 2-1. Switch and Jumper Locations

FRONT PANEL CONNECTOR

EIA PORT CONNECTOR

J3

LOWER BOUND ADDRESS

J2

J5

J4

J1

OPTION SWITCHES

MEMORY ERROR LOG ADDRESS

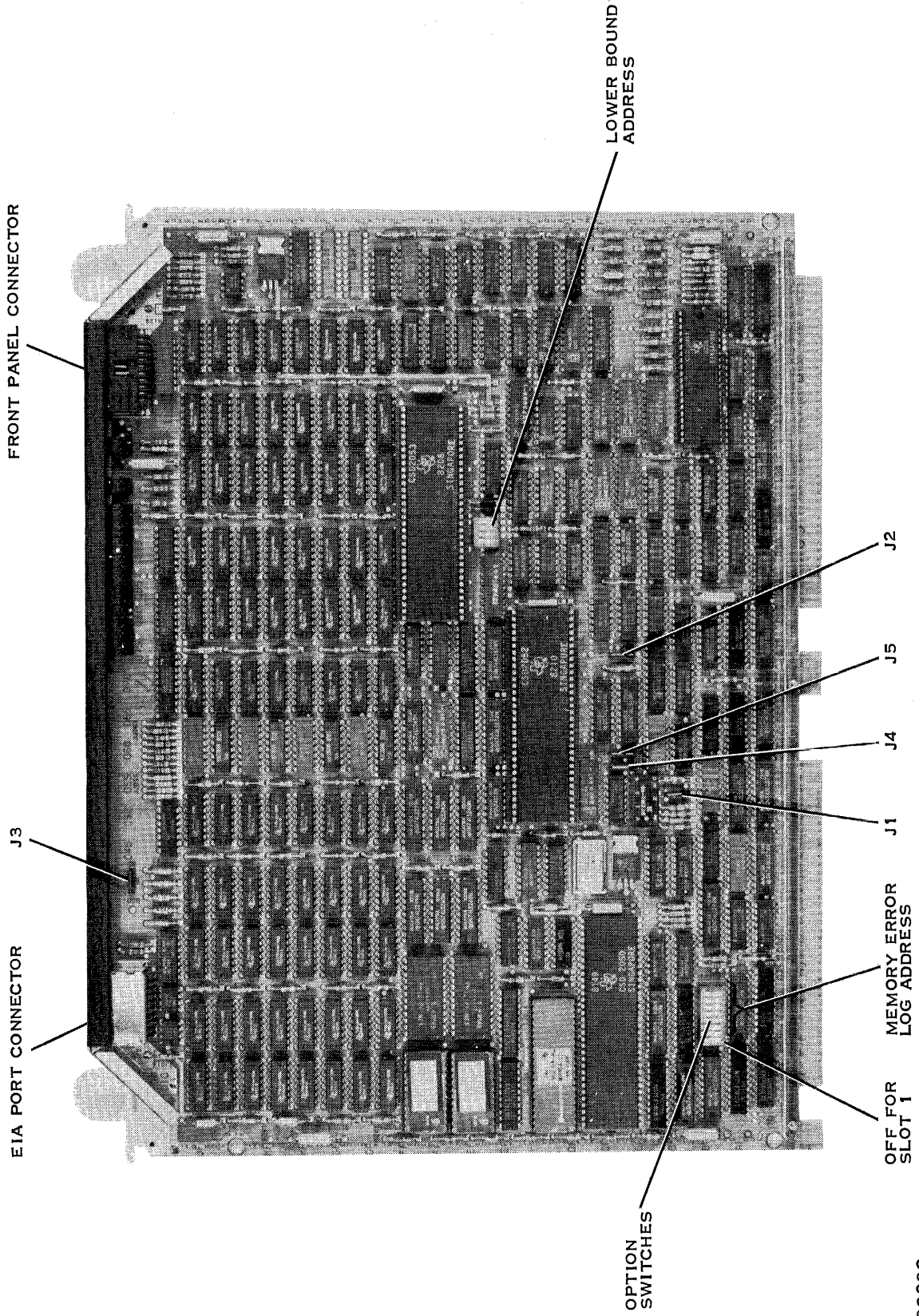OFF FOR SLOT 1

2283232

The five jumpers on the 990/10A are listed in Table 2-2. When installed, jumper J1 causes the EIA port interrupt to be "OR"ed with any backpanel interrupt connected to interrupt level 8. Jumper J2 is a two-position jumper that allows the user to select whether the real-time clock interrupts at interrupt level 5 or 15. For typical applications, the jumper should be installed in the level 5 position. If the jumper is removed, there will be no real-time clock interrupt. Jumper J3 allows the user to select whether data set ready or data carrier detect is used to determine the ready condition of the EIA device. Data set ready should be used for typical applications. Jumper J4 should be removed if the user desires the processor to perform a load on every power up. Jumper J5 should be removed to use the 990/10A as an auxiliary processor. Each of these functions is explained in more detail in Section 1 of this manual.

Table 2-2. Option Jumpers

Jumper	Pin 1 Location	Signature	Function
J1	BD053	PRT1INT–	EIA port interrupt
J2	CE074	RTCINT–	Real-time clock interrupt
J3	KF039	DSRA	Data set ready source
J4	CB057	PULOAD–	Load on power-Up
J5	CB059	HOST–	HOST/AUX jumper

3

Operation

3.1 GENERAL

This section describes operation of the 990/10A after installation in the chassis with appropriate peripherals and mass storage devices. This description assumes that the chassis is equipped with a control/display module (CDM) that provides a hexadecimal front panel display of error codes (Figure 3-1). An optional programmer panel is available to provide a means of manual entry of data into the computer or for modification of programs. Operation of the 990/10A with the programmer panel is described in the *990/10A Computer Maintenance Manual, Field Theory and Maintenance*.
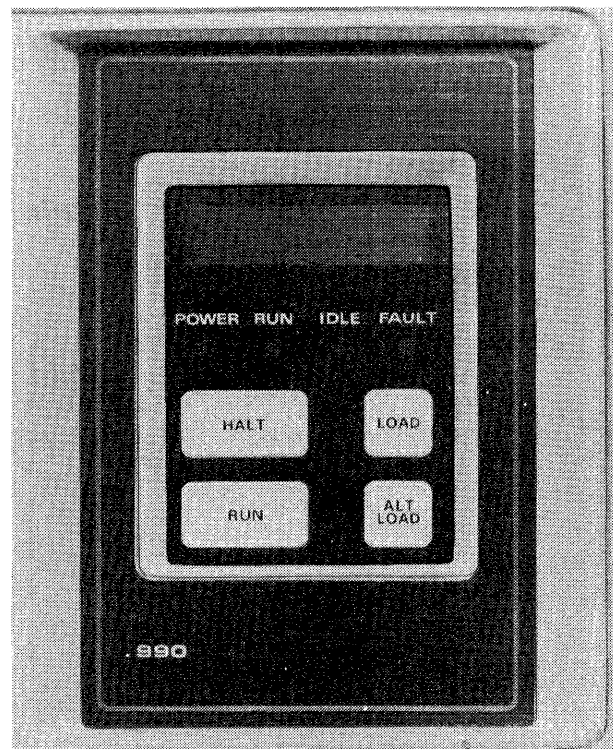
2283233

Figure 3-1. Control/Display Module

Some confusion results from the use of various terms for the 990 computer family front panels. Front panel is a general term that includes the following items.

- The control/display module described in this section

- The programmer panel described in this section

- The older operator panel that has a simple OFF-ON-LOAD switch and a minimal set of indicators

The 990/10A contains a self-test routine to verify the proper operation of the processor before loading the operating system. Self-test is normally executed on power-up as well as before loading. The time required for self-test execution ranges from one to forty seconds. In addition, the load device verification routine waits up to five minutes for the operator to provide a suitable load device. In case of error, self-test results are displayed on the front panel indicators.

3.2 CONTROL/DISPLAY MODULE CONTROLS AND INDICATORS

The control/display module provided with the computer chassis is fully explained in the *Model 990A13 Chassis Maintenance Manual, General Description*. This section provides a brief description of those controls and indicators necessary for normal operation with the 990/10A.

3.2.1 Controls
There are four switches of interest on the CDM. These switches are operational if the security switch is in the local position. The CDM is shipped with the switch in local; the remaining positions are for service only. The security switch is concealed by the chassis bezel when the CDM is installed.

3.2.1.1 HALT Switch. The HALT switch (Figure 3-1) suspends computer operation by forcing the processor to service a nonmaskable interrupt and execute the front panel service routine contained in the loader/self-test ROM. In this mode the HALT switch then signals a single instruction execution (SIE) which executes the interrupted program one instruction at a time. This feature is used and explained in the discussion on self-test failures.

3.2.1.2 RUN Switch. The RUN switch commands the processor to resume execution from the point at which it was halted.

3.2.1.3 LOAD Switch. The LOAD switch commands the processor to load from a TILINE device at address > F800 unit 0. Each load begins with an RSET instruction to reset all TILINE devices.

3.2.1.4   ALT LOAD Switch.   The ALT LOAD (alternate load) switch commands the processor to load from the TILINE device found by using the following universal search algorithm.

1.   Search for an online magnetic tape unit at > F880, units 0-3 respectively. If found, load from it.

2.   Search for an online disk unit at > F800, units 0-3 respectively, that is not write-protected. If found, load from it.

3.   Search for an online disk unit at > F810, units 0-3 respectively, that is not write-protected. If found, load from it.

4.   Search for an online disk unit at > F820, units 0-3 respectively, that is not write-protected. If found, load from it.

5.   If an online write-protected disk was found in steps 2-4, load from the first such device found. Otherwise, return to step 1 and reinitiate search.

Figure 3-2 shows a flowchart of the search algorithm.

3.2.2   Indicators
The CDM has four hexadecimal digits for displaying program data and four LEDs for displaying system state. These indicators are used jointly to display error codes.
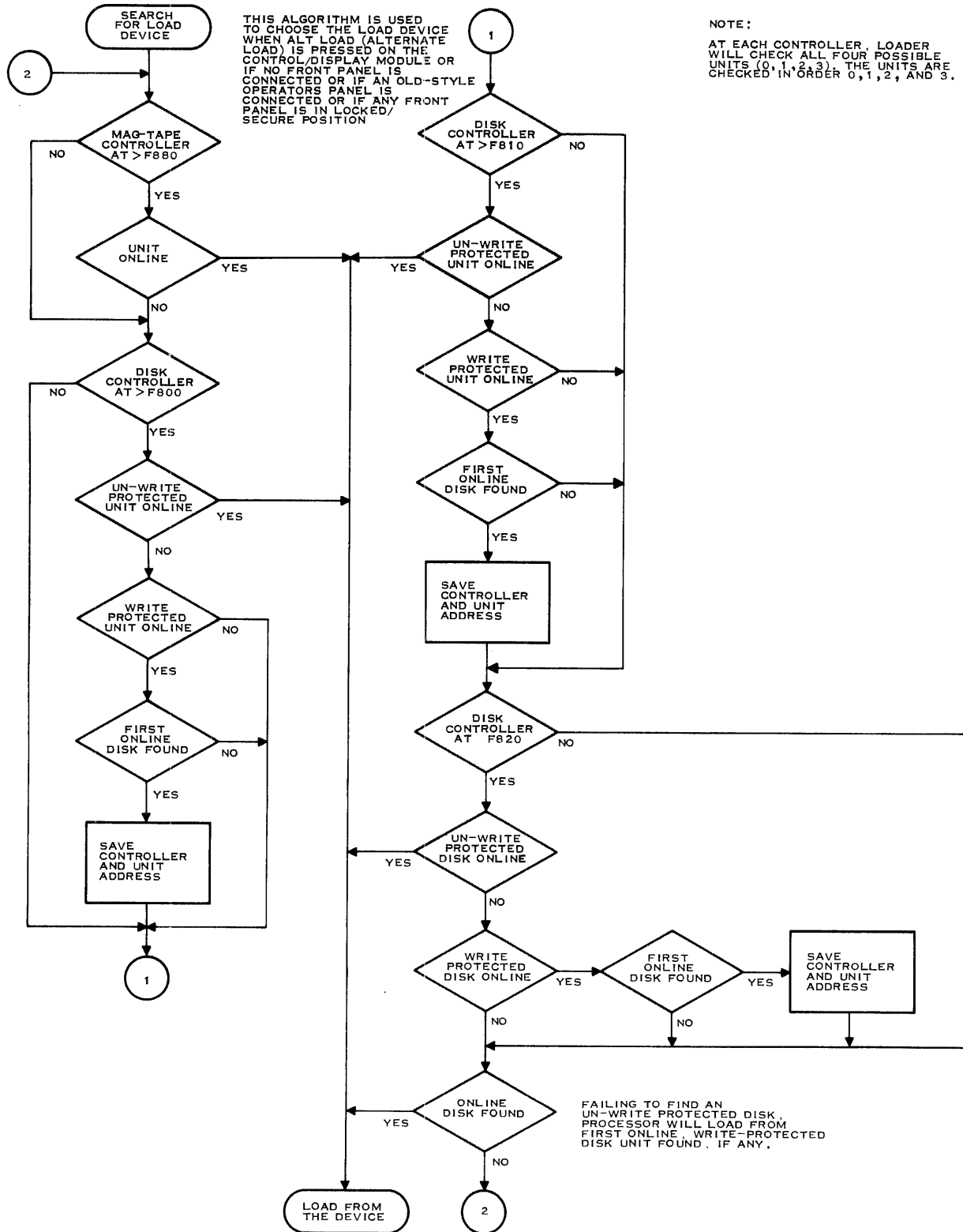
3.2.2.1   Hexadecimal Display.   The four hexadecimal digits are loaded via two byte CRU writes to address > 1FE0. They are under program control when the CPU is in the run mode. When the CPU is halted via the HALT switch, the program counter for the next instruction is displayed. Self-test, loader, or operating system error codes are displayed as explained in the following sections.

3.2.2.2   POWER LED.   The green POWER LED indicator is lighted whenever ac power is supplied to the computer chassis.

3.2.2.3   FAULT LED.   The red FAULT LED indicator is controlled by an addressable latch at CRU address > 1FF6. It is illuminated under program control to indicate self-test, loader, or operating system errors.

3.2.2.4   RUN LED.   The green RUN LED indicator is controlled by the loader/self-test code after power-up. When the HALT switch is pressed, processor operation is halted and the RUN LED is extinguished. It is illuminated again when the RUN switch restores processor operation.

3.2.2.5   IDLE LED.   The green IDLE LED indicator is illuminated whenever the processor executes the IDLE instruction. This is generally used to give an indication of processor activity or to distinguish among types of error codes.

