VARIAN 70 SERIES

OPTION BOARD

Specifications are subject to change without notice. Address comments regarding this document to Varian Data Machines, Publications Department, 2722 Michelson Drive, Irvine, California, 92664.

varian data machines / a varian subsidiary
2722 michelson drive / p.o. box e / irvine / california / 92664

TABLE OF CONTENTS

SECTION 1 GENERAL DESCRIPTION

SECTION 2 INSTALLATION

SECTION 3 OPERATION 3-1

SECTION 4 THEORY OF OPERATION

[redacted]

LIST OF TABLES

[redacted]

SECTION 1

GENERAL DESCRIPTION

This manual contains the specifications, installation information, mnemonics lists, maintenance data, and theory of operation for the option board. Depending on the system configuration, the option board contains some or all of the following features:

- Common logic
- I/O Control logic
- Teletype controller (TC)
- Keyboard/display terminal controller (CRT)
- Power failure/restart (PF/R)
- Real-time clock (RTC)
- Memory protection (MP)
- Memory parity logic
- Priority memory access (PMA) (see note)

Note: PMA cannot be added in the field.

1.1 FUNCTIONAL DESCRIPTION

The **Varian 70 Series option board** contains mainframe features for the *Varian 70 Series Computer Systems.* The board is available in seven standard configurations:

- I/O control logic
- Common logic, I/O control logic, RTC, TC, PF/R
- Common logic, I/O control logic, RTC, TC, PF/R, MP, and Memory parity logic*
- Common logic, I/O control logic, RTC, TC, PF/R, MP, Memory parity logic*, and PMA
- Common logic, I/O control logic, RTC, CRT, PF/R, Memory parity logic*, and PMA
- Common logic, I/O control logic, RTC, CRT, PF/R, MP, and Memory parity logic*
- Common logic, I/O control logic, RTC, CRT, PF/R, MP, Memory parity logic*, and PMA

Figure 1-1 is a block diagram of the option board and its interfaces. Communication between the option board and the processor board is through the common logic. Buffer repowering in the common logic reduces the load on the processor I/O bus and also reduces the numbers of integrated circuits (IC) required for the options. The processor I/O control section, which is located on the option board, is discussed in the Varian 70 Series Processor Manual (98 A 9906 02x).

1.1.1 Teletype Controller

The **Teletype controller (TC)** is a control and interface device for data transfers between the processor board and a Model 33 or 35 Teletype. Data is transferred between the processor board and the TC in 8-bit bytes over the I/O bus under either interrupt or sense control. Data is transferred between the TC and the Teletype over a serial, asynchronous, full duplex interface.

Note: The keyboard/display terminal (CRT) uses the same controller (with minor modification) as the Teletype. Throughout this manual, all references to Teletype (TTY) also pertain to CRT.

1.1.2 Power Failure/Restart

The **power failure/restart (PF/R)** protects, during loss or reduction of ac prime power, the program in progress and the contents of computer memory and registers. Upon restoration of power, the PF/R automatically restarts the computer and causes it to reenter the interrupted program at the point of interruption.

Power reduction, failure, or turn-off initiates a power-down cycle during which the PF/R completes execution of the current instruction and then interrupts the processor, directing it to the address of the SAVE subroutine. This service subroutine loads the contents of the volatile registers (A, B, X, P, and overflow) into preselected addresses in memory. After the execution of SAVE, the PF/R disables the processor and memory until power is restored.

When power is restored so that all power-up conditions are satisfied, the PF/R enables the processor and memory, initiates the system-start signal, and directs the processor to the address of the RESTORE subroutine. This service subroutine reloads the registers with the saved data, and contains a jump instruction that directs the processor to reenter the program at the point of interruption and continue execution.

1.1.3 Real-Time Clock

The **real-time clock (RTC)** provides the following real-time functions:

- Variable-interval interrupt
- Memory-overflow interrupt
- Readable free-running counter

* The memory parity logic is functional only when the entire system memory is comprised of optional parity core and/or parity semiconductor memory modules (18 bit).

[redacted]

1.1.4 Memory Protection

Memory protection (**MP**) prevents unauthorized or unintentional program access to, or modification of, protected areas of memory.

Memory is divided into 512-word segments, each of which can be either protected or unprotected. Segments not designated as protected are, by definition, unprotected. The protected/unprotected status of each segment is stored in the MP in four 16-bit mask registers that are loaded by I/O instructions from the processor. These mask registers can store the status of up to 64 memory segments.

The MP monitors the address of the instruction being processed, the address specified by the next instruction in sequence, and the address specified by the effective address. Using this information and the status of the stored segment, the MP determines and operates on errors.

When a program is executed in unprotected memory, any of the following operations constitutes an error:

- Overflowing into a protected segment
- Writing into a protected segment
- Jumping into a protected segment
- Executing an I/O instruction in an unprotected segment
- Executing a halt instruction in an unprotected segment

When the MP detects an error, execution of the current instruction is completed. However, if the instruction specifies writing or I/O operations, the contents of the A, B, and X registers and memory are not modified. Processor program processing is then interrupted and directed to one of five preassigned memory addresses. From this address, the program is directed to a user-written service subroutine for error analysis and correction.

NOTE: By definition, an error cannot exist during the execution of a protected instruction even when it indirectly references an unprotected address.

When the memory map option is installed, memory protection is disabled.

1.1.5 Memory Parity Logic

The **memory parity logic** generates and checks parity for all memory access. The parity check/generator generates parity on memory writing operations, and checks it on reading operations. The parity control generates an interrupt request when a parity error is detected. The parity logic is functional only when the entire system memory is comprised of optional parity core memory and/or parity semiconductor memory modules (18 bit).

1.1.6 Priority Memory Access

The **priority memory access** (**PMA**) is the interface (figure 1-2) between memory and the four data-transfer channels. The channels have hardware-fixed priorities. All signals transmitted on these channels are controlled by an interlock scheme that makes the interface independent of circuit and cable speed. This permits processing of data using a wide variety of transfer rates and circuit speeds.

The PMA continuously scans the four channels and generates acknowledgments as required. The order of priority for memory access on the B port is:

- PMA
- Direct memory access (DMA)
- 620 compatible direct memory access
- Processor

In dual port systems in which the processor is configured on port A, requests to separate memory modules will cause both requests to be serviced at the same time (figure 1-3).

After receiving an acknowledgment, the PMA controller clears the request unless consecutive memory cycles are required, in which case the line remains set. The controller then enables the 20-bit address bus, gates the data onto the 16/18-bit bidirectional bus, and, if a reading operation was requested, enables the read line. When the buses are stable, the data transfer begins. When the PMA detects the memory-data-ready signal from memory, it gates the data to the PMA bus and clears the acknowledgment. For reading operations, the controller uses the trailing edge of the acknowledgment signal to clock the data.

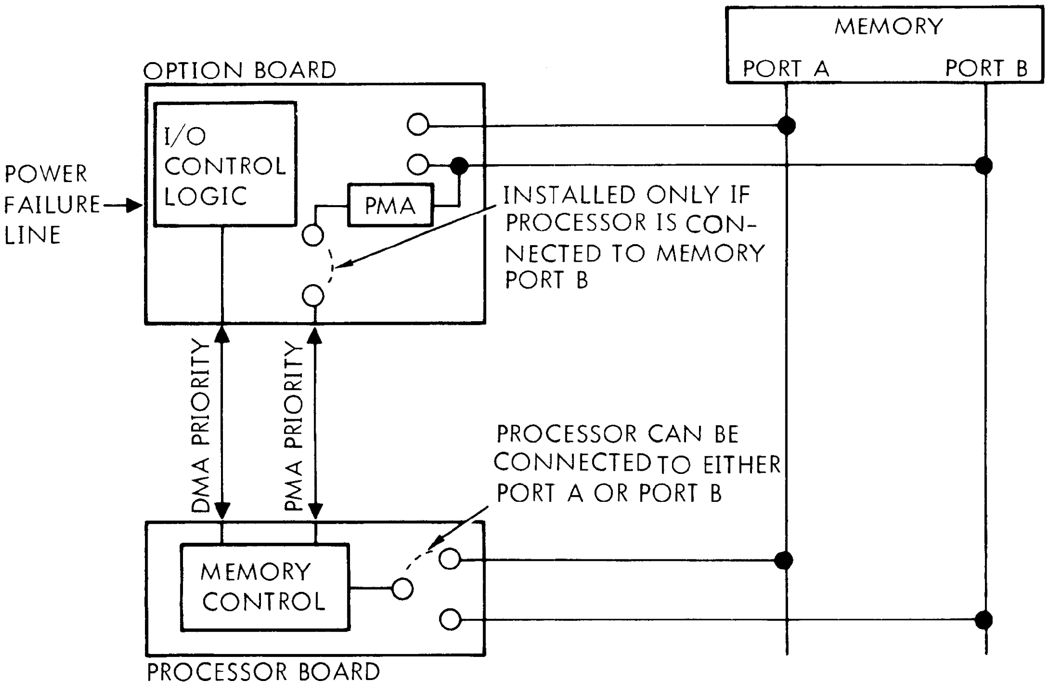
Up to eight controllers can share the PMA option. Where several controllers are connected to a given channel, only one can be active at a time.

The memory access logic scans the PMA and DMA request lines at the main system clock rate. If the lines are not enabled, the processor controls all memory cycles. If the PMA, DMA, or 620 compatible DMA request lines are enabled, the processor is locked out of memory after a request is recognized, and no other unit has access to memory as long as a PMA request is enabled. PMA requests thus take priority over those from the DMA. PMA requests are, however, ignored in case of power failure.

1.2 SPECIFICATIONS

Tables 1-1 through 1-7 contain the physical, electrical, and operational specifications for the option board and each of the options.

[redacted]



Note: The processor, parity, and PMA are all connected to port B on a V72 system.

VT11-1562A

Figure 1-3. Memory Access

Table 1-1. Option Board Specifications

Parameter	Description
Organization	Contains circuits for the common logic, I/O control logic, TC, PF/R, RTC, MP, memory parity logic, and PMA on a 15.6-by-19-inch (39.6-by-48.3-cm) circuit board
Logic Levels	High: +2.4 to +5.5V dc Low: 0.0 to +0.5V dc
Interconnection	Plugs into the backplane of the computer mainframe, interface signals are routed via card-edge connectors, PMA bus and TC signals pass through separate connectors at the rear of the board
Power requirements	-12V dc at 10 milliamperes 24V ac at 10 milliamperes +5V dc at 12 amperes
Operational Environment	0 to 50 degrees C; 0 to 90 percent relative humidity without condensation

Table 1-4. RTC Specifications

Parameter	Description
Modes of Operation	Variable-interval interrupt, memory overflow interrupt, interval accumulation, time-of-day accumulation, and event accumulation
FRC Capability	Counts up to 0177777 with three hardware-selectable timing sources
I/O Capability	Seven external control and two data transfer instructions
Timing Sources	10 kHz ± 1 percent, squarewave; variable-interval interrupt range: 100 microseconds to 409.5 milliseconds (in 100-microsecond increments) Line frequency, derived from 24V rms sinewave; variable-interval interrupt range (nominal): 16.7 milliseconds to 68.3 seconds (in 16.7-millisecond increments) at 60 Hz, and 20.0 milliseconds to 81.9 seconds (in 20.0-millisecond increments) at 50 Hz External timing (user-supplied): minimum positive duration, 5.0 microseconds; minimum negative duration, 5.0 microseconds
Interrupt Priority	Determined by position on the priority chain

Table 1-5. MP Specifications

Parameter	Description
Protection capacity	Up to 64 segments (512-word) of memory
Modes of Operation	Detects halt, overflow, I/O, writing, and jump errors
Interrupt Priority	Highest on the I/O bus priority chain
I/O Capability	Six external-control and eight data-transfer

Table 1-6. PARITY Specifications

Parameter	Description
Mode of Operation	Interrupt logic issues interrupt request upon detection of a parity error
Interrupt Priority	Follows PF/R and MP on the I/O bus
I/O	I/O interface via the interval I/O bus logic
Bus	Generates and checks parity on either the A or B bus, but not both. The hardware-selectable bus is the same as that used by the processor.

[redacted]

SECTION 2
INSTALLATION

This section describes the steps to be taken during the unpacking and installation of the option board, gives a physical description of the board, and lists the pin assignments.

2.1 INSPECTION

The option board has been inspected and packed to ensure its arrival in good working order. To prevent damaging the board, take reasonable care during unpacking and handling. Check the shipping list to ensure that all equipment has been received. Immediately after unpacking, inspect the equipment for shipping damage. If any is found:

- Notify the transportation company
- Notify Varian Data Machines
- Save all packing material

2.2 PHYSICAL DESCRIPTION

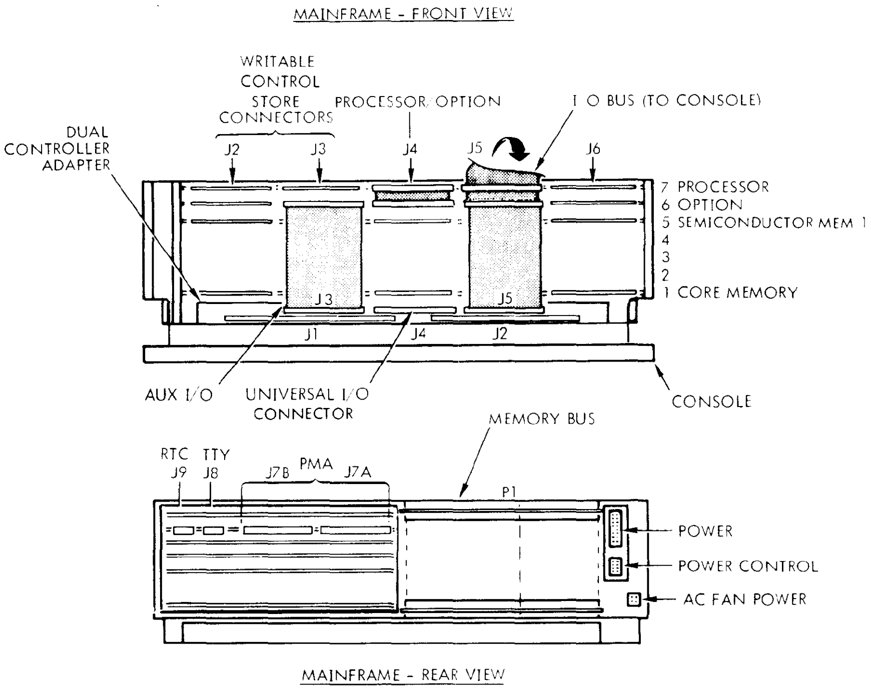
The option board is a 15.6-by-19-inch (39.6-by-48.3-cm) printed circuit (PC) board that mounts in any mainframe chassis slot (figure 2-1).

The integrated circuits (ICs) on the option board are dual-in-line, plastic-encased chips. Under certain circumstances, some ceramic ICs are used.

2.3 INTERCONNECTION

The option board interfaces with other system components through card-edge connectors plus connectors mounted at the rear of the board. These connectors carry all signals indicated by the block diagram (figure 1-1) as interfacing with the rest of the system. TC and PMA-bus signals pass through the connectors at the rear of the board to special cables leading to the Teletype and PMA controller (respectively). The memory bus provides the interface between memory and the option board (MP, PARITY, and PMA). Except for the PMA, which does not use the I/O bus, all I/O signals between the CPU and the option board are repowered in the option board common logic and distributed via the option bus within the board.

The option-board pin assignments are given on the logic diagram in volume 2. The signal mnemonics are defined in section 6.



VTII-2083A

Figure 2-1. Typical Option Board Installation

[redacted]

SECTION 3
OPERATION

The option board has no operating controls or indicators. Computer operations that affect the option board are explained in the Varian 70 Series Processor Manual (98 A 9906 02x).

[redacted]

SECTION 4

THEORY OF OPERATION

As shown in the block diagram (figure 1-1) all option board features (except PMA, if present) are interconnected by the option bus. This bus is the interface between the individual sections and the option board common logic. The common logic in turn is the interface between the option board and the bidirectional I/O bus to and from the rest of the system.

The PMA has its own PMA bus. The three options that deal directly with memory (MP, PARITY, and PMA) are also connected to the memory bus.

The option board common logic (logic diagram, volume 2; mnemonics, section 6) consists of buffered system control lines (IUJX—I to OIUJX +, DRYX—I to ODRYX +, FRYX—I to OFRYX +, SYRT—I to OSYRT +, IUAX—I to OIUAX +, IUCX—I to OIUCX +, and OIURX— via OIURX + to IURX—I) and a set of mutually exclusive drivers and gates through either transfer data from the option bus (OB00— through OB15—) to the bidirectional I/O bus (EB00—I through EB15—I) or vice versa. Thus, for a data-transfer-in (timing figure 4-1), receivers are disabled by OBOUTA— and OBOUTB— while the drivers are enabled by OBOUT + and OBOUTB +. On data-transfer-out, the reverse is true. LK82+ creates the 12.1212 MHz internal option-board clock.

4.1 TELETYPE CONTROLLER

The **Teletype controller (TC)** (figure 4-2) consists of the following functional blocks: instruction decoding and control logic, MOS transmitter/receiver, TC clock, data-transfer-out buffer register, and the TTY interface logic.

The *instruction decoding and control logic* responds to processor data-transmission requests and supplies interrupt signals for priority interrupt module (PIM) control. The TC decodes the device address (01) from the low-order six bits of the option bus (OB00— through OB05—) producing TDA+ after ANDing with not interrupt acknowledge (OIAUX—). If an address is on the option bus, it is recognized when OIUAX+ is false. The I/O-TC interface provides for TC initialization and storage for processor-TC data transfers in a two-word format. The control logic generates signals to control other TC logic blocks, including the MOS transmitter-receiver. It also provides logic for external control instructions and initialization.

The *MOS transmitter/receiver* is a single 40-pin chip. It monitors all data transfers between the TC and the TTY, including parallel-to-serial data conversion and transmission to the TTY, serial-to-parallel data conversion for transfer from the TTY to the processor, and data flags to the instruction decoding and control logic.

The *TC clock* supplies timing signals to the MOS transmitter/receiver and the control logic by counting-down a 24.2424 MHz signal from the processor internal crystal-controlled oscillator. The counter can be programmed using jumpers to vary the transfer rate (figure 4-3).

The *data-transfer-out buffer register* provides the necessary buffering for data transmission.

The *TTY interface logic* provides a three-line interface from the TC to the TTY. The interface lines are isolated from the TC by optical couplers to control possible high voltages in the TTY interface (up to 140V ac during power surges).

4.1.1 Signal Interfaces

The TC interfaces with the basic computer and a factory-modified TTY. No other options are required; however, the TC can interface with a PIM to provide character-ready interrupts.

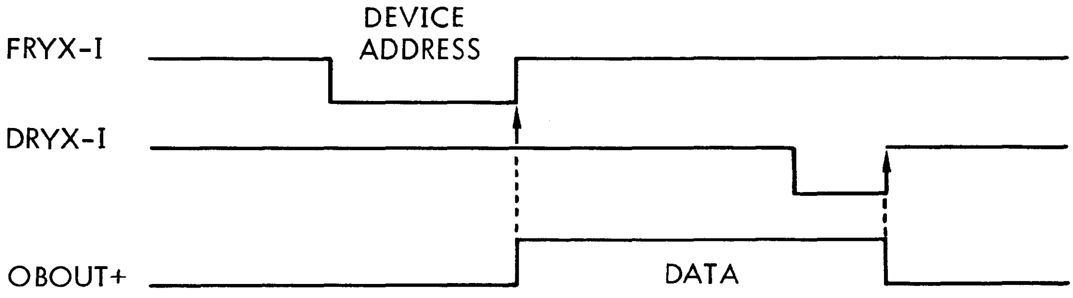


Figure 4-1. Common-Logic Timing for Data-Transfer-In

[redacted]

BAUD RATE	E42 TO E43	E40 TO E41	E38 TO E39	E36 TO E37	E34 TO E35	E32 TO E33	E30 TO E31	E28 TO E29	E26 TO E27	E24 TO E25	E22 TO E23	E20 TO E21
110 bps *	X	X	X	X		X	X	X		X		X
150 bps	X		X	X	X	X		X	X	X	X	
300 bps		X	X		X	X	X	X				
1200 bps			X	X			X	X	X	X		X

X = INSTALL JUMPER
* = 110bps IS STANDARD

VT11-1565A

Figure 4-3. TC Clock Jumpers for Various Transfer Rates

4.1.1.1 Teletype Interface

The TC interfaces a model 33 or 35 TTY via the option board TTY connector. The TTYs are modified at the factory prior to delivery to the customer. To modify the 33 ASR TTY:

a. Set the TTY for 20 mA operation. This includes the addition of a wire that enables the TTY to supply "battery" for the send and receive data loops to the clock card (the TTY is the current source for the TC optical couplers).

b. Set the TTY for full-duplex operation.

c. Disable the WRU contacts.

d. Disable parity on the keyboard.

e. Modify the answer-back drum.

f. Install the 180801 function lever.

The model 35 is similarly modified. Model 33 and 35 TTYs are electrically interfaced and cabled to the TC in almost the same manner, even though they are physically quite different in appearance and in their internal operation.

The cable used for the TC/TTY interface for the 33 ASR runs from S connector plug P2 in the TTY to option board TC connector J8 at the rear of the CPU. The S connector is located at the right rear, top row second connector from the right. The TTY cable is normally 20 feet long with nine leads, three of which are used with standard Teletypes.

The TTY end of the cable (P2) includes two other wires. Pins 7, 4, and 5 are interconnected. These connections tie together internal TTY leads brought into the S connector plug as part of the wiring. Note that both ends of the cable are keyed to ensure proper mating.

NOTE

The TTY cable is normally installed at the TTY by Varian Data Machines before customer delivery. The model 33 TTY requires about 3 amperes of ac power, and the model 35 TTY about 6 amperes.

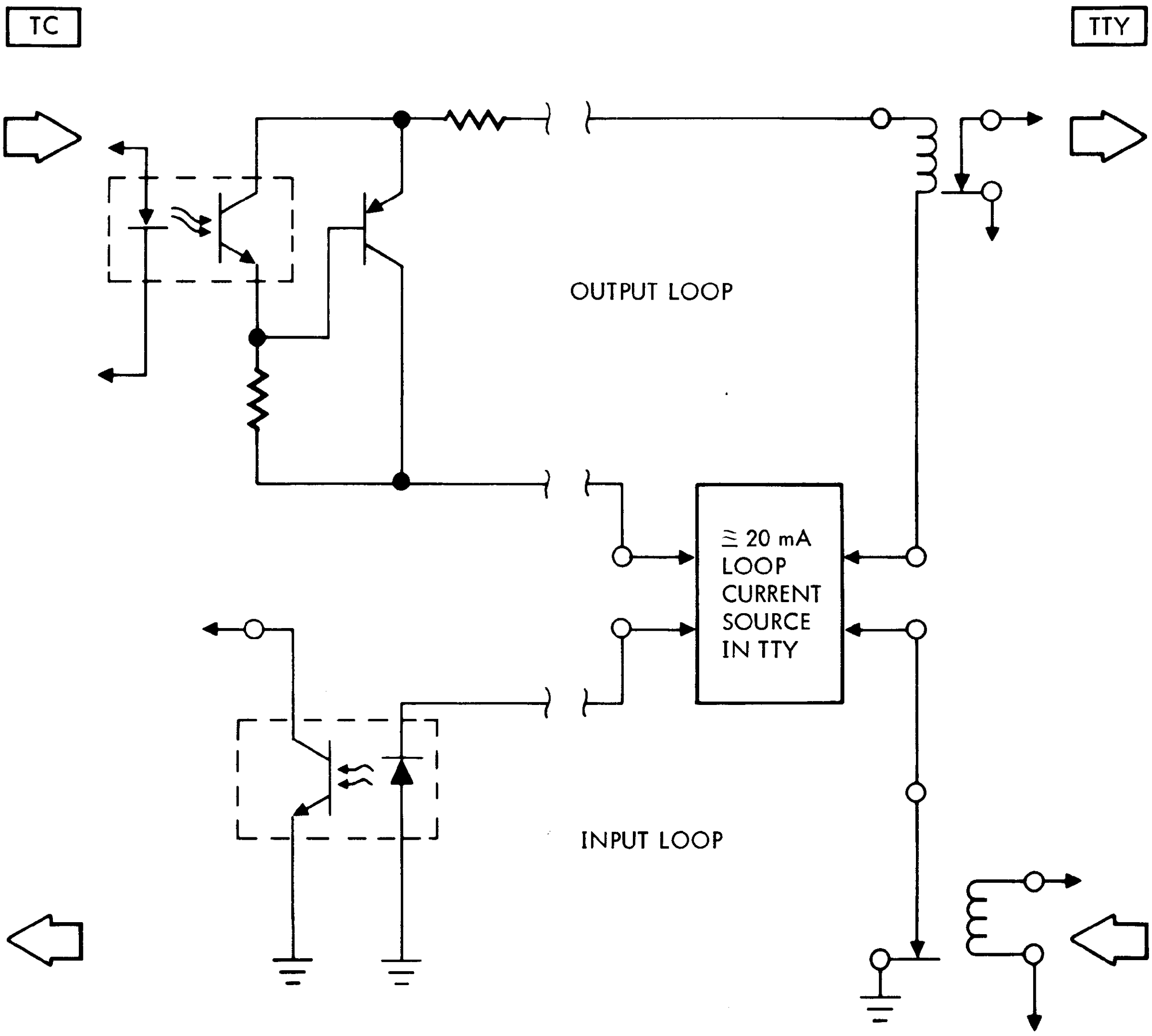
TTY DESIGN: The 33 ASR is primarily designed for light to medium use. Normally, it is the basic computer input/output device and is the most widely used unit. Its full-duplex operating mode allows simultaneous input and output.

The 35 ASR performs the same function as the 33 ASR, but it is designed for heavy sustained use.

The 35 KSR is used for keyboard send/receive only and lacks the paper tape punch (PTP) and paper tape reader (PTR) capability of the ASR models. The operating characteristics are similar to model 33 keyboard operation. This unit is also designed for heavy, sustained use.

TTY INPUT METHODS: TTY input can be via the keyboard or the PTR. At the keyboard, the operator types at a random rate not greater than 10 characters per second (cps), the maximum rate for TTY input. Standard eight-level paper tapes are read by the PTR at a rate of 10 cps.