SEARCH FOR LOAD DEVICE

2

THIS ALGORITHM IS USED TO CHOOSE THE LOAD DEVICE WHEN ALT LOAD (ALTERNATE LOAD) IS PRESSED ON THE CONTROL/DISPLAY MODULE OR IF NO FRONT PANEL IS CONNECTED OR IF AN OLD-STYLE OPERATORS PANEL IS CONNECTED OR IF ANY FRONT PANEL IS IN LOCKED/ SECURE POSITION

1

NOTE:

AT EACH CONTROLLER, LOADER WILL CHECK ALL FOUR POSSIBLE UNITS (0,1,2,3). THE UNITS ARE CHECKED IN ORDER 0,1,2, AND 3.

MAG-TAPE CONTROLLER AT >F880

NO

YES

DISK CONTROLLER AT >F810

NO

YES

UNIT ONLINE

YES

YES

NO

UN-WRITE PROTECTED UNIT ONLINE

NO

DISK CONTROLLER AT >F800

NO

YES

WRITE PROTECTED UNIT ONLINE

NO

UN-WRITE PROTECTED UNIT ONLINE

YES

YES

FIRST ONLINE DISK FOUND

NO

NO

WRITE PROTECTED UNIT ONLINE

NO

YES

SAVE CONTROLLER AND UNIT ADDRESS

FIRST ONLINE DISK FOUND

NO

YES

SAVE CONTROLLER AND UNIT ADDRESS

DISK CONTROLLER AT F820

NO

YES

1

UN-WRITE PROTECTED DISK ONLINE

YES

NO

WRITE PROTECTED DISK ONLINE

YES

FIRST ONLINE DISK FOUND

YES

SAVE CONTROLLER AND UNIT ADDRESS

NO

NO

ONLINE DISK FOUND

YES

NO

FAILING TO FIND AN UN-WRITE PROTECTED DISK, PROCESSOR WILL LOAD FROM FIRST ONLINE, WRITE-PROTECTED DISK UNIT FOUND, IF ANY.

LOAD FROM THE DEVICE

2

2282966

Figure 3-2.   Universal Loader Search Algorithm

3.3 990/10A PWB INDICATORS

On the top edge of the 990/10A printed wiring board (PWB), there are additional LED indicators that can be seen by looking into the chassis interior. Figure 3-3 shows an edge view of the 990/10A board and the LED indicators. These indicators are explained in the following paragraphs.

3.3.1 Major Fault LED

The MAJFAULT LED indicator is illuminated on power-up and extinguished after the kernel test portion of self-test shows that no major fault exists within the microprocessor-ROM core of the board. Unless the major fault logic itself fails, MAJFAULT should always be illuminated when power is initially applied to the board. If this LED fails to extinguish, the processor may have a failure that prohibits displaying other error indicators.

3.3.2 FAULT LED

The FAULT LED indicator is a duplicate of the FAULT indication on the CDM. It is software-controlled to show a self-test, loader, or operating system fault.

3.3.3 Memory Error Indicators

The remaining LEDs on the 990/10A PWB provide the service person with indications of a failed memory component. This information is also available to software through the memory error log and will not generally require the user's attention.

3.3.3.1 Double-Bit Error. The MER LED indicator designates the occurrence of a double-bit memory error since the last reset to the memory error log. If the MER LED is illuminated, all other memory error indicators except the row-in-error information are unreliable.

3.3.3.2 Single-Bit Error. The CER LED indicator designates the occurrence of a single-bit memory error (correctable error) since the last reset to the memory error log. If the CER LED is illuminated, but not the MER LED, then the row-in-error and bit-in-error LEDs pinpoint the memory integrated circuit (IC) causing the error.
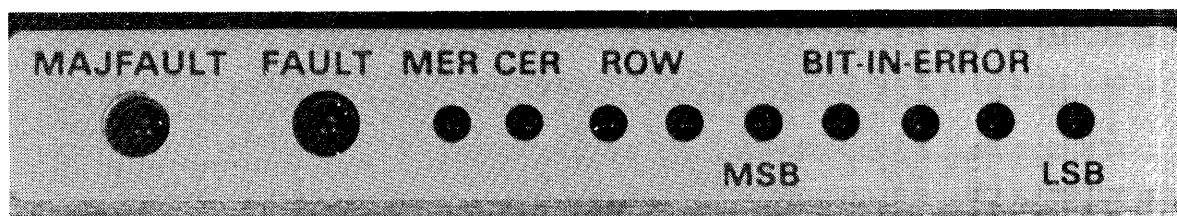
2283234

Figure 3-3.   990/10A Board Failure Indicators

3.3.3.3 ROW. The two ROW LEDs display in binary the logical row of memory chips that contains a single- or double-bit error. The leftmost LED is the most significant binary bit. Decoding and repair of this malfunction should be done only by trained service personnel using properly qualified parts.

3.3.3.4 BIT-IN-ERROR. The five BIT-IN-ERROR LEDs encode in binary the memory IC within the row designated by the ROW LEDs that caused a single-bit memory error.

3.4 990/10A SELF-TEST

The 990/10A has ROM-resident self-test code that exercises the logic on the 990/10A board and indicates the result of the test via the FAULT (or MAJFAULT) LEDs. Self-test, in general, is executed after the 990/10A is powered up (before any other process is performed) and before any load. Two parts of self-test, RAM test and load device verification, are not run every time self-test is executed.

RAM test is run only if it is determined that memory has not been initialized (as is the case on power-up with no standby power) or if an alternate load is being performed.

Load device verification is run as a part of self-test before any load from a TILINE device. Thus, on power-up when a load is not being performed, and on loads from CRU devices, load device verification is not run.

3.4.1 Self-Test Description
The first section of self-test deals with the major fault condition and the remaining sections deal with the logic outside of the major fault data path. A flowchart of the 990/10A self-test code is shown in Figure 3-4.

3.4.1.1 Major Fault Test. The first section of the self-test exercises the microprocessor data path. The microprocessor, the address latch, the loader/self-test ROMs, the scratch pad RAM, the CRU chip, the ready controller, the ready logic, and the oscillator constitute the microprocessor data path. A failure implies that the problem is likely to be a faulty microprocessor chip or loader/self-test ROM. Some of the tests performed in this section are:

- Execute a CRC of the self-test ROMs.

- Test the scatch-pad RAM.

- Execute a representative sample of instructions through the microprocessor and test the results.

If the major fault section of self-test is passed successfully, the major fault LED is extinguished and the remainder of self-test is initiated. If the test is not successful, one (MAJFAULT) or both fault LEDs on the processor board remain illuminated and the processor is inoperative. In case of a MAJFAULT indication, all accesses to on-board or off-board addresses are inhibited.
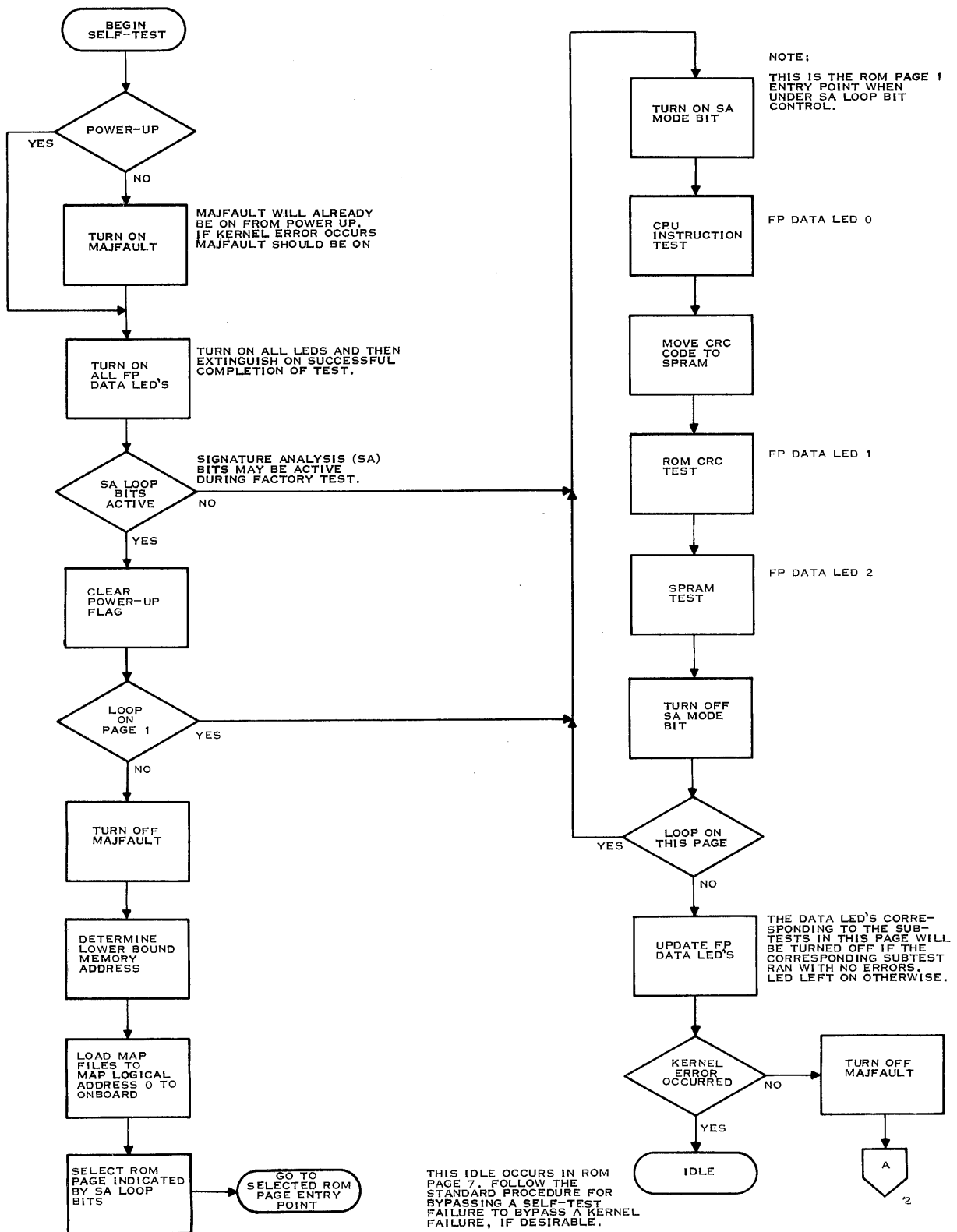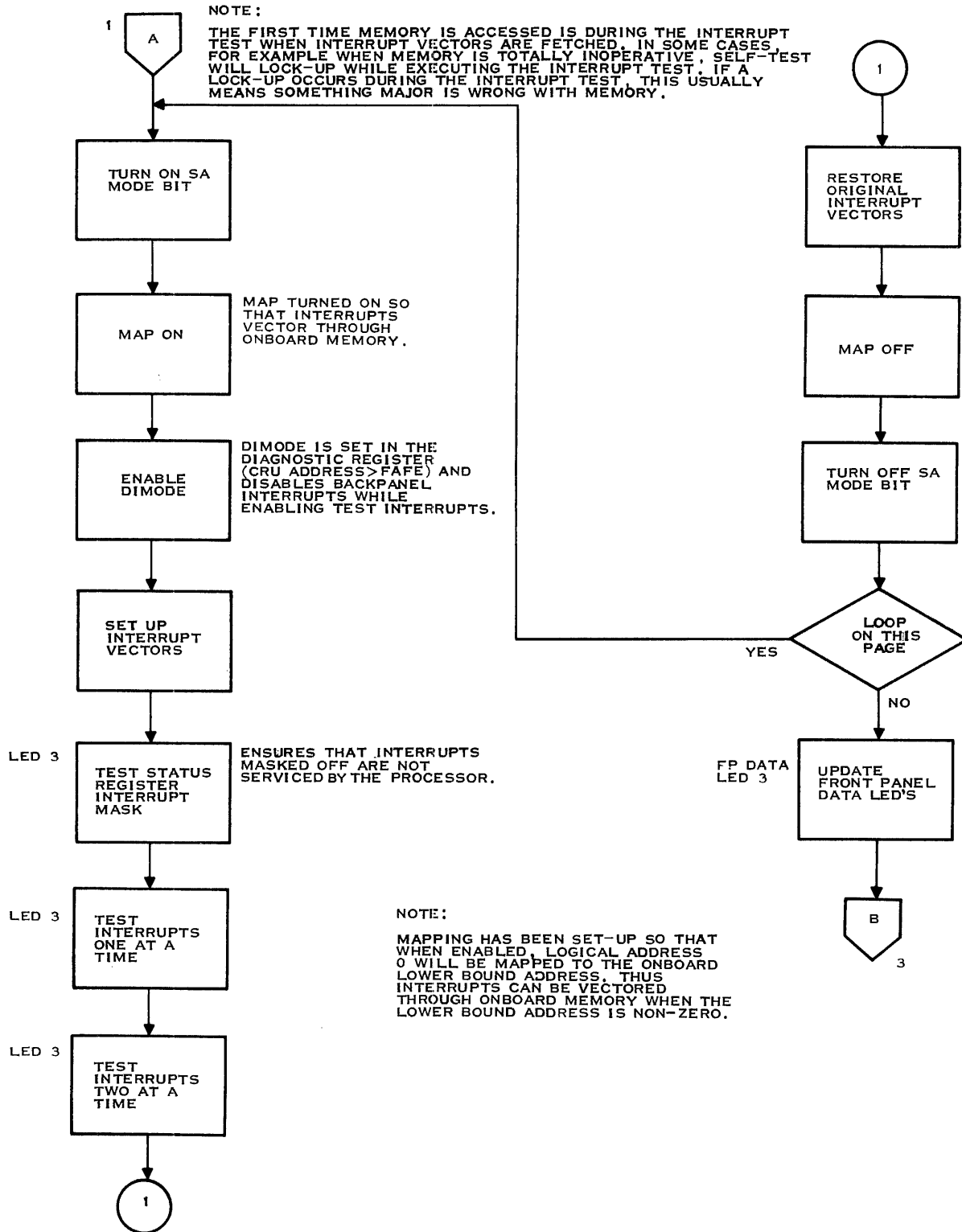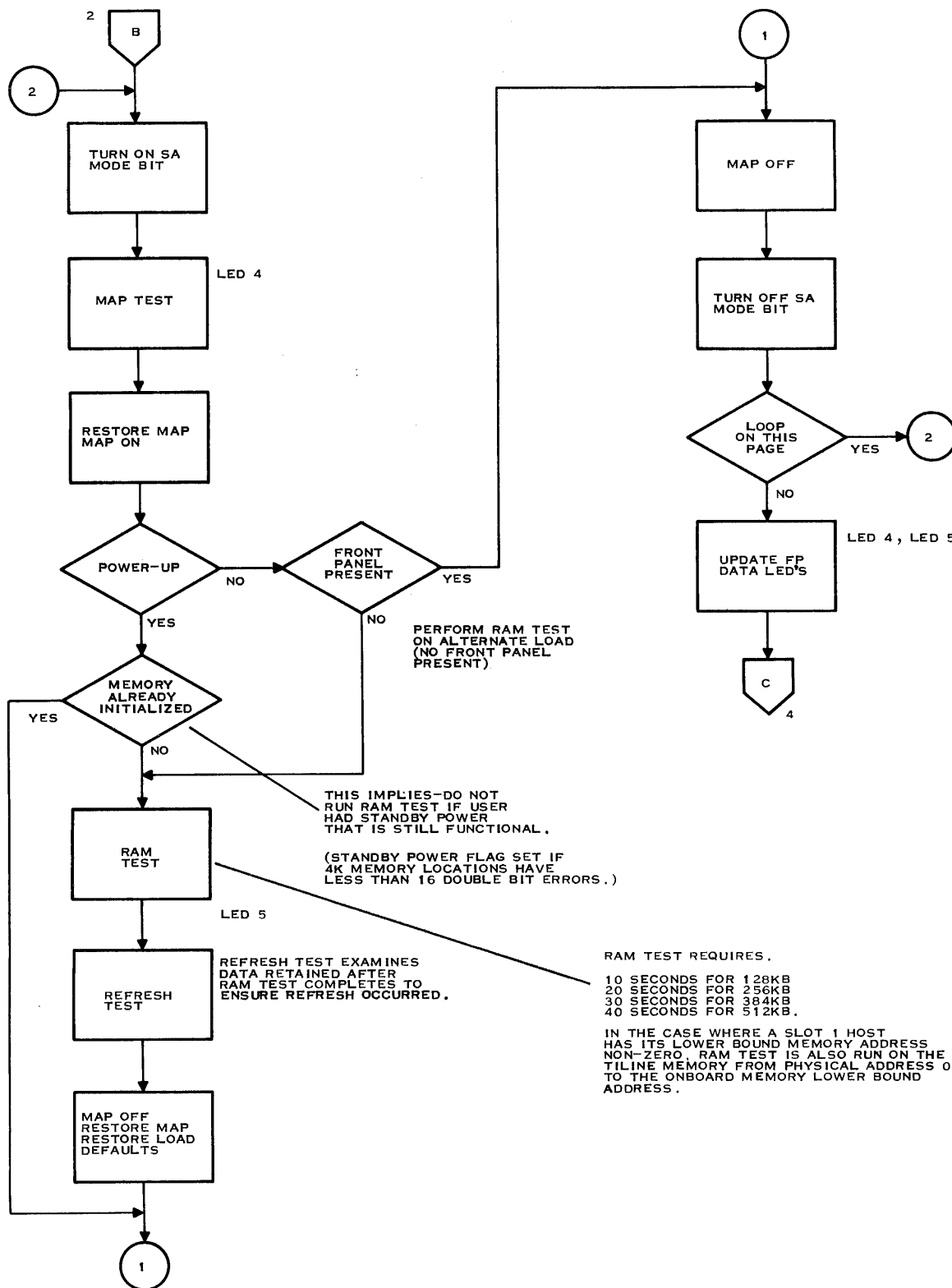
BEGIN SELF-TEST