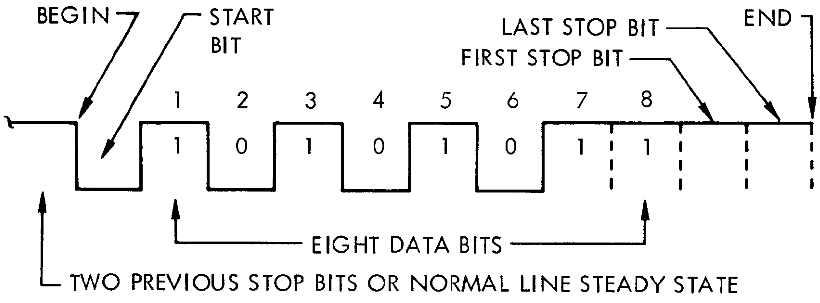
[redacted]



NOTE: Equivalent, not actual, circuit with two independent current loops for full-duplex operation. The current loops may have a single common current source and return wire. Isolation is achieved by optical couplers.

VTII-1567

Figure 4-4. Optical Coupler Isolation for TC/TTY Interface



VTII-1568

Figure 4-5. Typical TTY Character

[redacted]

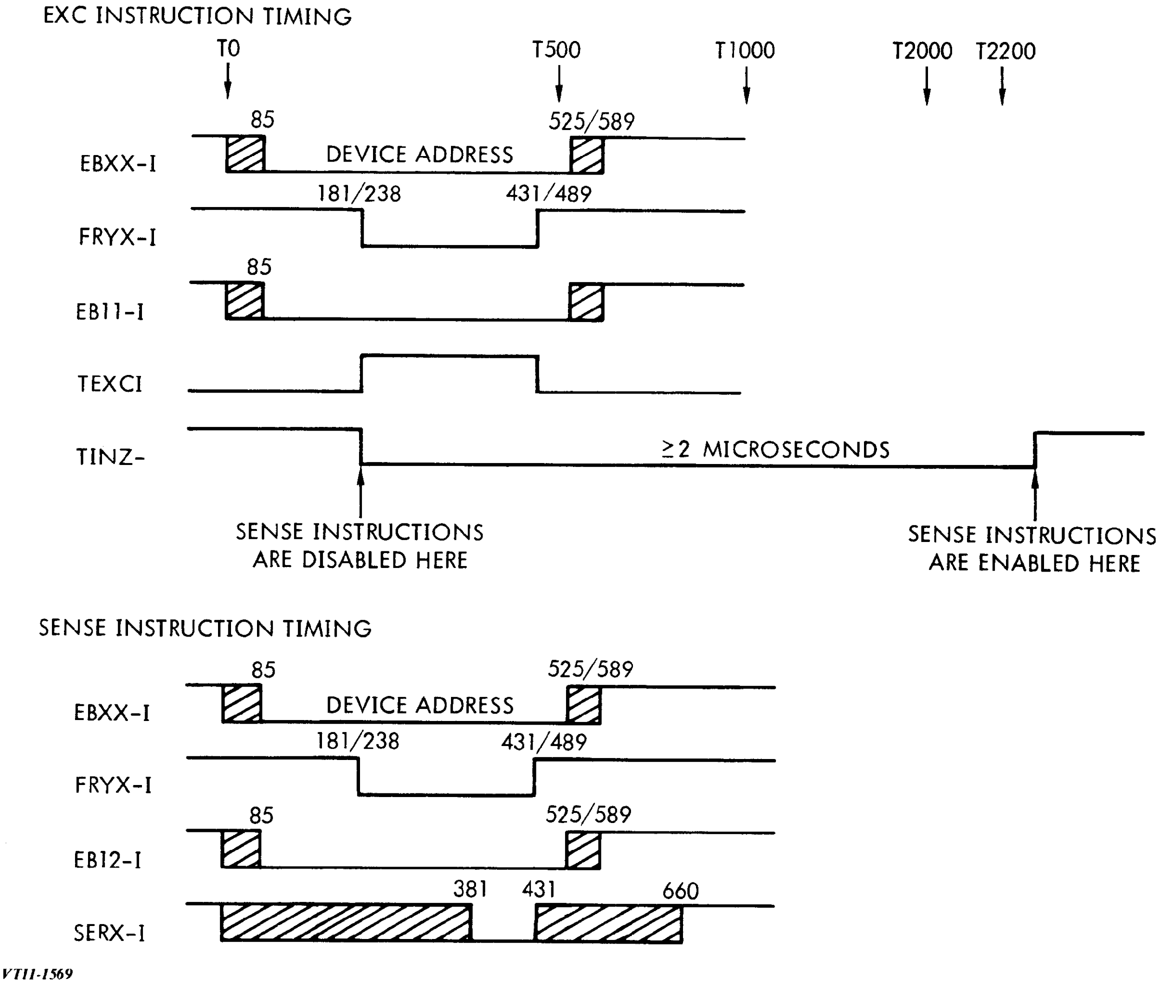


Figure 4-8. TC Control Timing

transmits a steady mark to the TTY by keeping the transmitting optical coupler active. The TC can also be initialized under program control but the instruction cannot be issued while the TC is in communication with the TTY. Figure 4-8 shows the software initialization timing.

NOTE

An initialization instruction performs the same function as pressing the RESET switch.

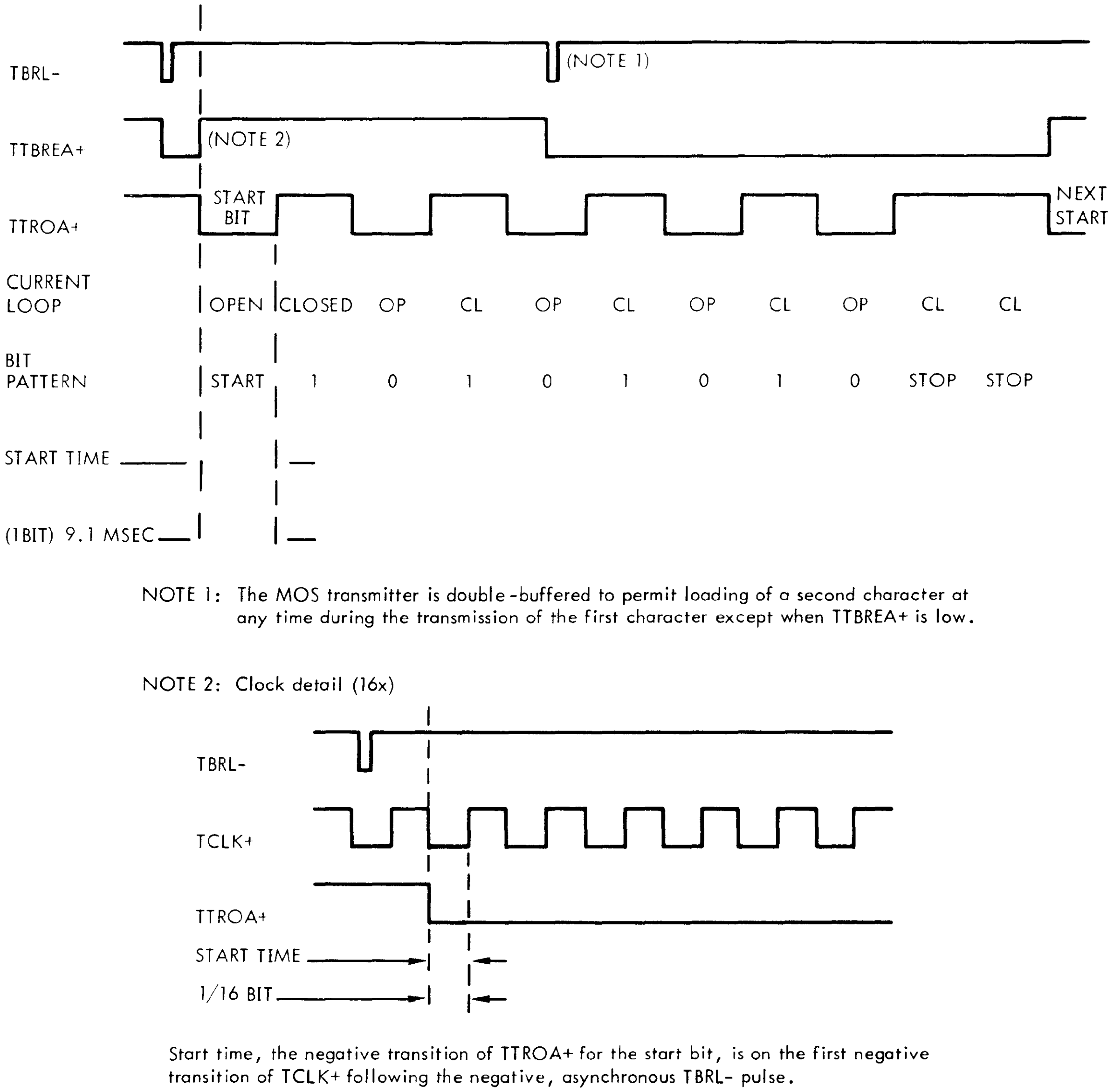
4.1.2.2 TC/Common-Logic Interface

The processor controls the TC with external control and data transfer instructions via the option bus. When such an instruction is issued, the following device sequence occurs: TDA+ is high if the proper address (01) is put on the option bus and OIUAX+ is low (no interrupt or DMA). TDA+ high enables the instruction-generation gates.

The functions of the option bus signals are: OB00– through OB05– enable the device address signal TDA+. OB06– enables the output-ready sense response TTS+. OB07– enables the input-ready sense response TRS+. In addition, the bidirectional signals OB00– through OB07– enable the data-transfer-out buffer register. OB11– enables the software reset one-shot, OB12– enables sense response SERX–I, OB13– enables the input sequence, and OB14– enables the output sequence.

OIUAX+ inhibits TC selection when the processor has an interrupt or direct memory access (DMA) in process. The processor can issue any of several instructions. All of the following instructions issue a device address and OFRYX+ (FRYX–I) and several issue ODRYX+ (DRYX–I).

[redacted]



NOTE 1: The MOS transmitter is double-buffered to permit loading of a second character at any time during the transmission of the first character except when TTBREA+ is low.

NOTE 2: Clock detail (16x)

Start time, the negative transition of TTROA+ for the start bit, is on the first negative transition of TCLK+ following the negative, asynchronous TBRL- pulse.

VTII-1571

Figure 4-10. TC Data-Transfer-Out Timing

mitter/receiver, and portions of the TC/TTY interface are used. Circuit elements are control flip-flops, a group of gates enabled by the option-bus signals, a programmable free-running clock that is synchronized with the processor, the transmitting half of the MOS transmitter/receiver, and an output coupler to the TTY with associated drivers. The data and control output waveforms illustrate the loading of the output from the processor to the TC and thence to the device (figures 4-9 and 4-10). For these waveforms, assume that the TC has been initialized, the transfer rate is 110 bps, the data is in 11-bit format, the processor has sensed write ready and has issued an output instruction, and the output character is 10101010 binary. Data are output from the processor in a two-word format: the first contains the device address (01) and tee function, and the second is character data. The device address is sampled on the trailing edge of FRYX—I, and data are sampled on the trailing edge of DRYX—I.

[redacted]

tialized, the device has just begun to transmit a character, and the input optical coupler is deactivated. Character reading is initiated by the negative transition of TRI– signaling the start-bit. From this transition to the positive transition of the data-ready signal TDR + all operations are under control of the MOS transmitter. The positive transition of TDR + indicates that the character has been received and loaded into the MOS device. About 500 nonoseconds later the data appear at the MOS device outputs TRR00 + through TRR07 + . *The processor has approximately one "character time" to execute a data-transfer-in instruction before data are lost.* During the execution of this instruction, the common-logic drivers are enabled by OBOUT– low. TDRR– clears the TDR flip-flop within the MOS device, and prepares the next character. Data are transferred to the processor on the trailing edge of DRYX– I.

4.1.3 Programming

The TC can be programmed directly under processor control (with or without interrupts).

The TC can supply interrupts (input ready and output ready) to the PIM when this option is included in the computer system, thus saving computing time and simplifying software since programmed delay and sense loops can be avoided. With this feature, the program running in the processor is interrupted at the proper time.

The software initializes the TC (as does pressing the RESET switch on the control panel). The TC is then ready to accept output from the processor and is also capable of accepting an input character from the TTY. The software normally issues a sense instruction and, if a sense-ready condition exists, follows it with a reading (input) or loading (output) instruction to enable the data transfer of one character between the TC and the processor.

Except for interrupts, the TC operates, for TTY operations, within the following general timing restrictions.

> **Output:** Maximum data transfer rate is 10 characters per second. There is no minimum rate. The processor can output a single character, discontinue output for an indefinite period of time (longer than 100 milliseconds), and then output another character without loss of data or synchronization.
>
> **Input:** Maximum data transfer rate is 10 characters per second (100 milliseconds per character). The processor must read the input character transferred from the TTY by the TC during the last (second) 9.1-millisecond stop-bit period. If the processor fails to read the character input during this time and before the TTY inputs again, the character is lost.

The TC and the TTY respond to the instructions listed in table 4-1 and the standard TTY codes in table 4-2. The initializing instruction performs the same function as the RESET switch on the computer console. The TC is prepared to accept processor output and to monitor TTY input. This instruction can not be issued while the TC is communicating with the device.