POWER-UP

YES

NO

TURN ON MAJFAULT

MAJFAULT WILL ALREADY BE ON FROM POWER UP. IF KERNEL ERROR OCCURS MAJFAULT SHOULD BE ON

TURN ON ALL FP DATA LED'S

TURN ON ALL LEDS AND THEN EXTINGUISH ON SUCCESSFUL COMPLETION OF TEST.

SA LOOP BITS ACTIVE

SIGNATURE ANALYSIS (SA) BITS MAY BE ACTIVE DURING FACTORY TEST.

NO

YES

CLEAR POWER-UP FLAG

LOOP ON PAGE 1

YES

NO

TURN OFF MAJFAULT

DETERMINE LOWER BOUND MEMORY ADDRESS

LOAD MAP FILES TO MAP LOGICAL ADDRESS 0 TO ONBOARD

SELECT ROM PAGE INDICATED BY SA LOOP BITS

GO TO SELECTED ROM PAGE ENTRY POINT

THIS IDLE OCCURS IN ROM PAGE 7. FOLLOW THE STANDARD PROCEDURE FOR BYPASSING A SELF-TEST FAILURE TO BYPASS A KERNEL FAILURE, IF DESIRABLE.

NOTE:
THIS IS THE ROM PAGE 1 ENTRY POINT WHEN UNDER SA LOOP BIT CONTROL.

TURN ON SA MODE BIT

CPU INSTRUCTION TEST

FP DATA LED 0

MOVE CRC CODE TO SPRAM

ROM CRC TEST

FP DATA LED 1

SPRAM TEST

FP DATA LED 2

TURN OFF SA MODE BIT

LOOP ON THIS PAGE

YES

NO

UPDATE FP DATA LED'S

THE DATA LED'S CORRE-SPONDING TO THE SUB-TESTS IN THIS PAGE WILL BE TURNED OFF IF THE CORRESPONDING SUBTEST RAN WITH NO ERRORS. LED LEFT ON OTHERWISE.

KERNEL ERROR OCCURRED

NO

TURN OFF MAJFAULT

YES

IDLE

A

2

2282965 (1/6)
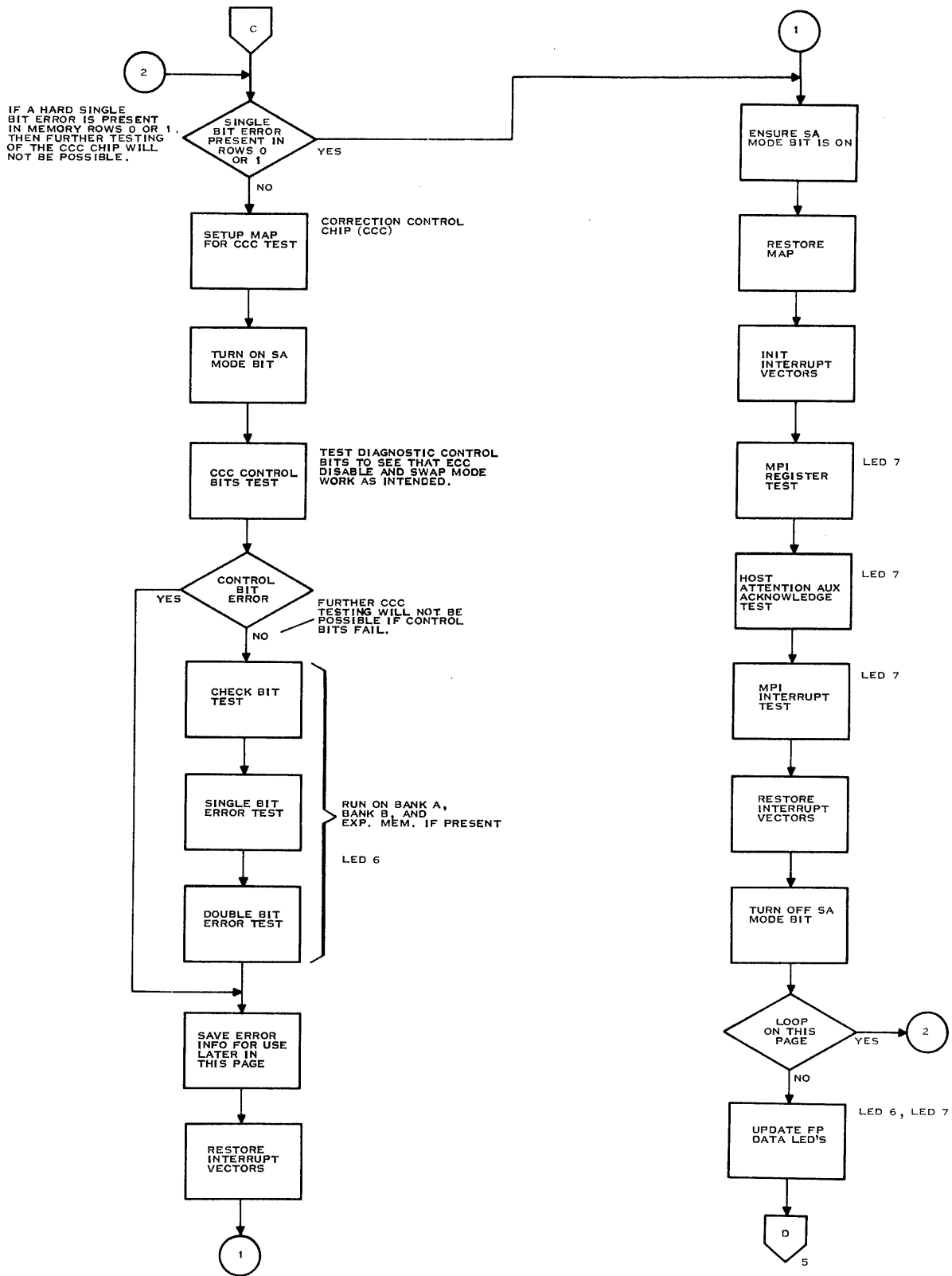
Figure 3-4.   Flowchart of 990/10A Self-Test Code (Sheet 1 of 6)

NOTE:

THE FIRST TIME MEMORY IS ACCESSED IS DURING THE INTERRUPT TEST WHEN INTERRUPT VECTORS ARE FETCHED. IN SOME CASES, FOR EXAMPLE WHEN MEMORY IS TOTALLY INOPERATIVE, SELF-TEST WILL LOCK-UP WHILE EXECUTING THE INTERRUPT TEST. IF A LOCK-UP OCCURS DURING THE INTERRUPT TEST, THIS USUALLY MEANS SOMETHING MAJOR IS WRONG WITH MEMORY.

A ... 1

TURN ON SA MODE BIT

MAP ON — MAP TURNED ON SO THAT INTERRUPTS VECTOR THROUGH ONBOARD MEMORY.

ENABLE DIMODE — DIMODE IS SET IN THE DIAGNOSTIC REGISTER (CRU ADDRESS>FAFE) AND DISABLES BACKPANEL INTERRUPTS WHILE ENABLING TEST INTERRUPTS.

SET UP INTERRUPT VECTORS

LED 3 — **TEST STATUS REGISTER INTERRUPT MASK** — ENSURES THAT INTERRUPTS MASKED OFF ARE NOT SERVICED BY THE PROCESSOR.

LED 3 — **TEST INTERRUPTS ONE AT A TIME**

LED 3 — **TEST INTERRUPTS TWO AT A TIME**

(1)

1

RESTORE ORIGINAL INTERRUPT VECTORS

MAP OFF

TURN OFF SA MODE BIT

LOOP ON THIS PAGE — YES / NO

FP DATA LED 3 — **UPDATE FRONT PANEL DATA LED'S**

B

3

NOTE:

MAPPING HAS BEEN SET-UP SO THAT WHEN ENABLED, LOGICAL ADDRESS 0 WILL BE MAPPED TO THE ONBOARD LOWER BOUND ADDRESS. THUS INTERRUPTS CAN BE VECTORED THROUGH ONBOARD MEMORY WHEN THE LOWER BOUND ADDRESS IS NON-ZERO.

2282965 (2/6)

Figure 3-4.   Flowchart of 990/10A Self-Test Code (Sheet 2 of 6)

Figure 3-4. Flowchart of 990/10A Self-Test Code (Sheet 3 of 6)

2282965 (3/6)

Figure 3-4. Flowchart of 990/10A Self-Test Code (Sheet 4 of 6)

Figure 3-4. Flowchart of 990/10A Self-Test Code (Sheet 5 of 6)

Figure 3-4.   Flowchart of 990/10A Self-Test Code (Sheet 6 of 6)

3.4.1.2   General Fault Test.   The remaining sections of the self-test exercise major portions of the 990/10A. An overview of the logical sections and their associated tests follows:

- Map chip, address buffer, and address latch check.

    — Latch and read back mapped addresses to verify all internal registers.

    — Verify that TPCS address generation is correct.

- RAM test.

    — Write and read from/to RAM memory with ECC enabled.

    — Write and read from/to RAM memory with ECC disabled. Reenable ECC and verify that single-bit errors are corrected and double-bit errors detected.

    — Read the error log to verify that errors were identified.

    — Verify that 1 and 0 can be written to every bit position in every word in the on-board memory address space. Verify address uniqueness in the on-board address space. This test is not run if battery backup is determined to be present and operational.

- TILINE test.

    — Initiate loopback test mode to verify operation of external address decode and TILINE slave controller.

    — Initiate TILINE master cycle to TILINE time-out address (FBFE) and verify that master timed out.

- CRU and interrupt check.

    — Check interrupts individually and in combinations.

    — Check CKON, CKOF, LREX, and IDLE external instructions.

    — Set error status latch, verify interrupt, and clear status individually.

    — Check front panel logic contained on the central processor.

- Nonload path logic test.

    — Perform test of multiprocessor logic.

    — Perform EIA port loopback test.

- Perform load device verification test (runs only before load is attempted).

  — Verify that a load device is present.

  — Verify that the load device controller is operational.

  — Verify that the load device unit is operational.

3.4.2 Self-Test Execution Time

Self-test, excluding RAM test, executes in approximately one second. RAM test requires 20 seconds for 256K bytes and 40 seconds for 512K bytes to execute. While RAM test is executing, the front panel LED's alternately display > 0F0F and > F0F0.

3.4.3 Self-Test Error Reporting

When self-test execution begins, all error LED's (FAULT and MAJFAULT) are turned on and all F's are displayed on the front panel. As each section of self-test completes, the bit position corresponding to the completed section is changed to 0 if the section completed with no errors. Refer to Figure 3-5 and Table 3-1 for the self-test section to bit position correspondence definition. The only exception to this is while RAM test is executing, the front panel LED's alternately display > 0F0F and > F0F0. After completion of RAM test, the front panel display resumes its error information display.