The sensing instructions are the sense ready-to-read and the sense ready-to-write instructions, which enable the processor to determine TC status. If the sense condition is met, a data transfer can proceed. If the sense condition is not met, the processor must wait to perform data transfer. A sense instruction can be issued at any time and normally precedes any data transfer instruction.

The data transfer instructions are, the read (input) and load (output) instructions, which transfer data between the processor and the TC through the read and write registers. Issuing a reading or loading instruction at the wrong time results in incorrect data transfer.

Table 4-1. TC and TTY Instructions

Mnemonic	Octal Code	Functional Description
	External Control	
EXC 0401	100401	Initialize
	Transfer	
OAR 01	103101	Transfer the A register to the transmit buffer register
OBR 01	103201	Transfer the B register to the transmit buffer register
OME 01	103001	Transfer the memory address to the transmit buffer register

(continued)

Table 4-2. ASCII TTY Codes *(continued)*

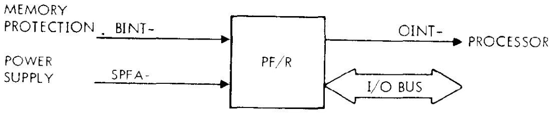
Teletype Symbol	ASCII Code	Teletype Symbol	ASCII Code
S	323	+	253
T	324	,	254
U	325	-	255
V	326	.	256
W	327	/	257

4.2 POWER FAILURE/RESTART

The operation of the **power failure/restart (PF/R)** (figure 4–13) includes:

- PF/R - power supply interaction
- PF/R - processor interaction
- PF/R internal functions

NOTE: The PF/R is operative only when the computer is in RUN mode. If the computer is in STEP mode when power is lost, the PF/R interrupt is not acknowledged and the contents of the volatile registers are lost.



VTII-1574

Figure 4-13. PF/R Functional Block Diagram

In the **PF/R-power supply interaction**, the computer power supply generates a low power-failure signal at the start of a power-up sequence. When all dc voltages are within their operating ranges and the line voltage is 105V ac or higher, the power-failure signal goes high. This positive transition triggers the PF/R power-up sequence.

When power is lost or below minimum requirements for computer operation, the power failure signal goes low. The PF/R then generates an interrupt request, causing the processor to execute the SAVE subroutine if the computer is in run mode. In halt mode, the processor does not acknowledge the request. In this case, the memory and processor are immediately disabled, and the contents of the volatile registers are lost.

During a power loss, the PF/R interrupt request is not acknowledged until the instruction in process completes execution. If the instruction requires multilevel addressing, this acknowledgment can be delayed up to five memory cycles, depending on the number of addressing levels (one machine cycle per level). DMA memory activity can also delay the interrupt processing up to one DMA cycle.

SAVE includes operations such as storing the contents of the volatile registers so that the interrupted program can complete execution when power is restored.

The power supply causes the system-reset signal to remain low from turn-on to 100 microseconds before the positive transition of the power-failure signal, thus disabling the processor. When the power-failure signal goes high, system reset is removed and the interrupt clock is enabled. Approximately 900 nanoseconds later, the PF/R simultaneously places the processor in run mode and requests an interrupt. The processor acknowledges the request by jumping to address 042 as specified by the PF/R. The processor then executes the PF/R RESTORE subroutine, during which all other interrupts are disabled. At the completion of RESTORE, the processor jumps back (under program control) to the program being executed when power was lost and reenables all other interrupts.

During a power-down sequence, the power-failure signal remains low as long as power failure conditions exist. Following the negative transition of the power-failure signal, the PF/R interrupt request logic is enabled. The processor then, after executing the current instruction, acknowledges the request by jumping to interrupt address 040 and executing the PF/R SAVE subroutine, during which all other interrupts except memory protection are disabled.

Approximately one millisecond after SPFA– goes low, the power supply generates SRST–, which disables the processor and memory.

The usual SAVE and RESTORE subroutines use a halt/run flag to indicate whether a power failure occurred while the processor was in HALT mode. If so, the flag is unchanged, since SAVE was not executed. Upon restoration of power, RESTORE examines the flag to determine whether the previous data can be restored in the volatile registers or a HALT is required.

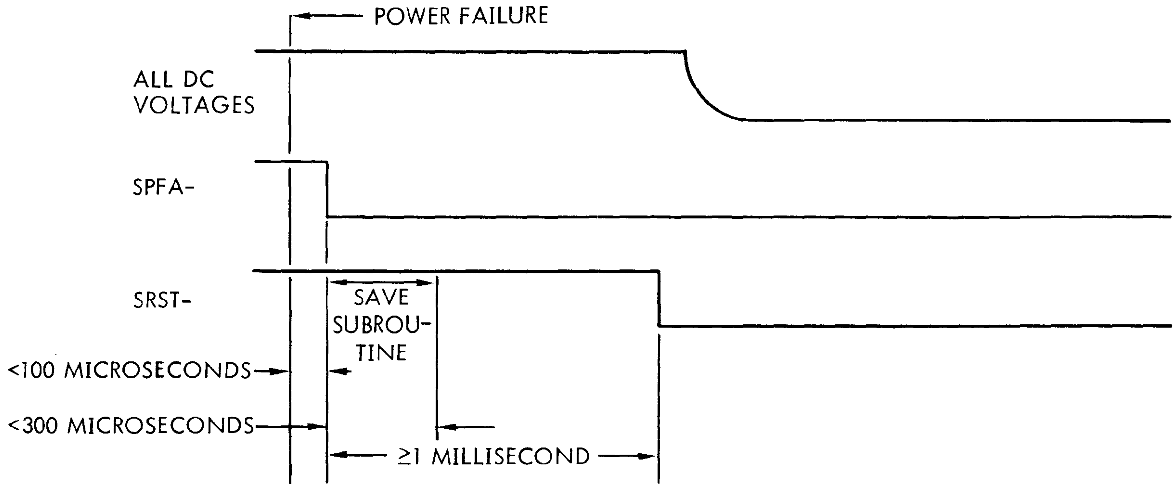
The **PF/R internal logic** consists of three functional blocks: sequencer, system start, and interrupt request.

The *sequencer* logic sends control signals to the other PF/R logic sections to generate the required output signals.

The *system start* logic produces a short pulse for starting the system when state 01 is true.

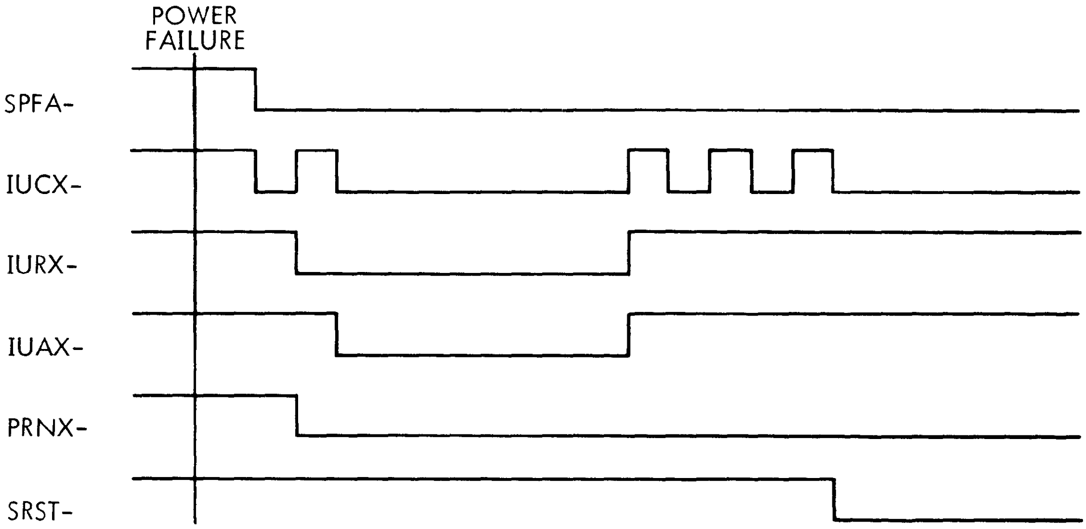
The *interrupt request* logic generates signals to request an interrupt cycle and to define the memory address of the interrupt. The interrupt priority logic generates a signal that disables lower-priority options and peripheral controllers when the PF/R is executing a power-up RESTORE subroutine or a power-down SAVE subroutine.

[redacted]



VTII-1577

Figure 4-16. Power-Down Timing at the Power Supply-PF/R Interface



VTII-1579

Figure 4-17. Power-Down Timing at the PF/R-Processor Interface

[redacted]

voltage is 105V ac or greater. A **power-down** condition exists when one or more of the power-up requirements is not fulfilled.

Prior to the beginning of a **power-up sequence**, SPFA- and SRST- are low. The sequence begins when the power supply forces SRST- high. Approximately 100 microseconds later, SPFA- goes high. The initializing flip-flop set-output FCX1 + goes high on the clock transition with the control logic going to state 01. This sets the one-shot and generates system-start pulses OPSTRT- and OINT- . Thus, the positive transition of SPFA- causes the PF/R to enable the interrupts and the CPU to begin execution of the resetore subroutine within 500 to 1000 nanoseconds.

The interrupt request to the CPU for the RESTORE subroutine is generated when the control flip-flop is reset (FC1X- high) and control state 01 signal FCS1- goes low, enabling, when BINT- is high, OIURX- low.

After a request, the CPU activates the interrupt-acknowledgment signal OIUAX+ high, and this is gated with BINTE- and FCS12+. If either control state 01 or 10 is active (FCS12+ high) and the memory-protection interrupt is disabled (BINT- high), the interrupt-address flip-flop set output FIOD+ goes high on the positive transition of clock FRYX+. Acknowledgment of the interrupt during power-up causes the control logic to enter state 11, the power-quiescent state. Internal interrupt signals FCS12+ and FIOAK+ generate the interrupt address (OB05- and OB01-) and output it to the I/O bus with I/O-transfer control signal OBOUT-.

When an interrupt address is accepted (FIOD+ high) in control state 01 (FCS1+ high), the control flip-flop set output FC1X+ goes high. The flip-flop clears when FRSTA- goes low.

RESTORE reloads the volatile CPU registers with the same data as they held before power loss, and a jump instruction at the end of RESTORE reenters the interrupted program.

All lower priority options including all interrupts except memory protect are disabled during this time by a low PF/R-active signal FINTE—, generated by either the initialization or control flip-flop. The initialization flip-flop is cleared by a low system-reset signal FRSTA—. Upon completion of restore, FINTE— returns to the high state, allowing the I/O priority chain to become active again.

The system-start logic produces the short pulse OINT- low when the one-shot is set and the PF/R is in control state 01, thus starting the system.

Upon power failure, the **power-down sequence** begins with SPFA- going low, where it remains as long as there is a power-down condition. The negative transition enables the PF/R interrupt-request signals IUCX- and IURX-. When SPFA- goes low and the one-shot expires, the initializing flip-flop output FCX1- goes high on the next negative transition of clock OIUCX+ with the control logic going to state 10. The control flip-flop set output FC1X+ is high, and is reset when FCS2+ goes high. If the memory-protection interrupt is disabled (BINT- high), OIURX- goes low to enable the PF/R interrupt request to the CPU and cause the CPU to execute the SAVE subroutine.

Memory protection interrupts, if enabled, will be acknowledged first. Acknowledgment of the interrupt after the processor completes the instruction in progress is analogous to the acknowledgment during power-up, except that the control logic enters state 00 on power-down.

The request generates the interrupt address 040 specified by the PF/R, and executes the SAVE subroutine. During this sequence of events, all other interrupts and options except memory protection are disabled.

NOTE: If the processor is in a multiprocessor system and does not have the highest-priority memory port, it may not get port access in time to process the SAVE subroutine after power loss. PMA requests are ignored when the PF/R is active.

4.2.3 Programming

The user writes the PF/R SAVE and RESTORE service routines. During power-down interrupt addresses 040 and 041 must contain a jump-and-mark to the SAVE routine and during power-up interrupt addresses 042 and 043 must contain a jump to the RESTORE routine.

If the computer is in the halt mode during a power loss, the PF/R interrupt request is not acknowledged, the memory and CPU are immediately disabled, and the contents of the volatile registers are lost.

If the program instruction in process when power is lost is a multilevel addressing instruction, the CPU's acknowledgement of the PF/R interrupt request can be delayed up to five memory cycles. An active DMA request at the time of power loss (SPFA- going low) could delay acknowledgment up to one DMA cycle.

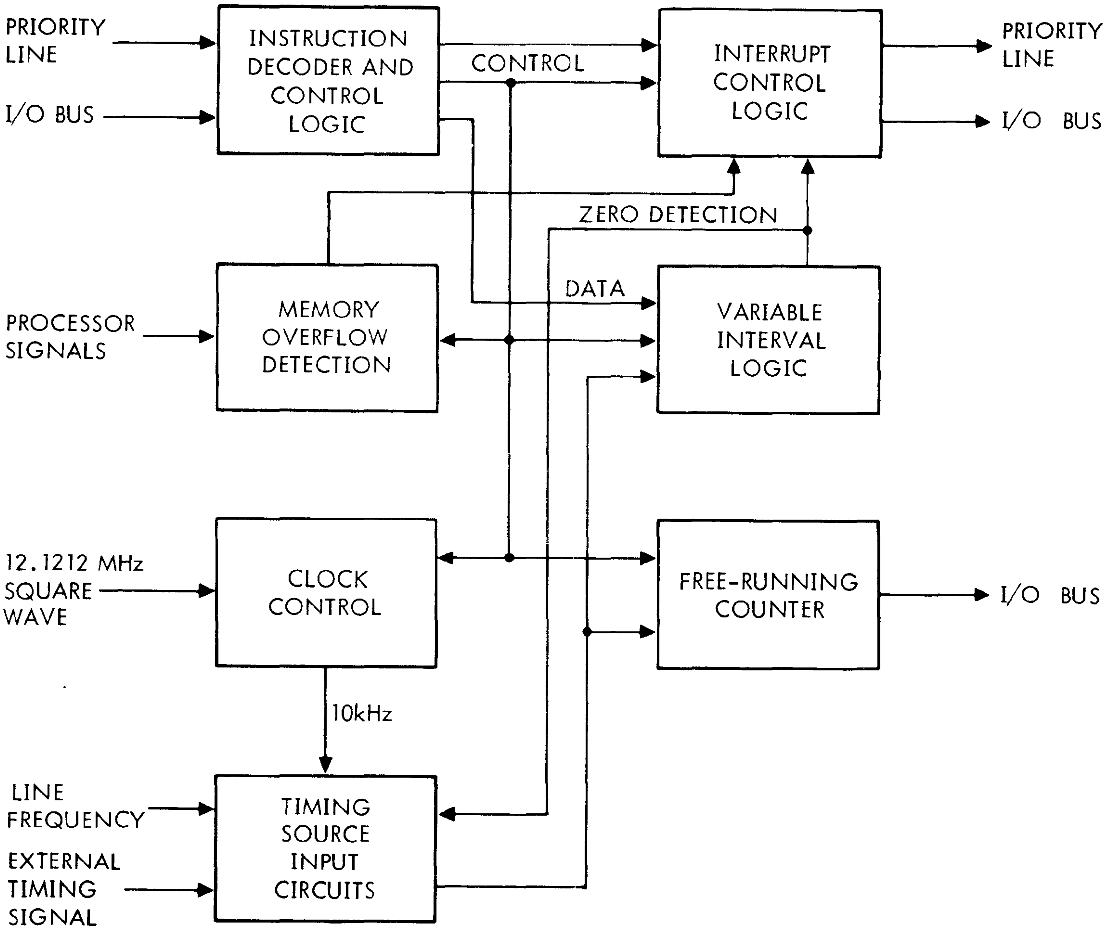
A halt instruction must be located at the end of the SAVE routine. This stops memory activity before SRST- is forced low by the power supply.

The service routine should contain a halt/run flag. This indicates whether a power failure occurred while the processor was in RUN or HALT mode, so that either the volatile registers can be restored or a HALT executed.

The RESTORE routine execution time must not exceed 400 microseconds, and the SAVE routine, 300 microseconds.

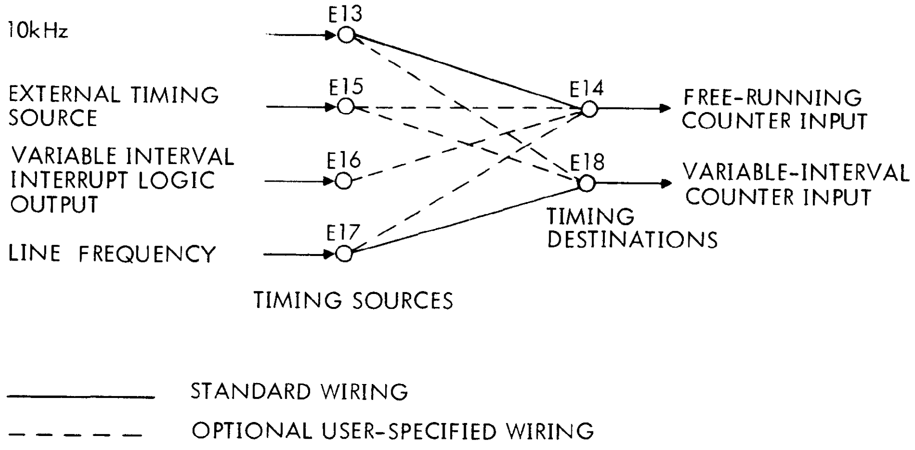
Figure 4-19 shows a typical PF/R service routine.

[redacted]



VT11-1578

Figure 4-20. RTC Block Diagram


10kHz
EXTERNAL TIMING SOURCE
VARIABLE INTERVAL INTERRUPT LOGIC OUTPUT
LINE FREQUENCY
E13
E15
E16
E17
TIMING SOURCES
E14
E18
FREE-RUNNING COUNTER INPUT
VARIABLE-INTERVAL COUNTER INPUT
TIMING DESTINATIONS
STANDARD WIRING
OPTIONAL USER-SPECIFIED WIRING

VT11-1581

Figure 4-21. Timing Source Selection

[redacted]

Table 4-3. Major RTC Functions *(continued)*

	Variable-Interval Interrupt		Free-Running Counter	
Function	Source	Range	Source	Range
High-rate interval interrupt and time-of-day accumulation	10 kHz	Same as above	Line frequency	Up to 18.2 minutes
High-rate interval interrupt and high-rate interval accumulation	10 kHz	Same as above	10 kHz	6.5 seconds in 100-microsecond increments
Low-rate interrupt and interrupt accumulation	Line frequency	16.7 milliseconds to 68.3 seconds in 16.7-millisecond increments	Variable-interval interrupt rate	18.2 minutes to 51.8 days, depending on interrupt rate
Low-rate interrupt and event accumulation	Line frequency	Same as above	External	Up to 65,536 events
Low-rate interrupt and time-of-day accumulation	Line frequency	Same as above	Line frequency	Up to 18.2 minutes
Low-rate interrupt and time-of-day accumulation	Line frequency	Same as above	10 kHz	6.5 seconds in 100-microsecond increments
Event count interrupt and interrupt accumulation	External	One to 4,095 events	Variable-interval interrupt rate	65,536 to 268 million events depending on interrupt rate
Event count interrupt and event accumulation	External	Same as above	External	Up to 65,536 events
Event count interrupt and time-of-day accumulation	External	Same as above	Line frequency	Up to 18.2 minutes
Event count interrupt and time-of-day accumulation	External	Same as above	10 kHz	6.5 seconds in 100-microsecond increments

4.3.2.1 Instruction Decoder and Control Logic

The instruction decoder and control logic consists of receivers and drivers, flip-flops, inverters, logic gates, and an instruction decoder. These logic components decode instructions from the CPU and generate the control signals to enable data transfers, mask or unmask interrupts, and initialize or clear the RTC.

The 16-bit bidirectional internal option bus interfaces the RTC and the processor (OB00– through OB15–). Instructions or data are accepted by the receivers and data are transmitted by the drivers. RB01 +, RB02 +, and RB05 + assert the RTC interrupt addresses on the option bus.

The instruction decoder is a 3-to-8-line decoder that decodes OB06, OB07, and OB08 into one of the states, REX00– through REX07– (REX05– not used). These signals enable or disable logic functions in the RTC circuits. The instruction decoder is enabled by RFRY +, OB11, and RDA47.

[redacted]

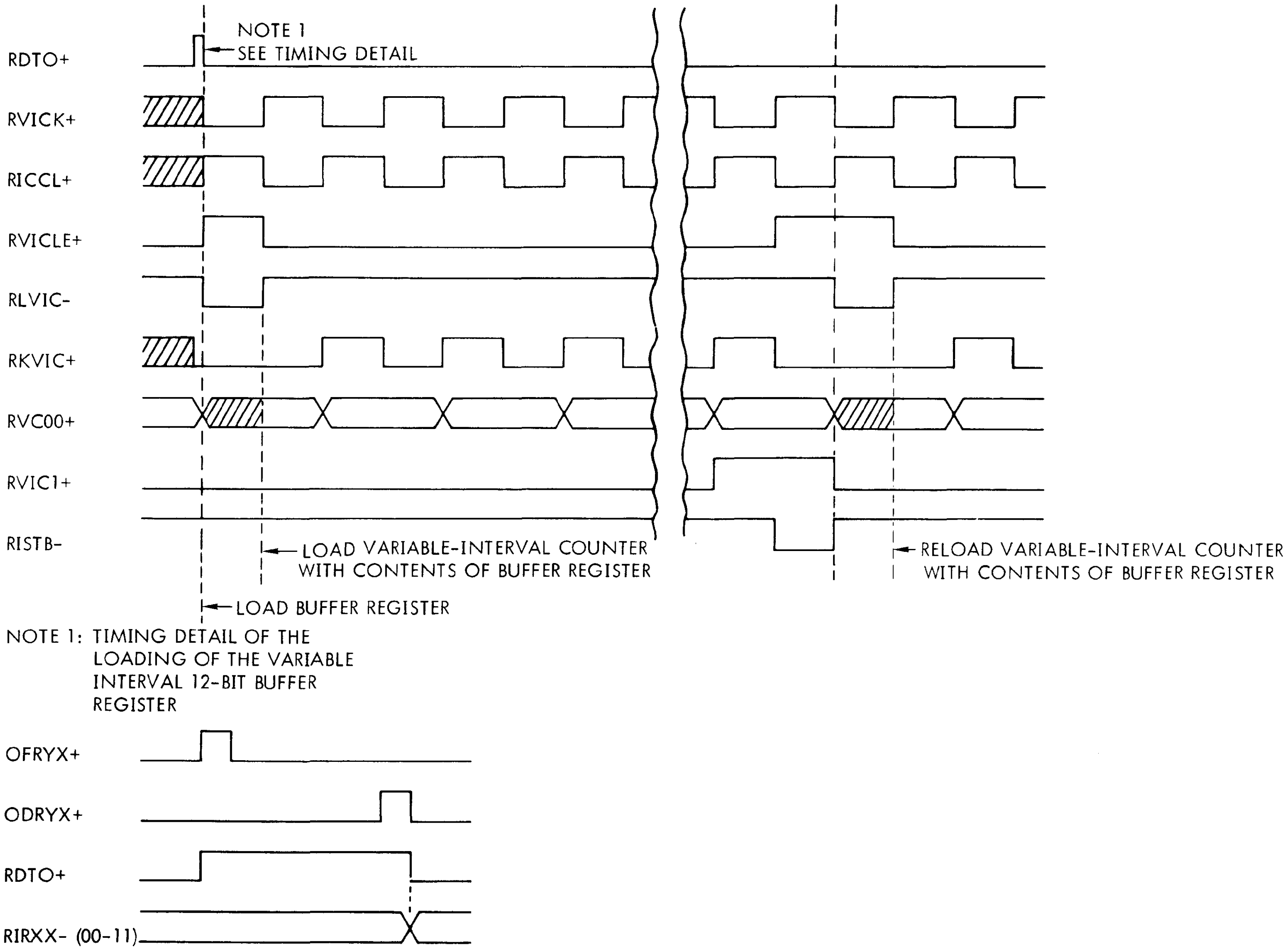


Figure 4-22. Variable Interval Counter and Load Buffer Timing

[redacted]

The second circuit consists of logic gates, flip-flops, and a 12-bit counter. RIR00– through RIR11– are transferred into the counter. The loading of the counter is sychronized with RVICK + so that the counter does not attempt to load and increment at the same time.

Note that if the 12-bit selection register is set to zero, the counter reacts as if the count were 4,096. When the counter reaches zero, it is again loaded and incremented.

The variable-interval interrupt continues to occur at a rate determined by the frequency source and the buffer register, until the contents of the buffer register are changed under program control.

The interrupt process is initiated by RISTB– setting RVIL every time the counter cycles, indicating a variable-interval interrupt is in process. RVII is then set, generating the variable-interval interrupt request when the variable-interval interrupt is enabled and the memory-overflow interrupt is not in process. RVIL can be cleared by OSYRT+, when the processor is transferring data out, or when the counter is initialized.

4.3.2.6 Memory Overflow Detection Logic

If an increment-memory-replace instruction is located at the variable-interval interrupt address (044), the memory overflow detection logic interrupts the processor when the contents of the incremented memory address overflow. Prior to incrementation, the contents of the memory address specified by the INR instruction in address 044 (typically address 045) are placed in the memory-input latch MILxx and the RTC memory overflow detection logic checks the 15th bit during the execution of an increment-memory-and-replace (INR) instruction. If this bit is set, indicating a count of 040000 or more, the RTC records an overflow condition.

The memory overflow detection logic consists of gates and flip-flops for generating the memory overflow interrupt request signal RMFI+. Flip-flop REMT sets while a variable-interval interrupt is being executed. The processor INR instruction signal CCS3D +, AK165 +, and REMT + combine to set test flip-flop RMTFF if the memory input latch bit 14 (MIL14A—) is set when overflow occurs. RMTFF— sets memory-overflow latch flip-flop RMFL. Memory-overflow interrupt flip-flop RMFI sets on the trailing edge of interrupt clock (OIUCX +) when RMFL is set and there is no variable-interval interrupt. RMFI + is sent to the interrupt control logic to generate an overflow interrupt.

4.3.2.7 Interrupt Control Logic

The interrupt control logic consists of the gates and flip-flops required to generate OIURX— and interrupt addresses, and to ensure RTC priority.