The error LED's are turned off as follows. Major fault (MAJFAULT) is turned off if the first three sections of self-test complete successfully. Should a failure occur in any of these three sections, the 990/10A processor will go to IDLE (if possible) since something major is wrong and further testing is probably not possible. This self-test failure can be bypassed like a self-test fault to force self-test to continue testing. See paragraph 3.4.4 for instructions on how to bypass a self-test failure. If all sections of the general fault test are passed successfully, the FAULT LED will be extinguished and the RUN LED will be illuminated. If the general fault test is unsuccessful, the FAULT LED will be illuminated and the CPU will idle with the IDLE LED on. At this time, the hexadecimal display will display an error code indicating a failure in one or more modules of self-test as defined in Table 3-1. If it is then determined that it is desirable to bypass the failure, refer to paragraph 3.4.4 for instructions on how to bypass a self-test failure.
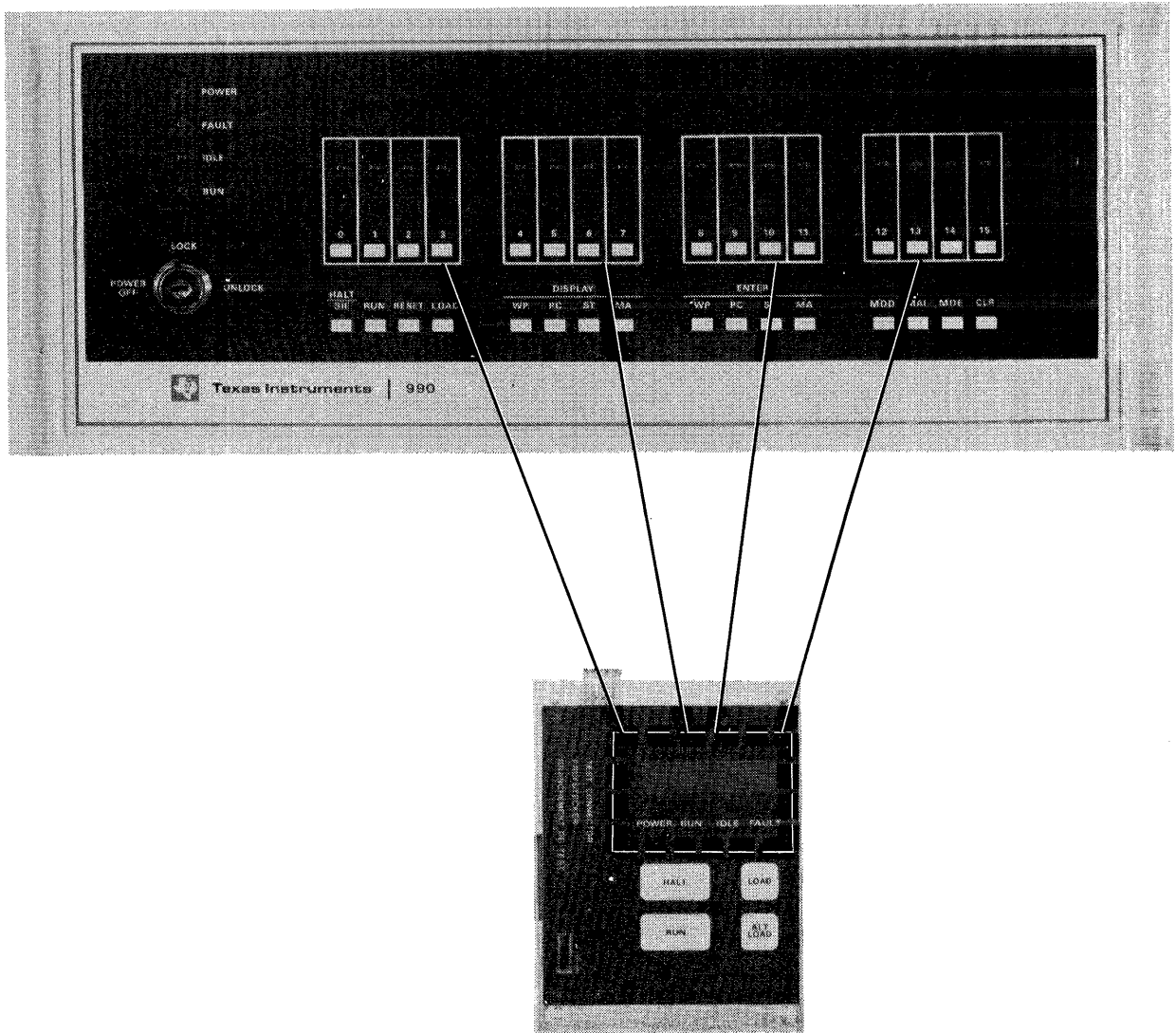
3.4.4 Bypassing a Self-Test Failure

When a self-test failure occurs, it is possible to bypass the failure and force the processor to continue.

To bypass a self-test failure:

1. Press HALT.

2. Press HALT 5 times.

3. Press RUN.

CAUTION

Do not attempt to execute the loader following a self-test failure without repairing the CPU unless you have a backup copy of the load medium.

2283235

Figure 3-5.   Programmer Panel and CDM

In the rare case where a major fault condition is bypassed, the processor will continue with the rest of self-test. If the preload self-test failed, the processor will continue with the load (if possible). If self-test is by-passed on power-up, the processor will continue as if self-test had passed successfully (run front panel code, load, or go to IDLE depending on the configuration).

3.4.5 Other Information Available as a Result of a Self-Test Failure
Should self-test fail load device verification, the status of the failing TILINE controller can be displayed on the front panel. Perform the following procedures to display the controller and unit status registers (R0 and R7 respectively):

To display the controller status register (R0):

1. Press HALT.

2. Press HALT 3 times.

3. Press RUN.

The front panel display now shows the contents of the controller status register.

To display the unit status register (R7):

1. Press HALT.

2. Press HALT 4 times.

3. Press RUN.

The front panel display now shows the contents of the unit status register.

If a single-bit memory error occurs during self-test, the memory correction circuits correct the error, and self-test does not fail. The memory error indicator LEDs are cleared before self-test completes, so no indication of a single-bit error remains. If the memory error indicators are observed during self-test, they remain illuminated long enough to be noted if a single-bit error occurs. If self-test fails and a memory error is suspected, a programmer panel can be used to read memory location > FADE for the value of the memory error log, word one, stored during self-test.

3.5 LOAD DEVICE VERIFICATION

Load device verification is the part of self-test that is run only before loads and checks to see that the 990/10A processor is able to communicate with the TILINE device that will be used for loading. Thus, if the 990/10A processor is unable to successfully complete the load device verification tests, there is a very high probability that the load will also fail due to hardware malfunction (or operator error).

Load device verification is run only on TILINE devices. On any other type of device, the load device verification tests are skipped. The four tests performed during load device verification are:

- Controller Present Test — Verifies that the TPCS controller registers can be read (no TILINE timeout occurs on a read from the controller address).

- Unit Online Test — By reading the TPCS controller registers, verifies that there is a unit online for that controller.

- Controller Operational Test — Issues a simple command to the controller and verifies that it responds as expected.

- Data Transfer Test — Transfers data, as the loader will, from the load device to memory.

The four least significant binary bits (rightmost hexadecimal digit on the 990 CDM) are used to report the results of the four tests just mentioned. Once a test fails, the remaining tests are not run. Thus, information is quickly given to the user as to why a load failed.

3.5.1 Special Features of Load Device Verification

Load device verification test 2 (Unit Online Test) will wait for a unit to come online if no unit is online when a load is attempted. This process of waiting is indicated by a 7 as the least significant hexadecimal digit on the front panel (a hexadecimal F for an alternate load). This test will wait for a maximum of five minutes, then timeout, and report a failure by going to IDLE.

Pressing the halt switch on the front panel at any time during load device verification will cause load device verification testing to terminate immediately and report a failure. To continue and ignore the failure, follow the instructions in paragraph 3.4.4.

3.5.2 Understanding Self-Test Error Codes

The self-test error codes are most easily explained by noting that the self-test code was written to provide maximum visibility for Field Service personnel using the 990 programmer panel. Figure 3-5 shows this programmer panel and the control display module in juxtaposition. Each hexadecimal display on the CDM corresponds to four binary LEDs on the programmer panel. At the beginning of self-test all the programmer panel LEDs are turned on (hexadecimal code > FFFF on the CDM). Each front panel bit corresponds to a particular section of testing. After each section of self-test completes, the corresponding front panel LED is turned off if there is no error. A successful self-test execution is displayed on the CDM as >0000 with the POWER and RUN LEDs illuminated and the FAULT and IDLE LEDs off. A failed self-test has the hexadecimally encoded display of the values of self-test errors that were not turned off by self-test. The POWER, FAULT, and IDLE LEDs will be illuminated. If more than one self-test error occurs, the CDM displays the sum of the self-test error values. For example, if both the CRU chip and the EIA port test failed, the CDM would display >00C0.

3.5.3 Understanding Loader Error Codes

It is possible (though unlikely) for the 990/10A to pass the load device verification portion of self-test and still encounter an error when performing the load function. Disk and tape loader error codes are explained in the *ROM Loader User's Guide*. If a loader error occurs, the POWER and RUN LEDs are illuminated and the FAULT LED flashes. The flashing FAULT LED indicates a loader error as opposed to a self-test error. A failure in the ROM loader code (explained in the *ROM Loader User's Guide*) can be distinguished from a system loader failure (explained in the appropriate operating system manual) by the two leftmost hexadecimal digits flashing >FF for a system loader error.

Table 3-1. Self-Test Error Reporting Bit Assignment

Front Panel LED Lit	Self-Test Section Result
0	Microprocessor instruction set test failed
1	Loader/self-test ROM CRC test failed
2	Scratch pad RAM test failed
3	Interrupt logic test failed
4	Map chip test failed
5	Memory test failed
6	ECC logic test failed
7	Multiprocessing logic test failed
8	CRU chip test failed
9	EIA port test failed
10	TILINE timeout test failed
11	Write inhibit controller test failed
12	Could not find load device controller
13	Load device not online and ready
14	Load device did not execute restore command
15	Could not load data from load device into memory

4

Programming

4.1 GENERAL

This section describes selected aspects of programming the 990/10A.

4.2 INSTRUCTION SET

The instruction set of the 990/10A is a superset of the 990/10 instruction set and a subset of the 990/12 instruction set. *The Model 990 Computer 990/10 and 990/12 Assembly Language Reference Manual* applies to the 990/10A although it is not specifically addressed. Exceptions to that manual are explained in this section.

4.2.1 Additional Instructions

The 990/10A implements five additional instructions in addition to the 990/10 instruction set described in the *Assembly Language Reference Manual.* These instructions are:

    BIND   — Branch indirect
    DiVS   — Divide signed
    MPYS   — Multiply signed
    LST   — Load status
    LWP   — Load workspace pointer.

These instructions execute as defined for the 990/12.

4.2.2 Mapping Instructions (LMF, LDS, LDD)

The TMS 99000 microprocessor handles the mapping instructions LMF, LDS, and LDD as macroinstructions. All undefined opcodes vector into the macrostore ROM where they are examined for LMF, LDS, or LDD. If the instruction is not one of these three, it is an illegal opcode, and the microprocessor returns from macrostore with an illegal instruction interrupt pending.

If the opcode was LMF, LDS, or LDD, short assembly language routines are executed to load the map file registers with user specified data. The entry and exit overhead associated with the macrostore ROM causes these instructions to execute more slowly with a 990/10A than with a 990/10. Also, a BLWP instruction preceded by an LDS instruction has its source operand mapped via map file two on a 990/10A but not on a 990/10.

4.3 STATUS REGISTER

The status register of the 990/10A is shown in Figure 4-1. Bits 0 through 8 are identical on the 990/10 (with mapping), 990/12, and 990/10A. Bit 9 (memory management and protection enable on the 990/12) is not implemented on the 990/10 or 990/10A. Bit 10 (overflow interrupt enable) was not implemented on the 990/10 but is on the 990/10A and 990/12. This bit performs on the 990/10A as it does on the 990/12 except that following an overflow interrupt on divide operations, the 990/12 does not modify status bits 0-2 while the 990/10A does modify these status bits. This is not judged significant as status bits 0-2 are of little value after an overflow occurs. Bit 11 is not implemented on the 990/10. It is writable control store enable on the 990/12. On the 990/10A, this bit enables the macrostore ROM on the microprocessor during execution of the XOP instruction. Such action is undesirable as no routines exist in macrostore for handling XOPs and they will be treated as illegal instructions. Bits 12-15 of the status register contain the interrupt mask on all 990 computers. More detailed definitions of each bit are given in the *Assembly Language Programmer's Guide* (except for the aberrations of the 990/10A as noted above).

4.4 ERROR STATUS REGISTER

The error status register on the 990/10A is explained in paragraph 1.5.5.3 Note that it is identical to the 990/10 except for the additional arithmetic overflow indication at bit 4 and the different processor ID code at bits 0-3. This ID code permits distinguishing the 990/10A and its features from the other 990 family processors in software. The error status register is located at CRU base address >1FC0.
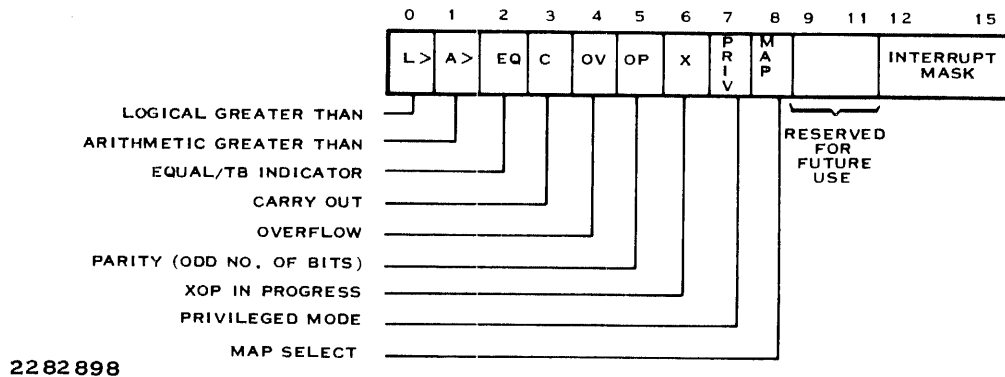

Figure 4-1.   990/10A Status Register

4.5 TMS9902 ASYNCHRONOUS COMMUNICATION CONTROLLER

The 990/10A uses the TMS 9902 asynchronous communication controller (ACC) to interface to RS-232C EIA devices such as printers, terminals, and asynchronous modems. The CRU serves to interface the ACC with the rest of the 990/10A. This interface consists of five address lines, an enable signal, and three CRU control lines (Figure 4-2). When the enable signal is active, the five select lines address the CRU bit to be accessed. When data is transferred to the ACC from the microprocessor, CRUOUT contains the valid data bit, and is strobed by CRUCLK. When data is transferred from the ACC to the microprocessor, CRUIN contains the data bit from the ACC. Level shifters translate the RS-232C levels to and from the ACC.

```
TMS 9902

                    INTERRUPT ─────────────────────────►
         ___
          φ  O◄──── CLOCK ──────────────────────────

CRUCLK  ◄────────────────────────────────────

CRUOUT  ◄────────────────────────────────────

 CRUIN  ─────────────────────────────────────►

    S0  ◄──── ADR10 ──────────────────────────
    S1  ◄──── ADR11 ──────────────────────────
    S2  ◄──── ADR12 ──────────────────────────
    S3  ◄──── ADR13 ──────────────────────────
    S4  ◄──── ADR14 ──────────────────────────

    __                     ┌─────────────┐
    CE  O◄────────O        │   DECODE    │◄═══ ADR0 ─ ADR9
                           └─────────────┘
```