The priority output signal PRRTCO— is sent to lower-priority I/O devices to indicate that the RTC has seized interrupt priority. OIURX— is sent to the processor, along with DCPRM. RINT— low indicates that the interrupt request flip-flop is set and no higher-priority interrupt request is active. Either a variable-interval interrupt or memory-overflow interrupt request signal can then request the processor to execute an instruction.

Either the interrupt address 044 or 046 or the contents of the FRC counter may be placed on the option bus. When reading the FRC counter the contents of the counter are placed on the option bus. When an interrupt is being processed the interrupt address is placed on the option bus.

4.3.3 Programming

The user writes the RTC service routines using the RTC instructions in table 4-4. The RTC device address is 047. Interrupt addresses 044 and 045 process variable interval (increment) interrupts, and addresses 046 and 047 process overflow interrupts.

If the memory-overflow interrupt is used, 044 must be loaded with an increment-memory-and-replace instruction (i.e., INR, INRI, or INRE). If the memory-overflow interrupt is enabled, but not used, and address locations 044 and 045 do not contain an increment-memory-and-replace instruction, an erroneous memory-overflow interrupt may occur. Normally a jump-and-mark instruction to a suitable processing subroutine is loaded in the memory-overflow interrupt locations 046 and 047. However, if the overflow interrupt capability is not used and is masked, the variable-interval interrupt locations 044 and 045 should contain a jump-and-mark instruction.

Figure 4-24 shows a typical RTC service routine.

The RTC service routine concludes with enabling instructions for other internal computer options (e.g., priority interrupt modules) whose interrupt requests have been inhibited.

[redacted]

```
                   ***RTC INITIALIZATION CODING ***

                SET INTV  =  1000 (interrupts to occur once each
                                  second)
                SET INSC  =  10 (variable interval interrupts, 1 per
                                  million)

INTV      EQU       1000
          ORG       044                 INITIALIZE ADDRESSES 044-047
          INR       PERIOD
PERIOD    DATA      040001—INTV
          JMPM      TIMUP

                        ***INITIALIZE RTC***

          ORG       0500
INSK      DATA      INSC                INTERVAL SELECT COUNT
BEGIN     EXC       0447                INITIALIZE RTC
          EXC       047                 Clear FRC
          LDA       INSK
          OAR       047                 OUTPUT INT SEL COUNT
          EXC       0147                ENABLE RTC

              (USER'S PROGRAM)

                     ***INTERRUPT PROCESSOR***

          ORG       01000
TIMUP     ENTR                          PROGRAM COUNTER STORED
          EXC       0247                INHIBIT MOI
          STA       SAVEA               SAVE A REGISTER CONTENTS
          LDAI      040001—INTV         REINITIALIZE RTC COUNTER
          STA       PERIOD

              (Other instructions can be inserted here)

          LDAI      0                   RESTORE A REG
SAVEA     BES       0
          EXC       0147                ENABLE MOI
          JMP*      TIMUP               RETURN TO INTERRUPT
          END
```

Figure 4-24. Typical RTC Service Routine (in symbolic coding form)

4.4 MEMORY PROTECTION

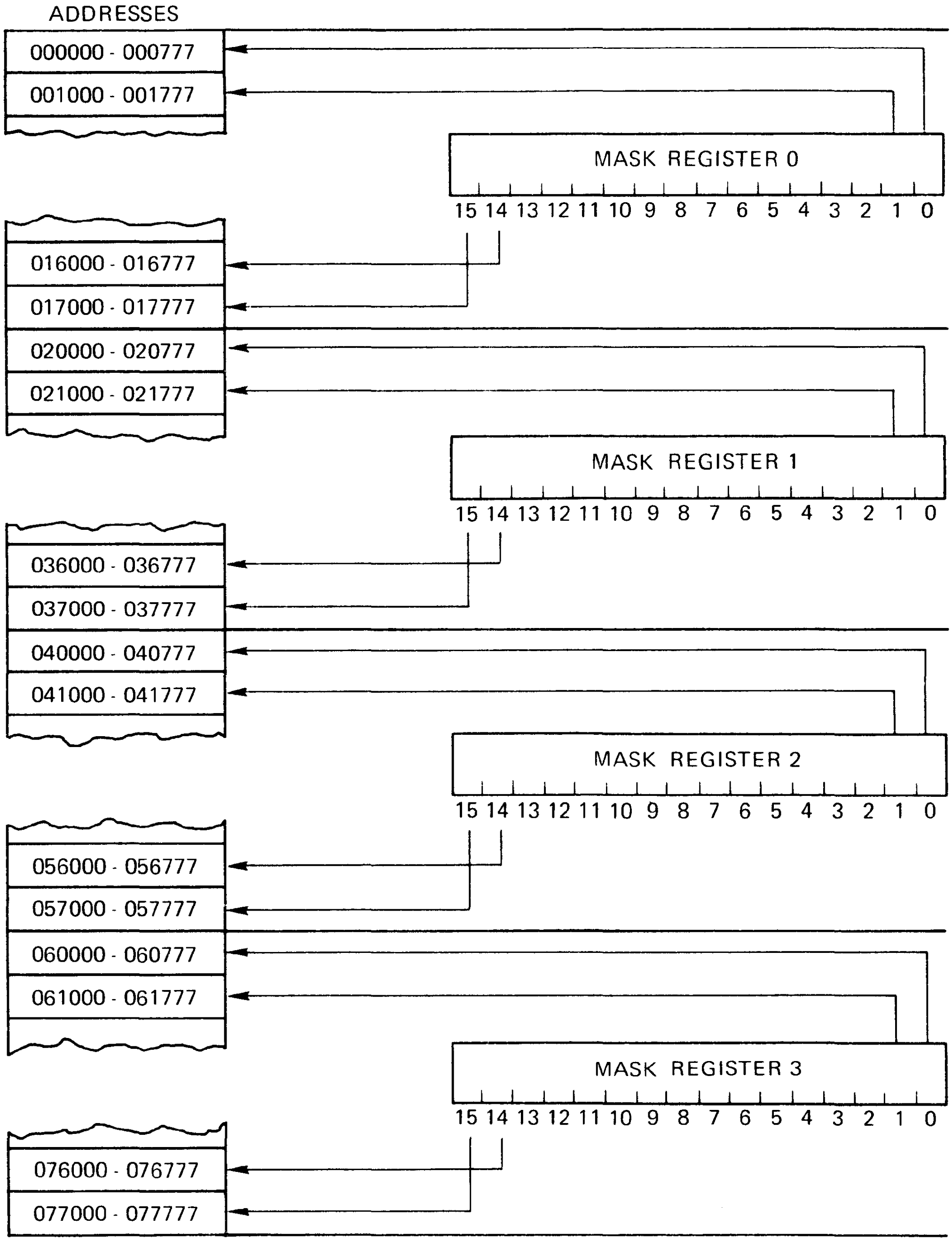
The **memory protection (MP)** option consists of the sections shown in the block diagram (figure 4-25) and described below.

The **address sequencer** checks for program entry into non-protected memory segments, controls loading of the instruction address register, checks for jumps and overflow from non-protected to protected memory segments, and checks for write to protected memory segments from a program originating in non-protected segments.

The **instruction sequencer** checks for loading of the program counter for a possible jump error and checks for input/output and halt instructions originating in a program residing in a non-protected memory segment.

The **mask register section** (figures 4-26 and 4-27) stores the protected/unprotected status of each 512-word segment of core memory. If a given flip-flop in the mask register is set, the corresponding segment of core memory is protected. If the flip-flop is reset, that memory segment is unprotected. The mask register section consists of eight 16-bit mask registers. To load a register, the register is

[redacted]



VTII-0237B

Figure 4-26. Mask Register Bit Assignment

[redacted]

The **input/output section** decodes the device address, and controls data transfers to the mask register and from the instruction-address register. The I/O section also gates the interrupt address on the I/O bus.

4.4.1 Error Sequences

The MP prevents: (1) modification of the contents of protected areas of memory by instructions in unprotected areas, (2) program entry into protected areas of memory while executing programs in unprotected areas, and (3) loss of computer control by protected-area programs to unprotected-area programs. Any attempt to accomplish one or more of these conditions is detected as an **MP error condition.** Tables 4–5 and 4–6 show the MP error conditions possible under various circumstances.

An instruction is a **protected instruction**, if the first (or only) word is in a protected area of memory. By definition, no MP error conditions can exist during the execution of such an instruction, nor can they exist during the execution of an instruction in a protected area that indirectly references an unprotected address.

Since the application of pipelining techniques makes it impossible to associate directly the address status with the instruction being decoded, two flip-flops (BANPT, and BINPT) keep track of address status for later use in error-checking. BANPT sets when an instruction from an unprotected memory address is placed in the instruction register. BINPT sets when an instruction is decoded while BANPT is set.

HALT ERROR: This condition exists when a HLT instruction is attempted from an unprotected area, or when a HLT is attempted from any area via an execution instruction located in an unprotected area. (However, an execution instruction in a protected area can execute a HLT located in any area.) Upon detection of a halt error, the HLT completes, the CPU is interrupted to 020, and halts after fetching the instruction at this address. The error detection causes the MP to hold its start signal (OINT—) true until completion of the interrupt. Thus, the CPU immediately reenters RUN mode after halting. *NOTE:* A manual halt using the STEP switch on the control panel cannot be detected by the MP.

OVERFLOW ERROR: This condition exists when a single word instruction, or either word of a two word instruction, is in the last location of an unprotected area and the first word of the next instruction lies in sequence in the next area, which is protected. (Note that, when the last location of an unprotected area contains the *second* word of a jump, indexed jump, jump-and-mark, jump-and-set return, or bit-test instruction for which the jump condition is met, an overflow error condition does not exist. When an overflow error is detected, the last instruction in the unprotected area completes, the next instruction in sequence is not executed, and the processor is interrupted to 030.

I/O ERROR: This condition exists when a single-word I/O instruction, or the first word of a double-word I/O instruction, is in an unprotected area, or if execution of an I/O instruction located in any area is attempted via an execution instruction located in unprotected area. (However, an execution instruction in a protected area can execute an I/O instruction located in any area). When an I/O error is detected, the I/O instruction in process completes. However, in all such cases I/O control commands and data transfers between the processor and peripheral controllers are inhibited. Thus, the contents of memory and of registers A, B, and X remain unmodified by the I/O instruction. This is accomplished by holding OINT— true. Upon completion of the instruction the processor is interrupted to 022.

WRITING ERROR: This condition applies only to instructions that write into memory (operand storage instructions) and is defined as existing when an instruction attempts to write into a protected area, and either the single word instruction, or the first word of a double word instruction, is in an unprotected area, or the instruction attempting to write is being executed as a result of an execution instruction in an unprotected area. (However, an execution instruction in a protected area can execute a writing instruction in any area.)

For an immediate instruction a writing error exists only when the first word is in an unprotected area, and the second word is in a protected area.

All operand store instructions end with an operand storage cycle, during which writing errors can be detected. All memory cycles addressing protected memory from unprotected memory will be changed unconditionally to reading cycles. To change a writing cycle to a reading cycle, the read/write (MWLY- MWRY-) inputs to the memory are pulled down to nearly 0V.

When a write error is detected, the instruction in process completes. However, the writing cycle is changed to a reading cycle to prevent modification of memory. The contents of registers A, B, and X remain unaffected. Upon completion of the instruction the processor is interrupted to 024.

JUMP ERROR: This condition applies only to a jump, indexed-jump, jump-and-mark, jump-and-set return, bit-test, or skip-on-register-equal instructions, and exists when an attempt is made to jump or skip to a protected area and the first word of the instruction lies in an unprotected area.

The jump, indexed-jump, jump-and-set-return, and bit-test instructions each ends in an addressing cycle, and the skip-on-registers-equal (SRE) instruction ends in a special skip cycle. The final (effective) address is not available until the final address is on the memory bus during the first memory cycle following the jump address cycle.

For the conditional jump instructions in the case where the jump condition is not met, a jump error cannot occur. However, an overflow error may still occur for this case.

[redacted]

Table 4-6. **MP Error Conditions Possible During Double Word Instructions in Unprotected Area** *(continued)*

Double-Word Instruction	Always possible	Under certain conditions	Always possible	Under certain conditions	Always possible	Under certain conditions
Immediate (Read)	--	--	O	--	O	--
Immediate (Write or INRI)	--	--	O,W	--	O	--
Execution (condition met)	--	H,I (not simultaneously)	O	--	--	H or O,I
Execution (condition mot met)	--	--	O	--	O	--
Jump, Jump-and-Mark, Indexed Jump, Jump-and-Set-Return, or Bit Test (condition met)	--	J	O	J	--	J
Jump, Jump-and-Mark, or Bit Test (condition not met)	--	--	O	--	O	--
Skip on Registers Equal (condition met)	--	J	J,O	--	J	--
Skip on Registers Equal (condition not met)	--	--	O	--	O	--

H = Halt error
O = Overflow error
I = I/O error
W = Writing error

4.4.2 Circuit Description

As aids in understanding this section, refer to the error sequences (section 4.4.1), programming considerations (section 4.4.3), mnemonics (section 6), and logic diagram (volume 2).

The **mask register** flip-flops store the protected/unprotected status of each 512-word segment of memory. If a flip-flop is set, the corresponding memory segment is protected. A memory segment is unprotected when the corresponding mask-register flip-flop is reset (figures 4-26 and 4-27).