2283236
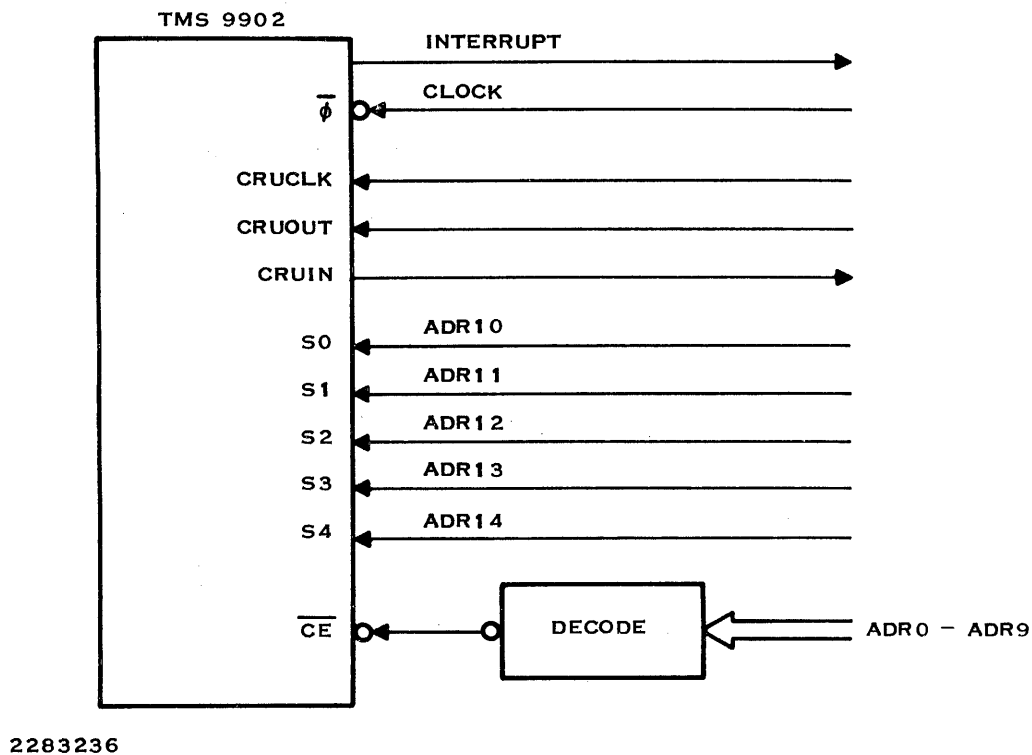
Figure 4-2.   Asynchronous Communication Port Interface

4.5.1 CRU Implementation External to the TMS 9902

In addition to the functions provided by the TMS 9902, there are five functions implemented external to the TMS 9902. These functions are used for communication with asynchronous modems and for test purposes. The CRU addresses and functions are as follows.

- 1740 – Data terminal ready (output). An SBO enables DTR.

- 1742 – Analog loopback (output). An SBO enables analog loopback.

- 1742 – Ring indicator (input). A TB true indicates ringing.

- 174C – Interrupt enable (output). An SBO enables the interrupt.

- 174E – EIA port communication enable (output). An SBO enables the port.

4.5.2 Interrupt Output

The ACC interrupt output is active when any of the following conditions occur:

- Data set ready (DSR-) or clear to send (CTS-) changes levels (data set change enable (DSCH) = 1)

- A character has been received and stored in the receive buffer register (RBL = 1)

- The transmit buffer register is empty (XBRE = 1)

- The selected time interval has elapsed (TIMELP = 1) CRU address > 174C enables the interrupt from the ACC.

4.5.3 Control and Data Output

The high order bits from the address bus are used to select the ACC, and the five LSBs connect to the ACC address select input. These bits are at CRU base address > 1700. Table 4-1 describes the output bit assignments for the ACC.

Table 4-1. CRU Output Bits to the ACC

Bit	Description
Bit 31 (RESET)–	Writing a one or zero to bit 31 resets the device, disabling all interrupts, initializing the transmitter and receiver, setting $\overline{RTS}$ inactive (high), setting all register load control flags (LDCTRL, LDIR, LRDR, and LXDR) to a logic one level, and resetting the BREAK flag. Do not perform any other input or output operations for 11 clock cycles after issuing the RESET command.
Bit 30–Bit 22	Not used.
Bit 21 (DSCENB)–	Data Set Change Interrupt Enable. Writing a one to bit 21 causes the $\overline{INT}$ output to be active (low) whenever data set status change (DSCH) is a logic one. Writing a zero to bit 21 disables DSCH interrupts. Writing either a one or zero to bit 21 resets DSCH.
Bit 20 (TIMENB)–	Timer Interrupt Enable. Writing a one to bit 20 activates $\overline{INT}$ output whenever timer elapsed (TIMELP) is a logic one. Writing a zero to bit 20 disables TIMELP interrupts. Writing either a one or zero to bit 20 resets TIMELP and timer error (TIMERR).
Bit 19 (XBIENB)–	Transmit Buffer Interrupt Enable. Writing a one to bit 19 activates the $\overline{INT}$ output whenever transmit buffer register empty (XBRE) is a logic one. Writing a zero to bit 19 disables XBRE interrupts. XBRE interrupts to be disabled. The state of XBRE is not affected by writing to bit 19.
Bit 18 (RIENB)–	Receiver Interrupt Enable. Writing a one to bit 18 activates the $\overline{INT}$ output whenever receive buffer register loaded (RBRL) is a logic one. Writing a zero to bit 18 disables RBRL interrupts. Writing either a one or zero to bit 18 resets RBRL.
Bit 17 (BRKON)–	Break On. Writing a one to bit 17 sends the transmitter serial data output (XOUT) to a logic zero whenever the transmitter is active and the transmit buffer register (XBR) and the transmit shift register (XSR) are empty. While BRKON is set, loading of characters into the XBR is inhibited. Writing a zero to bit 17 resets BRKON and the transmitter resumes normal operation.
Bit 16 (RTSON)–	Request-to-Send On. Writing a one to bit 16 causes the $\overline{RTS}$ output to be active (low). Writing a zero to bit 16 sends $\overline{RTS}$ to a logic one after the XSR and XBR are empty, and BRKON is reset. Thus, the $\overline{RTS}$ output does not become inactive (high) until after character transmission has been completed.
Bit 15 (TSTMD)–	Test Mode. Writing a one to bit 15 internally connects $\overline{RTS}$ to $\overline{CTS}$, XOUT to RIN, $\overline{DSR}$ is held low, and the interval timer operates at 32 times its normal rate. Writing a zero to bit 15 reenables normal device operation.

Table 4-1.   CRU Output Bits to the ACC (Continued)

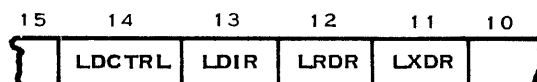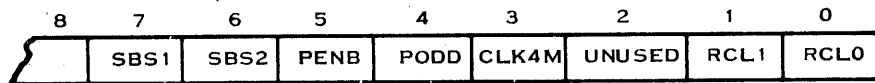| Bit	Description
Bit 14 (LDCTRL)-	Load Control Register. Writing a one to bit 14 sets LDCTRL to a logic one. When LDCTRL = 1, any data written to bits 0 through 7 are directed to the control register. Note that LDCTRL is also set to a logic one when a one or zero is written to bit 31 (RESET). Writing a zero to bit 14 resets LDCTRL to a logic zero, disabling loading of the control register. LDCTRL is also automatically reset to a logic zero when a datum is written to bit 7 of the control register which normally occurs as the last bit written when loading the control register with an LDCR instruction.
Bit 13 (LDIR)-	Load Interval Register. Writing a one to bit 13 sets LDIR to a logic one. When LDIR = 1 and LDCTRL = 0, any data written to bits 0 through 7 are directed to the interval register. Note that LDIR is also set to a logic one when a datum is written to bit 31 (RESET); however, interval register loading is not enabled until LDCTRL is set to a logic zero. Writing a zero to bit 13 resets LDIR to a logic zero, disabling loading of the interval register. LDIR is also automatically reset to logic zero when a datum is written to bit 7 of the interval register, which normally occurs as the last bit written when loading the interval register with an LDCR instruction.
Bit 12 (LRDR)-	Load Receive Data Rate Register. Writing a one to bit 12 sets LRDR to a logic one. When LRDR = 1, LDIR = 0, and LDCTRL = 0, any data written to bits 0 through 10 are directed to the receive data rate register. Note that LRDR is also set to a logic one when a datum is written to bit 31 (RESET); however, receive data rate register loading is not enabled until LDCTRL and LDIR have been set to a logic zero. Writing a zero to bit 12 resets LRDR to a logic zero, disabling loading of the receive data rate register. LRDR is also automatically reset to logic zero when a datum is written to bit 10 of the receive data rate register, which normally occurs as the last bit written when loading the receive data rate register with an LDCR instruction.
Bit 11 (LXDR)-	Load Transmit Data Rate Register. Writing a one to bit 11 sets LXDR to a logic one. When LXDR = 1, LDIR = 0, and LDCTRL = 0, any data written to bits 0 through 10 is directed to the transmit data rate register. Note that loading of both the receive and transmit data rate registers is enabled when LDCTRL 0, LDIR = 0, LRDR = 1, and LXDR = 1; thus, these two registers can be loaded simultaneously when data is received and transmitted at the same rate. LXDR is also set to a logic one when a datum is written to bit 31 (RESET); however, transmit data rate register loading is not enabled until LDCTRL and LDIR have been reset to logic zero. Writing a zero to bit 11 resets LXDR to logic zero, disabling loading of the transmit data rate register. Since bit 11 is the next bit addressed after loading the transmit data rate register, the register can be loaded and the LXDR flag reset with a single LDCR instruction where 12 bits (bits 0 through 11) are written, with a zero written to bit 11.

15	14	13	12	11	10
	LDCTRL	LDIR	LRDR	LXDR	

Table 4-1.   CRU Output Bits to the ACC (Continued)

Bit				Description

LDCTRL	LDIR	LRDR	LXDR	Register Enabled
1	X	X	X	Control Register
0	1	X	X	Interval Register
0	0	1	X	Receive Data Rate Register
0	0	X	1	Transmit Data Rate Register
0	0	0	0	Transmit Buffer Register

Control Register Bits

When the register load control flag identifies the control register, data in bit positions 0 through 7 select character length, device clock operation, parity, and the number of stop bits for the transmitter circuit.

8	7	6	5	4	3	2	1	0
	SBS1	SBS2	PENB	PODD	CLK4M	UNUSED	RCL1	RCL0

Name	Description
SBS1	Stop Bit Select
SBS2	
PENB	Parity Enable
PODD	Odd Parity Select
CLK4M	$\overline{\phi}$ Input Divide Select
—	Not Used
RCL1	
RCL0	Character Length Select

Bits 7 and 6 (SBS1 and SBS2)—

Stop Bit Selection. The number of stop bits to be appended to each transmitter character is selected by bits 7 and 6 of the control register as shown below. The receiver only tests for a single stop bit, regardless of the status of bits 7 and 6.

SBS1 Bit 7	SBS2 Bit 6	Number of Transmitted Stop Bits
0	0	1½
0	1	2
1	0	1
1	1	1

Bits 5 and 4 (PENB and PODD)—

Parity Selection. Bits 4 and 5 of the following control register select the type of parity to be generated for transmission and detection for reception. When parity is enabled (PENB = 1), the parity bit is transmitted and received in addition to the number of bits selected for the character length. Odd parity is such that the total number of ones in the character and parity bit, exclusive of stop bit(s), will be odd. For even parity, the total number of ones will be even.

Table 4-1.   CRU Output Bits to the ACC (Continued)

Bit	Description

PENB Bit 5	PODD Bit 4	Parity
0	0	None
0	1	None
1	0	Even
1	1	Odd

Bit 3 (CLK4M) $\bar{\phi}$ Input Divide Select. The $\bar{\phi}$ input to the TMS 9902 ACC generates internal dynamic logic clocking and establishes the time base for the interval timer, transmitter, and receiver. The $\bar{\phi}$ input is internally divided by either 3 or 4 to generate the 2-phase internal clocks required for MOS logic, and to establish the basic internal operating frequency (f_{int}) and internal clock period (t_{int}). When bit 3 of the control register is set to a logic 1 (CLK4M = 1), $\bar{\phi}$ is internally divided by 4, and when CLK4M = 0, $\bar{\phi}$ is divided by 3. For example, when $f\bar{\phi} = 4$ MHz, as in a standard 4 MHz TMS 9900 system, and CLK4M = 1, $\bar{\phi}$ is internally divided by 4 to generate an internal clock period t_{int} of 1 μs. The following figure shows the operation of the internal clock divider circuitry. When $f\bar{\phi} \leqslant 3.0$ MHz, CLK4M should be set to a logic 0.

$\bar{\phi}$ External Input $\rightarrow$ ÷ n { n=4 if CLK4m=0 ; n=3 if CLK4m=1 } $\rightarrow$ ϕ1 int / ϕ2 int $\rightarrow$ to internal logic

$$f_{int} = \frac{f\bar{\phi}}{n}$$

$$t_{int} = \frac{1}{f_{int}} = \frac{n}{f\bar{\phi}}$$
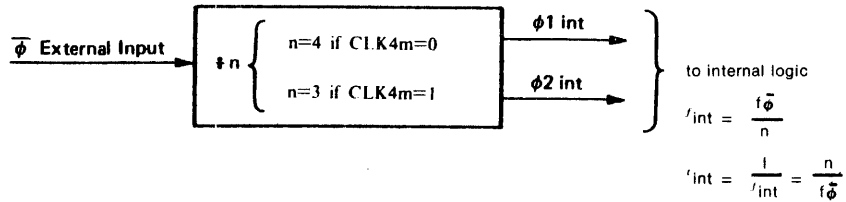
Bits 1 and 0 RCL1 and RCL0)—   Character Length Select. Bits 1 and 0 of the following control register determine the number of data bits in each transmitted and received character.

RCL1 Bit 1	RCL0 Bit 0	Character Length
0	0	5 bits
0	1	6 bits
1	0	7 bits
1	0	8 bits

Table 4-1. CRU Output Bits to the ACC (Continued)

Bit	Description
Interval Register Bits	When the interval register flag is set, data in bit positions 0 through 7 indicate the rate at which interrupts are generated by the interval timer of the TMS 9902.