There are four 16-bit mask registers. An external control instruction selects the register to be loaded. A data-transfer-out instruction then transfers 16 bits of data to the selected register via the option bus (OB00 through OB15). *NOTE: The mask register cannot be read back to the processor.*

[redacted]

Table 4-7. Interrupt Addresses for MP Errors

Error	Interrupt Address
HALT	020
OVERFLOW	030
I/O	022
WRITING	024
JUMP	026

OPERATION IN PROTECTED AREA: When leaving a protected area to enter an unprotected area, the next-to-last location (i.e. the one preceding the jump-type instruction)of the exit subroutine must be an enable-MP instruction. If the unprotected area is entered by " overflowing" from the protected area, the last instruction in the protected area must be an enable-MP instruction.

I/O INSTRUCTIONS: The I/O instructions for programmed communication between the MP and the CPU are listed in table 4-8.

I/O FUNCTIONS: To load one of the four 16-bit mask registers, first select the register using the proper select-mask-register external-control instruction. Any one of the three available data-transfer-out instructions can then be issued to load the mask register. *NOTE:* The MP need not be enabled to load a mask register.

MP error detection is enabled by the enable-MP external-control instruction. Detection is disabled by the disable-MP instruction, by CPU acknowledgment of an MP error interrupt, or a true system-reset signal from the control panel or power supply.

The enabled status of the MP is lost when power fails. When power is restored the MP is automatically disabled by system reset from the power supply. The contents of the mask registers are reinstated prior to reenabling the MP.

The address (plus 1) of the instruction during which the error was detected can be transferred to the processor using any one of the five data-transfer instructions. This capability is provided for use during the error subroutines.

One means of returning to the executive program in a protected area upon completion of the program being executed in an unprotected area is to program a jump to a previously assigned address in the protected area. This creates an MP-error condition. The jump error subroutine then allows the jump to occur upon recognition of the special jump address.

4.5 MEMORY PARITY

The three main blocks of the memory parity option (PARITY) are the parity generator/checker, control logic, and interrupt logic (figure 4-28). Inputs to PARITY include the memory data bus, memory writing-control lines, memory-completed line, and I/O bus signals.

4.5.1 Circuit Description

The **parity generator/checker** checks odd parity in an output (read) sequence and generates odd parity in an input (write) sequence. During input, MWRY+ and MWLY+ are high. The ninth bits, (EINR+, EINL+) into the parity generator/checkers are high, and if all of the memory bus data bits (MYD00- through MYD15-) are high, there are high outputs EWRPR+ and EWLPR+. These high outputs generate parity for right and left bytes to memory on lines MYD16- (low) and MYD17- (low), respectively. During output, MWRY+ and MWLY+ are low. Either EINR+ or EINL+ can be high or low, depending upon the previous states of MYYD16- and MYD17-. If MYD16- produces an even number of lows on the parity generator/checker input, then EWRPR+ is high and there is no output parity error on the right byte.

Table 4-8. MP I/O Instructions

Instruction Type	Octal Code	Function	
External Control	100045	Select mask register 0	
	100145	Select mask register 1	
	100245	Select mask register 2	
	100345	Select mask register 3	
	100645	Enable memory protection	
	100745	Disable memory protection	
Data Transfer	102545	Clear and input to A register	From Instruction
	102645	Clear and input to B register	Address Register
	102145	Input to A register	
	102245	Input to B register	
	102045	Input to memory	
	103145	Output from A register	To
	103245	Output from B register	Mask
	103045	Output from memory	Register
Sense	------	None	

[redacted]

The control logic determines priority for the four input channels, provides enabling signals to the other functional blocks, full-interlock control of the four input channels, and control signal interface with the memory access logic.

The priority logic assigns priority to the four PMA channels on a hardware basis and controls the channel-acknowledgment lines. Channel one has the highest priority and channel four the lowest. The priority logic constantly monitors the four channels. When a request is received, priority is established and the proper acknowledgment line is enabled. The priority logic checks priority when ACKEN+ is received from the data transfer logic. This allows a higher priority channel request between each PMA memory cycle even if a lower priority channel is requesting consecutive memory cycles. If a higher priority channel leaves its request enabled, all lower priority channels are locked out.

The data transfer logic controls the flow of data through the PMA.

The eighteen data drivers and eighteen input data receivers provide the gating and interface between the data buffer (DBxx) and the PMA 16-bit bidirectional data bus (PDxx+).

The eighteen address buffers (ABxx) provide gating and interface between the memory address bus (MYAxx) and the PMA address bus (PAxx). The most significant and least significant bits of the PMA data and address buses correspond to the most significant and least significant bits of the memory data and address buses, respectively.

4.6.1 PMA Option/Controller Interface

The four PMA option channels share the 20-bit address bus, 18-bit bidirectional data bus, and the READ, SYRT—P, PMRS, and GO lines. Each channel has its own request (REQx—) and acknowledgment (ACKx—) lines.

The PMA controller provides interfacing synchronization between the I/O device and the PMA. The controller provides buffering to prevent memory rate reduction. If program-initiated block data-transfers are desired, the PMA controller can also interface with the external I/O bus. In this instance, the PMA controller is sent an initial address and block length via the I/O bus. The controller interrupts the program when the block transfer is completed.

Figure 4-29 shows typical control-line-out circuitry, 4-30 bidirectional bus circuitry, and 4-31 address-or control-line-in circuitry.

No more than eight controllers can tie onto the PMA option. They can be distributed among the four channels in any manner.

Maximum cable length is 20 feet (6 meters). The cable typically is a pair of 40-conductor flat ribbon cables. Termination resistors are mounted on the PMA option board. On the last controller there is a termination shoe at the end of the cable.

4.6.2 Timing

The timing for PMA output (reading) and input (writing) is explained in this section.

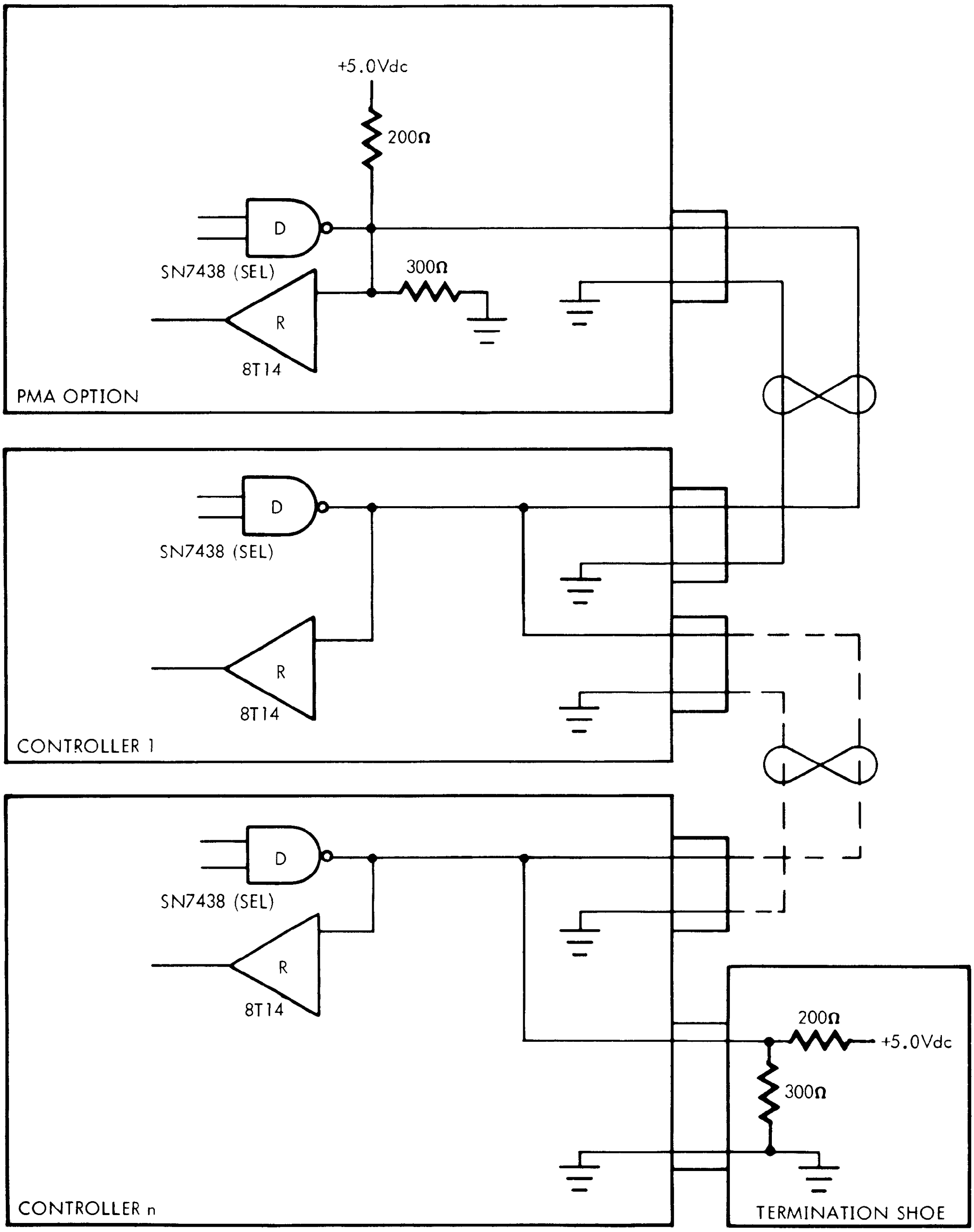
PMA output (Reading): In a reading operation (figure 4-32), when a request (REQx—) from a controller is received, priority is first established. The PMA can operate with the exclusive use of a memory port, in which case memory priority is absolute and immediate. This mode is hardwired in by a jumper from E89 to E90 that grounds ACKJ—. The PMA can also share a memory port, in which case PMA sends ORQM+ to the processor requesting memory access. When the acknowledgment (MAKO+) is received, the port will be free for the PMA on the following cycle.

The PMA then sets the proper PMA acknowledgment (ACKx—) and sends a PMA memory-request (ORQM—). The PMA controller gates the memory address (PAxx+) onto the PMA address bus, and makes READ— true and enables GO—. The PMA waits for GO— and the memory access logic to respond with the memory acknowledgment signal. Then the read flip-flop (AREAD), memory-address buffer (MYABxx+), and memory-port request (MRQY—) are put on the memory bus to await the memory-complete (YDNM—). YDNM— resets ACKx—, removes the PMA address from the memory bus, and resets the PMA memory-request if there are no other requests. Then YDNM— gates the memory output data to the PMA buffer (DBxx) and sends the data in the data buffer to the PMA data bus (PDxx+). The trailing edge of ACKx— tells the PMA controller that the data on the PMA bidirectional bus (PDxx+) are stable for at least 75 nanoseconds. Except when using the continuous-request (HOG) line, the controller removes all signals from the PMA option interface (including GO—) when ACKx— goes false.

PMA Input (Writing): A PMA writing operation (figure 4-33) is similar to reading. However, the PMA option output data is not gated to the PMA bus (PDxx) and READ— remains false. Input data to memory are gated onto the PMA bidirectional bus (PDxx—) when the address is gated onto the PMA address bus (MYABxx+). The PMA, in turn, gates the data into the data buffer (DBxx) when the memory address is gated into the address buffer (ABxx). The return of YDNM+ from memory completes the operation.

PMA Continuous-Request (HOG) Line: When a PMA controller is requesting consecutive memory cycles, its request line (REQx—) remains true. The basic sequence remains unchanged, including the determination of priority preceding each cycle. After the previous acknowledgment has cleared, the PMA responds in 165 nanoseconds with another ACKx— if the request line remains true and no higher priority device is active. By delaying GO— after ACKx— is received, the memory cycling is synchronized to controller demand. Normally, during the delay between ACKx— and GO—, the determination of PMA channel priority requests is suspended. The controller can force memory to appear busy on port A by activating the continuous-request line (HOG—). Priority now stays with port B even if it is idle. This reduces PMA latency times.