8	7	6	5	4	3	2	1	0
	TMR7	TMR6	TMR5	TMR4	TMR3	TMR2	TMR1	TMR0
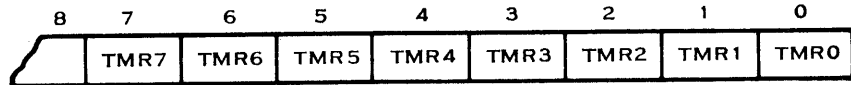
As an example, if the interval register is loaded with a value of 80_{16} (128_{10}), the interval at which timer interrupts are generated is $'ITVL = 'int * 64 * M = (1 \mu s)(* 64)(* 128) = 8.192$ ms. when $'int = 1 \mu s$.

signal	ϕ INT $\rightarrow$	$\div 64$ $\rightarrow$	$\div m$ $\quad m = (TMR7 \quad TMR0)$ $\rightarrow$ TIMELP $\rightarrow$
frequency	$'int$	$'int/64$	$'int/(64)(m)$
time	$'int$	$64 \, 'int$	$64 \, m \, 'int$
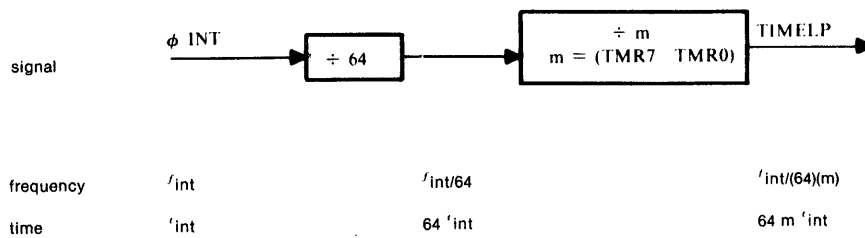
TIMER INTERVAL SELECTION

Bit	Description
Receive Data-Rate Register Bits	When the receive data-rate register flag is set, data in bit positions 0 through 10 select the bit rate at which data is received from a source external to the TMS 9902.

11	10	9	8	7	6	5	4	3	2	1	0
	RDV8	RDR9	RDR8	RDR7	RDR6	RDR5	RDR4	RDR3	RDR2	RDR1	RDR0
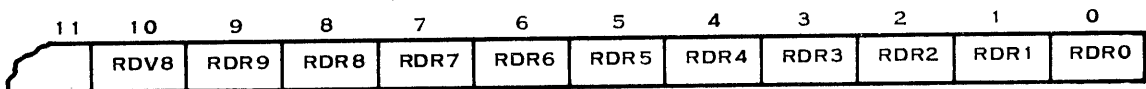
Table 4-1. CRU Output Bits to the ACC (Continued)

Bit	Description
	The following diagram describes the manner in which the receive data rate is established. Basically, 2 programmable counters determine the interval for ½ the bit period of receive data. The first counter divides the internal system clock frequency (f_{int}) by either 8 (RDV8 = 1) or 1 (RDV8 = 0). The second counter has 10 stages and can be programmed to divide its input signal by any value from 1 (RDR9–RDR0 = 0000000001) to 1023 (RDR8–RDR0 = 1111111111). The frequency of the output of the second counter (f_{RHBT}) is double the receive-data rate. Register is loaded with a value of 11000111000, RDV8 = 1, and RDR9–RDR0 = 1000111000 = 238_{16} = 568_{10}. Thus, for f_{int} = 1 MHz, the receive-data rate = $1 \times 10^6 \div 8 \div 568 \div 2 =$ 110.04 bits per second.

signal   φINT   ÷ m   m = 8 (RDV8 = 1)   or m = 1 (RDV8 = 0)   ÷ n   n = (RDR9–RDR0)   RHBT

frequency f_{int} $\dfrac{f_{int}}{m}$ $f_{int} = f_{RHBT}$ $\dfrac{1}{m^*n}$
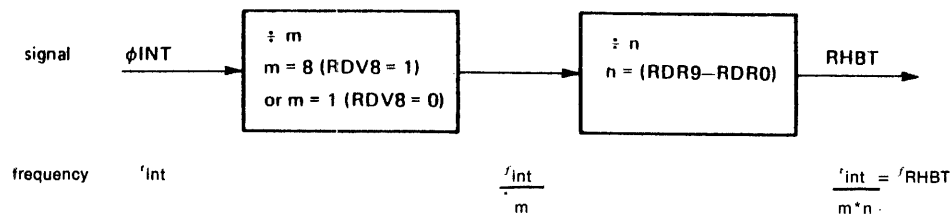
RECEIVE DATA RATE SELECTION

Quantitatively, the receive-data rate f_{RCV} may be described by the following algebraic expression:

$$f_{RCV} = \frac{f_{RHBT}}{2} = \frac{f_{int}}{2mn} = \frac{f_{int}}{(2)(8^{RDV8})(RDR9-RDR0)}$$

Bit	Description
Transmit Data-Rate Register Bits	When the transmit data-rate register flag is set, data in bit positions 0 through 10 select the bit rate that data is to be transmitted to a destination external to the TMS 9902.

11	10	9	8	7	6	5	4	3	2	1	0
	XDV8	XDR9	XDR8	XDR7	XDR6	XDR5	XDR4	XDR3	XDR2	XDR1	XDR0
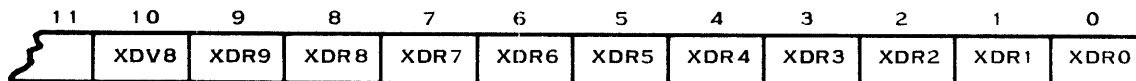
Selection of the transmit data rate is accomplished with the transmit data rate register in the same way that the receive data rate is selected with the receive data rate register. The algebraic expression for the transmit data rate f_{XMT} is:

$$f_{XMT} = \frac{f_{XHBT}}{2} = \frac{f_{int}}{(2)(8^{XDV8})(XDR9 - XDB0)}$$

For example, if the transmit data rate register is loaded with a value of 00110100001, XDV8 = 0, and XDR9–XDR0 = $1A1_{16}$ = 417, the transmit data rate = $1 \times 10^6 \div 2 \div 417 = 1199.04$ bits per second.

Table 4-1.   CRU Output Bits to the ACC (Continued)

Bit	Description
Transmit Buffer Register Bits	When the transmit buffer register flag is set, data in bit positions 0 through 7 contains the next character to be transmitted to a destination, external to the TMS 9902. The transmit buffer register stores the next character to be transmitted. When the transmitter is active, the contents of the transmit buffer register are transferred to the transmit shift register each time the previous character has been completely transmitted. The bit address assignments for the transmit buffer register are shown below. All eight bits should be transferred into the register, regardless of the selected character length. The extraneous high-order bits will be ignored for transmission purposes; however, loading of bit 7 is internally detected to cause the transmit buffer register empty (XBRE) status flag to get reset.

8	7	6	5	4	3	2	1	0
/	XBR7	XBR6	XBR5	XBR4	XBR3	XBR2	XBR1	XBR0

4.5.4   Status and Data Input

The CRU is used to read status and data information from the ACC. Table 4-2 defines the CRU bit assignments for input from the ACC. These bits are at CRU base address > 1700.

Table 4-2.   CRU Input Bits from the ACC

Bit	Description
Bit 31 (INT)–	INT = DSCINT + TIMINT + XBINT + RBINT. The interrupt output (INT) is active when this status signal is a logic 1.
Bit 30 (FLAG)–	FLAG = LDCTRL + LDIR + LRDR + LXDR + BRKON. When any of the register load control flags or BRKON is set, FLAG = 1.
Bit 29 (DSCH)–	Data Set Status Change Enable. DSCH is set when the DSR or CTS input changes state. To ensure recognition of the state change, DSR or CTS must remain stable in its new state for a minimum of two internal clock cycles. DSCH is reset by an output to bit 21 (DSCENB).
Bit 28 (CTS)–	Clear to Send. The CTS signal indicates the inverted status of the $\overline{CTS}$ device input.
Bit 27 (DSR)–	Data Set Ready. The DSR signal indicates the inverted status of the $\overline{DSR}$ device input.
Bit 26 (RTS)–	Request to Send. The RTS signal indicates the inverted status of the $\overline{RTS}$ device output.
Bit 25 (TIMELP)–	Timer Elapsed. TIMELP is set each time the interval timer decrements to 0. TIMELP is reset by an output to bit 20 (TIMENB).

Table 4-2. CRU Input Bits from the ACC (Continued)

Bit	Description
Bit 24 (TIMERR)–	Timer Error. TIMERR is set whenever the interval timer decrements to 0 and TIMELP is already set, indicating that the occurrence to TIMELP was not recognized and cleared by the CPU before subsequent intervals elapsed. TIMERR is reset by an output to bit 20 (TIMENB).
Bit 23 (XSRE)–	Transmit Shift Register Empty. When XSRE = 1, no data is currently being transmitted and the XOUT output is at logic 1 unless BRKON is set. When XSRE = 0, transmission of data is in progress.
Bit 22 (XBRE)–	Transmit Buffer Register Empty. When XBRE = 1, the transmit buffer register does not contain the next character to be transmitted. XBRE is set each time the contents of the transmit buffer register are transferred to the transmit shift register, XBRE is reset by an output to bit 7 of the transmit buffer register (XBR7), indicating that a character has been loaded.
Bit 21 (RBRL)–	Receive Buffer Register Loaded. RBRL is set when a complete character has been assembled in the receive shift register and the character is transferred to the receive buffer register. RBRL is reset by an output to bit 18 (RIENB).
Bit 20 (DSCINT)–	Data Set Status Change Interrupt. DSCINT = DSCH (input bit 29) * DSCENB (output bit 21). DSCINT indicates the presence of an enabled interrupt caused by the changing of state of DSR or CTS.
Bit 19 (TIMINT)–	Timer Interrupt. TIMINT = TIMELP (input bit 25) * TIMENB (output bit 20). TIMINT indicates the presence of an enabled interrupt caused by the interval timer.
Bit 17 (XBINT)–	Transmitter interrupt. XBINT = XBRE (input bit 22) * XBIENB (output bit 19). XBINT indicates the presence of an enabled interrupt caused by the transmitter.
Bit 16 (RBINT)–	Receiver Interrupt. RBINT = RBRL (input bit 21) * RIENB (output bit 18). RBINT indicates the presence of an enabled interrupt caused by the receiver.
Bit 15 (RIN)–	Receive Input. RIN indicates the status of the RIN input to the device.
Bit 14 (RSBD)–	Receive Start Bit Detect. RSBD is set one-half bit time after the 1-to-0 transition of RIN indicating the start bit of a character. If RIN is not still 0 at this point in time, RSBD is reset. Otherwise, RSBD remains true until the complete character has been received. This bit is normally used for testing purposes.
Bit 13 (RFBD)–	Receive Full Bit Detect. RFBD is set one bit time after RSBD is set to indicate the sample point for the first data bit of the received character. RSBD is reset when the character has been completely received. This bit is normally used for testing purposes.

Table 4-2.   CRU Input Bits from the ACC (Continued)

Bit	Description
Bit 12 (RFER)–	Receive Framing Error. RFER is set when a character is received in which the stop bit, which should be a logic 1, is a logic 0. RFER should only be read when RBRL (input bit 21) is a 1. RFER is reset when a character with the correct stop bit is received.
Bit 11 (ROVER)–	Receive Overrun Error. ROVER is set when a new character is received before the RBRL flag (input bit 21) is reset, indicating that the CPU failed to read the previous character and reset RBRL before the present character is completely received. ROVER is reset when a character is received and RBRL is 0 when the character is transferred to the receive buffer register.
Bit 10 (RPER)–	Receive Parity Error. RPER is set when a character is received in which the parity is incorrect. RPER is reset when a character with correct parity is received.
Bit 9 (RCVERR)–	Receive Error. RCVERR = RFER + ROVER + RPER. RCVERR indicates the presence of an error in the most recently received character.
Bit 7–Bit 0 (RBR7–RBR0)–	Receive Buffer Register. The receive buffer register contains the most recently received character. For character lengths of fewer than 8 bits the character is right justified, with unused most significant bit(s) all zero(s). The presence of valid data in the receive buffer register is indicated when RBRL is a logic 1.

4.6   MEMORY ERROR LOG

The following paragraphs describe features of the memory error log.

4.6.1   Error Correction Test

The memory controller may be placed in a diagnostic mode under program control. Two TILINE peripheral control space addresses for the control function and the error logging function are switch selectable from the list in Table 4-3. (The remaining two addresses are occupied by the multiprocessor interface.) Writing to the first address controls the error correction and control (ECC) logic in the manner described in the section on output bits. Reading from the addresses provides the memory error logging functions described in the section on input bits.

Table 4-3.   Memory Control TPCS Addresses

Logical Address (Hexadecimal)	Physical Address (Hexadecimal)	SW2	SW3	SW4	SW5	SW6	SW7	SW8
F800-06	FFC00-03	off	off	off	off	off	off	off
F808-0E	FFC04-07	off	off	off	off	off	off	on
:	:	:	:	:	:	:	:	:
FBF0-F6	FFDF8-FB	on	on	on	on	on	on	off
FBFC-FE	FFDFC-FF	on	on	on	on`	on	on	on

Manipulation of the bits in the memory error log causes the controller to operate in the following modes.

1.   Error detection and correction are inhibited in banks as specified by bits in the memory error log. Banks of the memory can be specified individually. In this mode, data is not affected by the ECC logic.

2.   The ECC logic is modified such that on a write cycle the correction bits generated from the data word are not stored in the memory but are replaced by the six most significant data bits (DATA0 thru DATA5). During a read operation, the correction logic is inhibited and the correction bits stored in the memory replace the six most significant data bits.

4.6.2   Output Bits
The bit assignments for a word written into word 0 are as shown in Figure 4-3.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

```
B   B   E   M               R
A   A   X   O               E
N   N   P   D               S
K   K       E               E
                            T
A   B
```
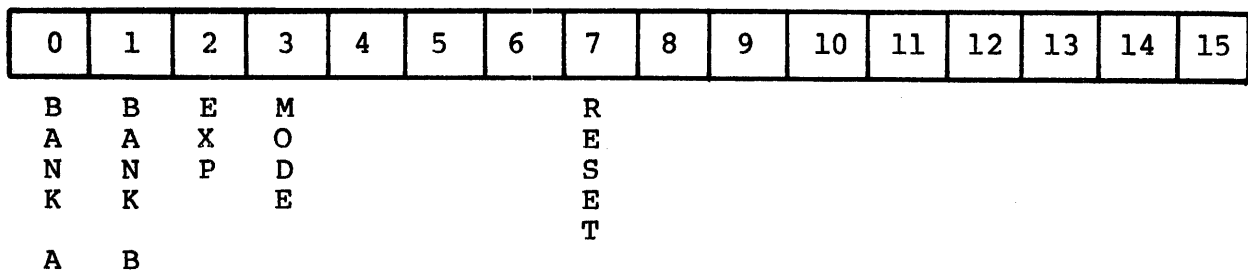
Figure 4-3.   Diagnostic Control Output Bit Assignment — Word 0

BIT 0 and BIT 1 — These bits select 32K banks of memory within the first row of RAM chips that will be controlled by the diagnostic mode. This feature is necessary if diagnostic memory tests are to be run on a computer having a single row of 64K RAM memory (128K bytes) and the diagnostic program is resident in the same memory.

BIT 2 — This bit selects the diagnostic function for all rows of memory after the first. Thus a computer with more than one row of memory could perform diagnostic tests on all memory above the first 64K addresses with a program resident in the first 64K.

BIT 3 — When set to zero, this bit disables error detection and correction in the banks or rows selected by bits 0, 1, and 3. When set to one, this bit swaps the most significant data bits with the error correction bits in the banks or rows selected.

BIT 4 through Bit 6 — No effect.

BIT 7 — When set to one, this bit clears the error latches and error LEDs. It also clears all of the diagnostic control features to reenable normal error correction and addressing. This bit is not latched, so a single write cycle reconfigures the correction logic.

BIT 8 through Bit 15 — No effect.

4.6.3 Input Bits
The significance of bits read from Words 0 and 1 of the Error Control Log are shown in Figure 4-4 and Figure 4-5.

WORD 0

```
+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+
| 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 | 11 | 12 | 13 | 14 | 15 |
+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+
+——BASE  ADDRESS  OF  MEMORY————+    | +  |  +  |  +  | +   |  +   | +ID CODE+
      (bit  0 MSB) (inverted  data)  |        EXP                     (10)
      (Bits  4,5,6,&7  always  high) BANKA |          MODE
                                     BANKB                      CORRECTABLE
                                                                ERROR  OCCURRED
                UNCORRECTABLE  ERROR  OCCURRED-+
```
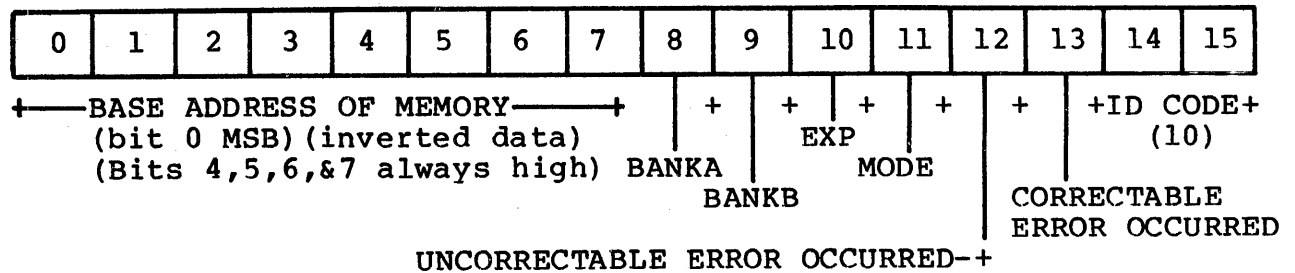
Figure 4-4. Error Log Input Bit Assignment — Word 0

WORD 1

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

```
+SIZE OF+————————ROW  IN  ERROR—+——BIT  IN  ERROR————+—SAME  AS—+
 MEMORY                 (Bit  2  MSB)        (Bit  8  MSB)       WORD  0
```
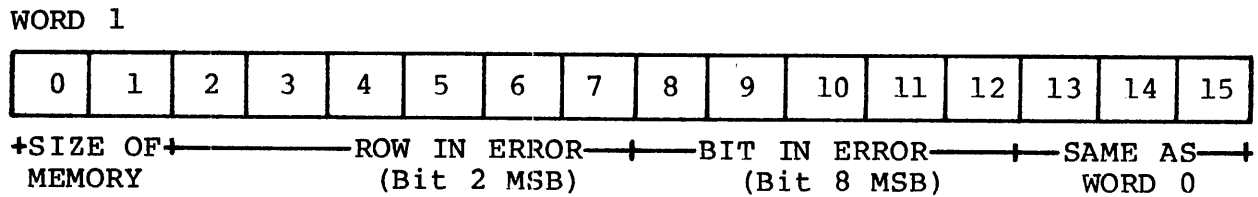
Figure 4-5. Error Log Input Bit Assignment — Word 1

Bit 0 through Bit 3 — These bits show the value of four switches (LB0- through LB3-) that determine the lower bound address of all memory controlled with the CCC.

Bit 4 through Bit 7 — These bits are fixed at logic 1 to be compatible with previous memory error logs where memory bounds were switch selectable on 4K boundaries. With 64K RAMS, the lower bound is selectable only on 128K-byte boundaries.

BIT 8 — This bit echoes Bit 0 of output Word 0

BIT 9 — This bit echoes Bit 1 of output Word 0

BIT 10 — This bit echoes Bit 2 of output Word 0

BIT 11 — This bit echoes Bit 3 of output Word 0

BIT 12 — This bit shows the value of the double-bit error latch. Double bit errors will generally cause a level two interrupt but this bit may be used for validation or for diagnostic testing.

BIT 13 — This bit shows the value of the single-bit error latch. It will be a one after the first occurrence of a single bit error.

Bit 14 and Bit 15 — These bits are fixed with a one-zero pattern so that the CPU can recognize these words as originating within the 990/10A and hence distinguish them and their significance from other memory error logs.

Bit 0 and Bit 1 — These bits show the size of memory being controlled in the following manner:

    00 = 1 row of RAM chips
    01 = 2 rows of RAM chips
    10 = 3 rows of RAM chips
    11 = 4 rows of RAM chips

Bit 2 through Bit 7 — These bits report the value of the row-in-error latch. The bit assignment followed is that of the 16K memory controllers already being manufactured. Thus bits 6 and 7 report the row-in-error as 16K word blocks within the 64K RAM chips. This information should be ignored for the 990/10A. The significance of the bits is defined in the following manner:

    0000XX   Error in first row (Row 0)
    0001XX   Error in second row (Row 1)
    0010XX   Error in third row (Row 2)
    0011XX   Error in fourth row (Row 3)

Bit 8 through Bit 12 — These bits report the value of the bit-in-error latch. Thus they provide the software with a binary representation of the chip causing a single bit error.

Bit 13 through Bit 15 — These bits are duplicated from Word 0.

Appendix A

Using the Multiprocessor Interface

A.1 GENERAL

This appendix describes the features and use of the multiprocessor capability of the 990/10A. The basic operation of the multiprocessor interface is discussed first. Two possible configurations, a typical sequence of operations, and a discussion of some system considerations are included.

A.2 MULTIPROCESSOR OPERATION

Supervisory communication between the host and auxiliary processors is by means of interrupts. Two words in the TILINE peripheral control space (TPCS) are used to set and to clear these interrupts. The addresses of these two words in the TPCS are selected by the on-board switches for the memory error log. The host controls word two of the error log and the auxiliary controls word three. (Words zero and one are the memory error log itself). The host sends interrupts to the auxiliary processor using the auxiliary's level three interrupt, and the auxiliary sends interrupts to the host using the backpanel interrupt on P1-66. The interrupt to begin an interaction between the host and auxiliary is designated as an attention interrupt, and the response is designated as an acknowledge interrupt. The hardware for both host and auxiliary functions is implemented on each 990/10A board.

NOTE

An auxiliary processor can access another auxiliary processor's multiprocessor interface through the TPCS, but the acknowledge interrupt will occur at the host, limiting the usefulness of this capability.

A.2.1 Host Operation
As a host, the 990/10A processor resides in slot 1 of the chassis with the option switch in the slot 1 position, and the host/auxiliary jumper installed. The slot one switch causes the 990/10A to perform the following functions:

- It drives the 14 CRU address bits, the CRU clock (STORECLK), and the CRU data signal (CRUDAT).

- It drives the 24 CRU module select lines.

- It receives the backpanel interrupts.

- It is connected to the front panel.

- On power up, it performs a self-test function and extinguishes the front panel FAULT LED.

In addition to the pencil switch that indicates that the processor is in slot 1, the host/auxiliary jumper can be read through the self-test register to determine if the processor is a host or an auxiliary. It is assumed that 990/10As can serve as an input/output (I/O) processor in slot 1 of an expansion chassis; in that case, the slot 1 switch alone is insufficient to distinguish between host and auxiliary. As an auxiliary in slot 1 of an expansion chassis, the 990/10A can perform many of the functions from the previous list (such as service interrupts from video display terminals) and yet be subordinate to a host processor. The host/auxiliary jumper causes the self-test not to set an idle condition after self-test is complete. The processor executes the appropriate code for the option jumpers and front panel present.

A.2.2 Auxiliary Operation

When used as an auxiliary in a slot other than slot 1, many of the 990/10A functions are disabled by the slot 1 switch. Specifically, when not in slot 1, an auxiliary has the following restrictions:

- The auxiliary cannot drive the CRUBIT address lines, the CRU clock, or CRU data.

- The auxiliary cannot drive the 24 module select lines.

- The auxiliary is disconnected from most backpanel interrupts and can be interrupted only by the following signals:

  - The NMI from a front panel (connected to the auxiliary).

  - TILINE power reset (level 0).

  - TILINE power failure warning pulse (level 1).

  - Error conditions (level 2).

  - Multiprocessor interrupt (level 3).

  - Real-time clock (level 5 or 15).

  - EIA port interrupt (level 8).

  - Level 14 in some cases (paragraph 1.5.5.7).

- The auxiliary must have the host/auxiliary jumper removed. This jumper causes the processor to go to IDLE after completion of self-test. An auxiliary processor is typically assigned a task by the host by loading code into the auxiliary's on-board RAM, and then interrupting the auxiliary from IDLE with an attention interrupt. Self-test sets the map file of an auxiliary so that interrupts are vectored through RAM locations in the auxiliary processor.

A.2.3 Host Processor Interface

The host can interrupt the auxiliary processor by setting the attention interrupt bit to one in the auxiliary's multiprocessor interface. The auxiliary can enable this interrupt by setting the attention interrupt mask to one, but self-test initially leaves the mask reset to zero. The host must initialize the auxiliary with an appropriate interrupt vector and then enable the attention mask before generating the attention interrupt. The auxiliary acknowledges the interrupt by clearing the attention interrupt from the host. This automatically causes an acknowledge interrupt to the host. The host enables acknowledge interrupts by setting the acknowledge mask to one. Table A-1 lists the bit definitions of the host interrupts.

A.2.4 Auxiliary Processor Interface

An auxiliary processor can interrupt the host by setting the attention interrupt bit. If the host sets the attention interrupt mask, the host is interrupted from the backplane. The host acknowledges the interrupt by clearing the attention interrupt that automatically generates an acknowledge interrupt back to the auxiliary. Table A-2 lists the bit definitions of the auxiliary interrupts.

Table A-1. Interface Bits As Seen by Host

Address	Bit	Value	Meaning When Read	Value	Meaning When Written
Word2	0	0	Attn to aux inactive	0	Clear attn int (test mode)
		1	Attn to aux active	1	Send attn int to aux
Word2	1	0	Ack int from aux is masked	0	Mask out and clear Acknowledge int from aux
		1	Ack int from aux is unmasked	1	Enable ack int from aux
Word2	2	0	Ack to aux not active	0	Clear ack int (test mode)
		1	Ack to aux active	1	Clear attn int from aux and send ack int
Word2	3	0	Attn from aux masked	0	Mask out attn int from aux
		1	Attn from aux unmasked	1	Enable attn int from aux
Word2	4	0	Attn to host inactive		NA
		1	Attn to host active		
Word2	5	0	Ack int from host masked		NA
		1	Ack int from host unmasked		
Word2	6	0	Ack to host not active		NA
		1	Ack to host active		
Word2	7	0	Attn from host masked		NA
		1	Attn from host unmasked		
Word2	8	0	Aux processor busy		NA
		1	Aux processor idle		
Word2	9	0	Aux passed self-test		NA
		1	Aux failed self-test		

Table A-2.   Interface Bits As Seen by Auxiliary

Address	Bit	Value	Meaning When Read	Value	Meaning When Written
Word3	0	0	Attn to host inactive	0	Clear attn int (test mode)
		1	Attn to host active	1	Send attn int to host
Word3	1	0	Ack int from host is masked	0	Mask out and clear Acknowledge int from host
		1	Ack int from host is unmasked	1	Enable Ack int from host
Word3	2	0	Ack to host not active	0	Clear ack int (test mode)
		1	Ack to host active	1	Clear attn int from host and send ack int
Word3	3	0	Attn from host masked	0	Mask out attn int from host
		1	Attn from host unmasked	1	Enable attn int from host
Word3	4	0	Attn to aux inactive		NA
		1	Attn to aux active		
Word3	5	0	Ack int from aux masked		NA
		1	Ack int from aux unmasked		
Word3	6	0	Ack to aux not active		NA
		1	Ack to aux active		
Word3	7	0	Attn from aux masked		NA
		1	Attn from aux unmasked		

A.2.5   TILINE Address Assignment

The starting address of memory on the 990/10A is switch-selectable. For a single processor, these switches are normally set at zero. Address translation does not occur going from TILINE to the internal bus. If multiple 990/10As are used in a single chassis, set the switches so that all memory in the system has a unique physical address; in other words, all memory in the system is global (with the exception of the scratch pad RAM used for self-test). All memory can be accessed by any master device in the system. Figure A-1 shows two processors properly configured with 256 kilobytes each.
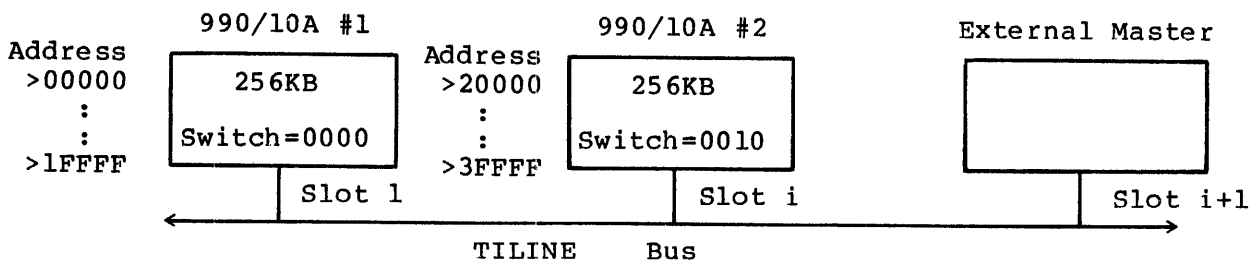
Figure A-1.   Properly Configured Multiprocessor Memory

Figure A-2 shows a configuration where the physical addresses are overlapping. When accessing memory in the range > 10000 to > 1FFFF, 990/10A number one sees its on-board memory. Likewise 990/10A number two sees its on-board memory when accessing in that range. In the range > 20000 to > 2FFFF, processor number one performs a TILINE master cycle to the memory of 990/10A number two. However, if the external master were to access > 15000, both processors respond. The memory between > 10000 and > 1FFFF can be considered local (as opposed to global) for 990/10A number two; however, the problem of both responding to an external master makes this configuration undesirable.

A.3 MULTIPROCESSOR CONFIGURATIONS

A typical configuration of a multiprocessor system is shown in Figure A-3.

In this configuration, the host processor is responsible for all I/O. The auxiliary processors' operating systems require a host to be present in the system. The host uses the auxiliary processors to execute tasks. The auxiliary processors are loaded with a kernel that supports the multiprocessor interface, and to a limited extent, the communications port (if applicable). The kernel used by the auxiliary processor does not support a user without interaction with the host.

A second possible configuration is shown in Figure A-4.

In Figure A-4, the host processor communicates with the auxiliary processors 1, 2, 3 and the slot one auxiliary processor, as in Figure A-3. The slot one auxiliary processor controls auxiliary processor S1 with the same relationship as the host processor controls its auxiliary processors. The difference between this configuration and the previous configuration is that the slot one auxiliary processor is loaded with a kernel that allows it to schedule tasks among its auxiliary processors and to perform I/O. The slot one auxiliary and auxiliary processor 3 share the same interrupt level to the host. Thus the slot one auxiliary processor must be capable of generating an interrupt that can be passed by the TILINE coupler to the host. One primary advantage of this configuration is the expansion of the host's I/O capabilities by utilizing the slot one auxiliary.
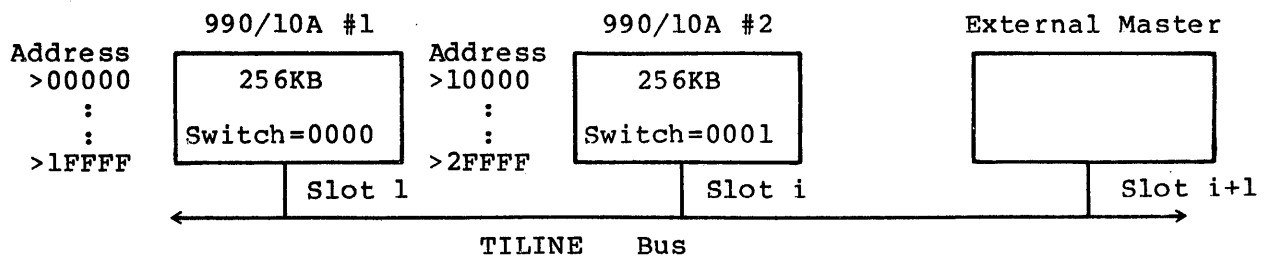
Figure A-2. Improperly Configured Multiprocessor Memory

```
        ┌──────────────────────────────┐
        │  Auxiliary  Processor    3   │
        └──────────────┬───────────────┘
                       │
        ┌──────────────┴───────────────┐
        │  Auxiliary  Processor    2   │
        └──────────────┬───────────────┘
                       │
        ┌──────────────┴───────────────┐
        │  Auxiliary  Processor    1   │
        └──────────────┬───────────────┘
                       │
        ┌──────────────┴───────────────┐
        │  Host  Processor             │
        └──────────────┬───────────────┘
                       │        Slot 1
        ├──────────────┴───────────────┤
```
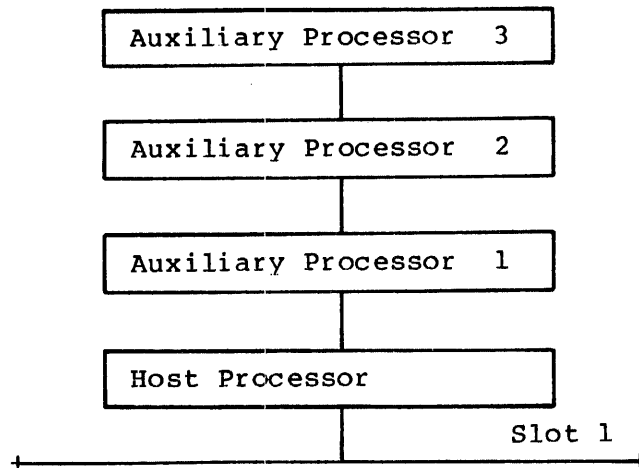
Figure A-3.   Simple Multiprocessor with a Single Host

```
┌───────────────────────────┐              ┌───────────────────────────┐
│  TILINE  Coupler  A       │◄────────────►│  TILINE  Coupler  B       │
└─────────────┬─────────────┘              └─────────────┬─────────────┘
              │                                          │
┌─────────────┴─────────────┐              ┌─────────────┴─────────────┐
│  Auxiliary  Processor  2  │              │  Auxiliary  Processor  3  │
└─────────────┬─────────────┘              └─────────────┬─────────────┘
              │                                          │
┌─────────────┴─────────────┐              ┌─────────────┴─────────────┐
│  Auxiliary  Processor  1  │              │  Auxiliary  Processor  S1 │
└─────────────┬─────────────┘              └─────────────┬─────────────┘
              │                                          │
┌─────────────┴─────────────┐              ┌─────────────┴─────────────┐
│  Host  Processor          │              │  Slot  1  Auxiliary       │
└─────────────┬─────────────┘              └─────────────┬─────────────┘
              │           Slot 1                         │          Slot 1
├────────────┴────────────────┤          ├─────────────┴─────────────────┤
```
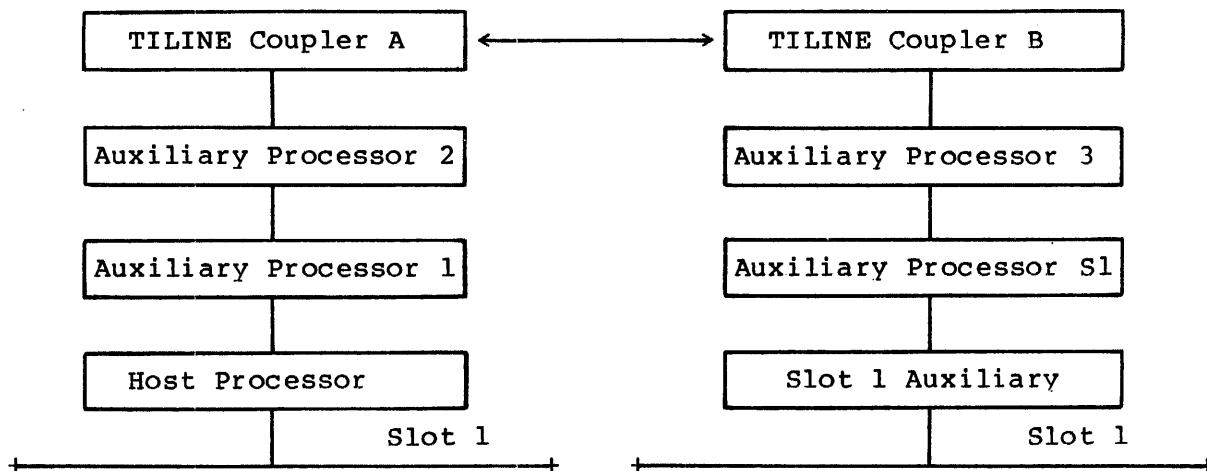
Figure A-4.   Simple Multiprocessor with a Single Host and a Subhost

A.4 TYPICAL SEQUENCE OF OPERATIONS

A typical sequence of operations for a simple multiprocessor system might be:

- The auxiliary processor powers up and executes self-test (except the load portion), extinguishes the FAULT light, and goes to IDLE.

- The host processor checks to see if the auxiliary processor passed self-test (FAULT is false), and checks for auxiliary IDLE to be true. To provide initial instructions to the auxiliary processor, the host may write a task into the auxiliary processor's memory for it to execute, or the auxiliary may be loaded with a kernel causing it to read an interprocessor message area. The host processor then sets the attention interrupt to the auxiliary processor and simultaneously enables the acknowledge interrupt back from the auxiliary processor.

- The auxiliary processor acknowledges that it received the attention interrupt (and possibly that it understood its instructions) by clearing the attention interrupt. The clearing of the attention interrupt sets the acknowledge interrupt.

- The host responds to the acknowledge by testing the acknowledge bit for a 1. If the host processor has multiple auxiliary processors on the same interrupt level and more than one is capable of generating the acknowledge interrupt at any one time, the host will need to test the acknowledge bit to determine which auxiliary processor has generated the interrupt. The host will then set the enable acknowledge bit to zero to clear the acknowledge bit. The acknowledge interrupt is cleared on the high-to-low transition of the enable acknowledge bit.

- The auxiliary processor, upon completion of its task or in need of service, will begin the same sequence just described. In this case, the words auxiliary and host need to be interchanged in the description above.

If the processor that initiates the sequence (sets attention to one) desires not to be interrupted in acknowledgement but prefers to poll for acknowledgement, the following sequence should be followed.

- The auxiliary processor powers up and executes self-test (except the load portion), extinguishes the FAULT light, and goes to IDLE.

- The host processor checks to see if the auxiliary processor passed self-test (FAULT is false), and checks for auxiliary IDLE to be true. To provide initial instructions to the auxiliary processor, the host may write a task into the auxiliary processor's memory or the auxiliary may be loaded with a kernel causing it to read an interprocessor message area. As a part of the initialization sequence, the host must enable the auxiliary's attention interrupt by setting the attention mask to one. The host processor then sets the attention interrupt to the auxiliary processor and simultaneously disables the acknowledge interrupt from the auxiliary processor. This keeps the acknowledge interrupt from being generated when the auxiliary clears the attention interrupt.

- The auxiliary processor acknowledges that it received the attention interrupt (and possibly that it understood its instructions) by setting the acknowledge interrupt. The setting of the acknowledge interrupt clears the attention interrupt. In this case, the acknowledge interrupt will not be generated because it is masked by the host.

- The host polls for the acknowledge by testing the acknowledge bit for a one. The host must set the enable acknowledge bit to one, and then to zero. This sequence is necessary to clear the acknowledge bit.

- The auxiliary processor, upon completion of its task or in need of service, will begin the same sequence just described. In this case, the words auxiliary and host need to be interchanged in the description above.

In *no* case should a processor write to its control word after setting its attention bit to one until it has seen acknowledge from the other processor or the other processor is verified to have halted (FAULT is true). If a processor were to attempt to modify a bit in its control word with the attention bit set to one, it could not correctly predict the value to be written into the attention bit. This is because the state of the attention bit can be changed at any time by the processor responding to the attention. A read followed by a write will not guarantee successful prediction because the responding processor may have its memory access interleaved with the desired read-write sequence.

The processor that sets the acknowledge bit should not clear the bit. Acknowledge will be cleared by the processor responding to the acknowledge. If the processor that sets the acknowledge also resets the acknowledge, the processor responding to the acknowledge may not be able to successfully test the acknowledge bit before it is reset and therefore may not be able to determine the source of the interrupt.

The examples above will suffice when messages are not queued across the interface. If queued requests are desired then the following sequence is required.

- The host writes a message in the interprocessor communication area and sets attention to the auxiliary. The auxiliary reads (copies) the message and responds with acknowledge. When the auxiliary is finished, it will respond with an attention to the host.

- The host wishes to put the next message in the auxiliary queue. It is possible that both the host and the auxiliary will get attention set on consecutive TILINE cycles and lock up the interface. To prevent this, the host disables interrupts (LIMI 2), sets the attention mask and then reenables interrupts. This is necessary because the auxiliary can set the interrupt the TILINE cycle before the mask was set. When the auxiliary responds with an acknowledge for the message just sent, the host can disable interrupts, enable attention, and reenable interrupts. When the auxiliary sees the attention from the host, it checks and sees that the host has masked off the attention from the auxiliary. Therefore, the host will not acknowledge the attention and the state of the attention will not change, allowing the auxiliary to write into its control word even though attention is set.

The queuing routine proposed above requires that interrupts be disabled for a short period of time. Another approach can be used to overcome the requirement of disabling interrupts. A lockword convention can be established, using a dedicated memory location. This location signals availability by showing a negative value. A processor desiring access tests the lockword, using the ABS instruction. The ABS instruction sets a positive value if the location contains a negative value. The status register indicates whether the value was changed or was already positive, indicating another processor had access. When access is complete, the processor sets the lockword negative. (A negate instruction is sufficient for this operation, since a conflict is not possible.) A processor finding the lockword positive postpones its access until a negative value is found.

A.5   SYSTEM CONFIGURATION CONSIDERATIONS

The system configuration for a multiprocessor system using the 990/10A as an auxiliary needs to be carefully considered. Although the software will impact the performance of the multiprocessor system more drastically than the system configuration, some considerations still need attention.

The host processor for a multiprocessor system can be a 990/10, 990/10A, or a 990/12. The use of either a 990/10 or a 990/12 as a host processor may degrade system performance because of the requirement of using the TILINE to execute instructions. The 990/10A with its on-board memory will not require as many TILINE accesses as either the 990/10 or the 990/12. Even though the host processor has the lowest priority use of the TILINE, any other master must wait for the current cycle to complete. By limiting the number of TILINE cycles, performance can be enhanced.

Alphabetical Index

Introduction

HOW TO USE INDEX

The index, table of contents, list of illustrations, and list of tables are used in conjunction to obtain the location of the desired subject. Once the subject or topic has been located in the index, use the appropriate paragraph number, figure number, or table number to obtain the corresponding page number from the table of contents, list of illustrations, or list of tables.

INDEX ENTRIES

The following index lists key words and concepts from the subject material of the manual together with the area(s) in the manual that supply major coverage of the listed concept. The numbers along the right side of the listing reference the following manual areas:

- Sections — Reference to Sections of the manual appear as "Sections x" with the symbol x representing any numeric quantity.

- Appendixes — Reference to Appendixes of the manual appear as "Appendix y" with the symbol y representing any capital letter.

- Paragraphs — Reference to paragraphs of the manual appear as a series of alphanumeric or numeric characters punctuated with decimal points. Only the first character of the string may be a letter; all subsequent characters are numbers. The first character refers to the section or appendix of the manual in which the paragraph may be found.

- Tables — References to tables in the manual are represented by the capital letter T followed immediately by another alphanumeric character (representing the section or appendix of the manual containing the table). The second character is followed by a dash (-) and a number.

  Tx-yy

- Figures — References to figures in the manual are represented by the capital letter F followed immediately by another alphanumeric character (representing the section or appendix of the manual containing the figure). The second character is followed by a dash (-) and a number.

  Fx-yy

- Other entries in the Index — References to other entries in the index preceded by the word "See" followed by the referenced entry.

USER'S RESPONSE SHEET

Manual Title: Model 990/10A Computer, General Description (2302633-9701)

Manual Date: 1 September 1982   **Date of This Letter:** _____

User's Name: _____   **Telephone:** _____

Company: _____   **Office/Department:** _____

Street Address: _____

City/State/Zip Code: _____

Please list any discrepancy found in this manual by page, paragraph, figure, or table number in the following space. If there are any other suggestions that you wish to make, feel free to include them. Thank you.

Location in Manual   **Comment/Suggestion**

CUT ALONG LINE