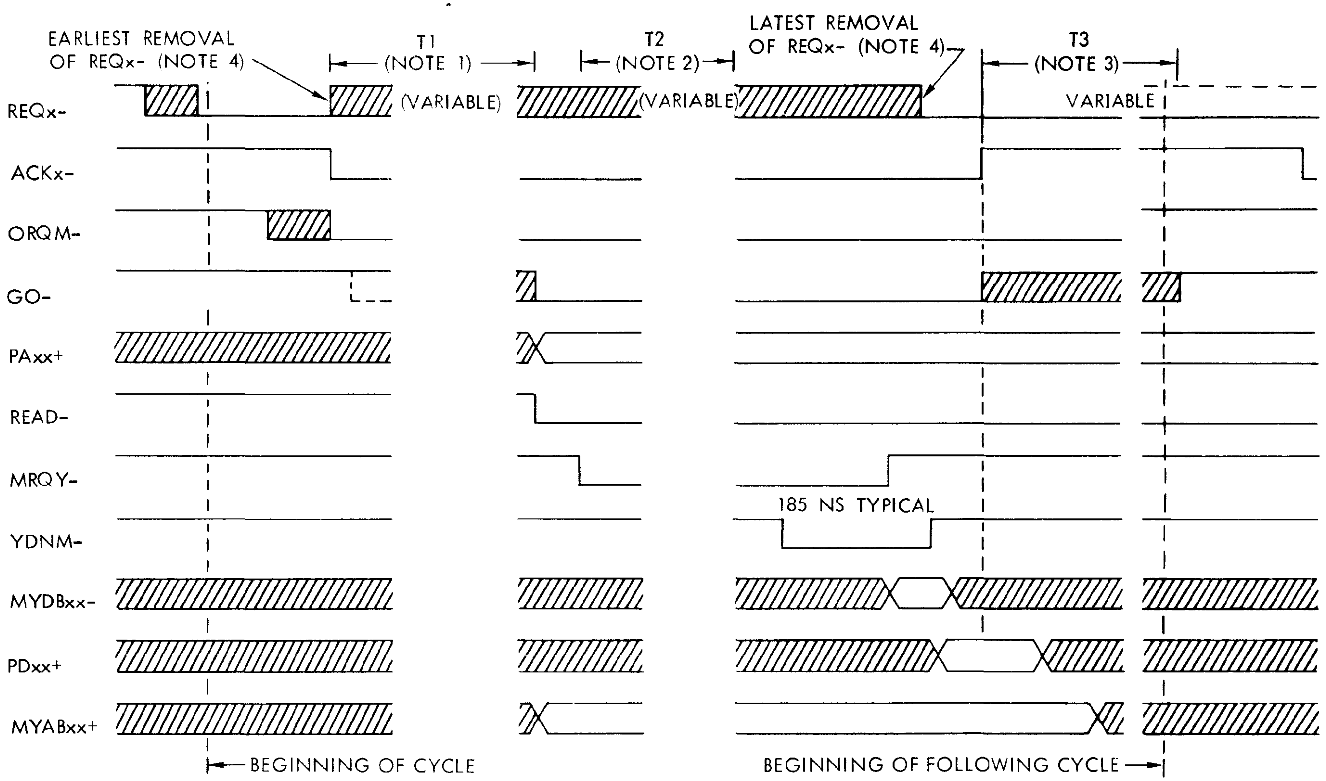
[redacted]



VT11-1595A

Figure 4-30. Typical PMA Bidirectional-Bus Line

[redacted]



Note 1. T1 = Delay for controller to return GO. 0 < T1 < ∞

Note 2. T2 = Delay for previous memory cycle to complete

Case 1: Clip (E89-E90) Installed PMA accessing different 8K of memory than CPU. T2 = 0

Case 2: Clip installed. PMA accessing same 8K of memory as CPU.

IC Memory	Core Memory
T2 = { 330 ns maximum 0 ns minimum 165 ns average	T2 = { 660 ns maximum 0 ns minimum 330 ns average

Case 3: Clip not installed.

T2 is the same as in case 2 above.

Note 3. T3 = Delay for controller to remove GO. 0 < T3 < ∞

Note 4. The PMA request (REQx-) may be removed as early as the leading edge of PMA acknowledge (ACKx-). It must be removed no later than 400 ns after the leading edge of PMA acknowledge if subsequent cycles are to be avoided.

VT12-412A

Figure 4-32. PMA Output (Reading)

[redacted]

4.6.3 Circuit Description

This section describes the circuits indicated in the block diagram (figure 4-34).

REQUEST AND ACKNOWLEDGMENT: The request and acknowledgment logic consists of four channels that have hardware-assigned priorities in the sequence channel 1 through channel 4.

Channel priority is determined by gating the output of the four channel flip-flops ARQ1 through ARQ4. These are set by the request signals REQ1– through REQ4–, respectively, on the 82-nanosecond clock AK82–. If more than one request signal is true, the parity gating determines which one of these signals is to be next copied onto one of the acknowledgment flip-flops ACK1 through ACK4, thus producing one of the acknowledgment signals ACK1– through ACK4– inverted from the flip-flop set outputs ACK1+ through ACK4+.

CONTROL AND CLOCK LOGIC: The control and clock logic has four non-inverting receivers that accept control signals HOG–, PMRS–, GO–, and READ–.

HOG– true gates the memory-port override signal MHGY– or MHMY– from PMA to memory if there is no power failure (FINTE– high).

A controller resets the PMA by enabling PMRS–, which gates reset signal ARST– low to reset the request, memory, and data sequencer flip-flops.

The PMA option receives the controller acknowledgment (GO–), activating the request and memory sequencer logic.

The PMA bus control signal READ– gates READA– and sets the memory access mode (read/write) for a given transfer.

The three free-running common-logic clocks MOCLK–, LK82+, and MFCO+ produce clock signals AK41+, AK82±, and AK165+ respectively. The numbers in the clock-signal mnemonics indicate noanoseconds. AK165+, derived from the processor clock, is constantly being phased by processor memory control and is synchronous with all processor operations. AK41+ and AK82+ have no phase relation to AK165+.

REQUEST SEQUENCER: The three-stage request sequencer is composed of flip-flops that are triggered on the high-to-low transition of the free-running 82-nanosecond clock (AK82+).

The request-sequencer set output (ARF1+) goes high when: (1) a request is present (ARQ+ high), (2) there is no power failure (FINTE– high), (3) the request-sequencer decoder is idle (ARDCI+ high), and (4) a PMA controller acknowledgment is not active (AGO– high).

ARF2+ goes high when memory-sequencer signal AM1S+ is high.

The request-sequencer set output (ARF3–) goes low at the same time as flip-flop 2. It is used only in the write mode. In read mode the set is blocked by READA+ high. The request sequencer flip-flops are reset by ARIR+ being high when a clock pulse AK82+ is present.

The memory priority request delay flip-flop set output (AROQ+) is enabled when AR1S+ goes high. It is clocked by AK82+. AROQ– low (or AR1S– low) then makes the 165-nanosecond request flip-flop set output (ORQM+) high on the high-to-low transition of clock AK165+. ORQM– low is the memory-priority request.

All request-sequencer flip-flops are cleared by ARST– (reset PMA).

MEMORY SEQUENCER: The three stage flip-flop memory sequencer is clocked on the free-running 41-nanosecond clock (AK41+). This Johnson counter progresses through six states during each memory sequence.

The memory-port request (MRQY–) is output when the first memory-sequencer flip-flop set output (AMF1+) is made high by AM1S+. AM1S+ goes high when: (1) the PMA has memory-port priority (AACKA+ high); (2) the PMA GO signal (AGQ+) is high, and (3) memory-sequencer flip-flops 2 and 3 are off (AMF2– and AMF3– high). The first flip-flop is reset by AMDC7+ when all three flip-flops are set (AMF1+, AMF2+, and AMF3+ high) and AK82+ is high.

The memory-sequencer flip-flop 2 set output (AMF2+) goes high when flip-flop 1 is set (AMF1+ high) and flip-flops 2 and 3 are both off (AMF2– and AMF3– high). At the same time, the memory-sequencer signal AMDC1– is activated. Flip-flop 2 resets at the next clock pulse after flip-flop 1 is reset.

The flip-flop 3 set output (AMF3+) goes high when the first two flip-flops are set (AMF1+ and AMF2+ high), AMF3– is high, and the memory-sequence complete signal (ADDN+) is high. It is reset on the next clock pulse after flip-flop 2 is reset.

The memory-sequencer-idle and PMA address and data-buffer clock signals are activated when the memory sequencer goes off (AMF1–, AMF2–, and AMF3– high).

The enable PMA memory-port signal (AACKA+) is active when the memory priority request flip-flops (AROQ+ and ORQM+), and the memory-priority-acknowledgment signal (MAKO+) are all high. ACKJ– (jumpered to ground) enables AACKA+ if no other user is one the memory port.

The low PMA bus output-received signal (READ–) is clocked by the low-to-high transition of AMF2+ to produce set output (AREAD–) low, inhibiting AMDEN+. The high reset output (AREAD+) enables the memory bus signals for input when AACKA+ is high. These signals are low for output (read) operations.

The PMA controller acknowledgment-received signal GOA+ is delayed 30-nanoseconds (GOAD+) to set the PMA GO

[redacted]

flip-flop set output pulse AGO+ high. The flip-flop is clocked on the high to low transition of AK41+. When AMF1+ is high, AGOL+ goes high to reset the AGO+ flip-flop. The AGOL+ flip-flop is reset when there is no PMA controller acknowledgment received (GOA- high) and the first memory sequencer flip-flop is reset (AMF1- high). This dual flip-flop latching circuitry ensures that only one GO signal occurs for every PMA request.

DATA SEQUENCER: The three-stage data sequencer flip-flops are clocked by the free-running 41-nanosecond clock (AK41±) and the 82-nanosecond clock (AK82+). The data sequencer starts when the memory-sequencer signal AMDC7+ is high and AREAD+ is received. This produces data-sequencer signal AD1S+ high, which sets flip-flop ADF1. At the same time, reset output ADF1- low gates PMA data-bus enabling signal APDEN+.

The second flip-flop set output signal ADF2+ goes high when ADF1+ and ADDN- (memory sequence incomplete) are high. This resets the first flip-flop and sets the third flip-flop (ADF3- low). ADF3- low enables the PMA data bus. The second flip-flop is reset by AD25+ low and AK41- going high. The first two flip-flops can also be cleared by the low reset PMA signal (ARST-).

The memory data buffer clock (AKMD+) is active when a memory sequence is incomplete (ADDN+ high) and ADF1+ is high. The memory-sequence complete signal is YDNM-.

ADDRESS BUFFER: The inputs to the address buffer consist of the PMA address-bus signals (PA00+ through PA19+) from the selected controller. These 20 signals go through separate non-inverting receivers to three six-bit latches. When the PMA address clock (AMDCI—) goes low, the latches are set by the incoming signals.

The memory-address signals (MYAB00+ through MYAB08+, MYPM09+ through MYPM15+, and MYK16+ through MYK19+) are then gated out to the memory bus when the PMA has memory-port priority (AACKA+ high).

DATA BUFFER: The data buffer has 18 data drivers and 18 data receivers that provide an interface between the memory bus and the 18-bit bidirectional data bus.

During writing, data signals PD00+ through PD17+ enter the data buffer from the active controller. Each of the 18 signals goes through a separate, non-inverting receiver to set one of the 18 flip-flops on the low-to-high transition of AMDC1-.

Data signals MYDB00- through MYDB17- are then gated out to the memory bus when the memory data bus enable (AMDEN+) goes high.

During reading, data signals MYDB00- through MYDB17- enter the data buffer from the memory bus. These 18 data signals set the three six-bit latches when the memory-data buffer clock (AKMD+) goes low. The inverted signals AD00L+ through AD17L+ are reinverted and then gated onto the bidirectional data bus when the PMA enable data bus signal (APDEN+) goes high.

4.6.4 Programming

Program control of devices that utilize PMA memory cycles occurs exclusively between the processor and the individual device controllers attached to the PMA channels. This control is through the I/O bus.

[redacted]

SECTION 5

MAINTENANCE

The option board common-logic is exercised by testing the individual options and has no maintenance procedures of its own.

5.1 TELETYPE CONTROLLER

TC maintenance consists of running test programs, troubleshooting, and making repairs if required. The test executive program (92A0107-001) and TTY test program (92A0107-005), described in the 620 test program manual (98 A 9952 06x), in conjunction with the Varian 70 Series maintenance manual, help isolate an error condition. Troubleshooting is facilitated by familiarization with the operation of the TC and use of the logic diagram. This section provides troubleshooting data, program tests, and a list of reference documents to be used as maintenance aids.

5.1.1 Equipment

The following is a list of recommended test equipment for maintaining the TC.

a. Oscilloscope, Tektronix type 547

b. Multimeter, Triplett type 630

5.1.2 Test Programs

The condition of the TTY unit should be periodically checked using program tests. These tests for TC and TTY are provided as part of the regular troubleshooting package.

NOTE

> One section of the TTY test program for ASR models includes a print suppression test. The 33 ASR does not perform this function, so this test program should be bypassed when testing the 33 ASR (refer to the 620 test program manual).

The TTY test program is a good diagnostic aid because tee data being sent are printed out and can be analyzed. Also, known input patterns can be generated (via keyboard or paper tape) and data can be analyzed in the computer or returned to the TTY for printed analysis. If for some reason, such as PTR failure, the test program tapes cannot be read, a simple input/output program for verification and troubleshooting of the TTY-TC operation can be entered through the computer control panel. This program (figure 5-1) tests keyboard input and printer output:

a. Enter the program through the control panel.

b. Turn the TTY to ON-LINE.

c. Program starts at address 000000.

d. Any character input from the TTY is transferred back to the TTY as an output from the processor TC almost immediately. Various character patterns and functions of the TTY can be checked by this echo method.

Location	Command	Description
00000	101201	Sense read ready.
00001	000004	If yes, jump to 00004.
00002	001000	Jump back to 00000.
00003	000000	
00004	102501	Clear and input TTY character to A register.
00005	101101	Sense write ready.
00006	000011	If yes, jump to 00011.
00007	001000	
00010	000005	
00011	103101	Output A register (to TC).
00012	001000	Jump back to 00000.
00013	000000	

Figure 5-1. Basic Input/Output Test Program

5.1.3 TC/TTY Troubleshooting

The TTY units are normally trouble-free and require little attention; however, if operation is faulty, the following troubleshooting procedures are suggested. Visually inspect for broken belts, loose cams or components, loose or poorly seated connectors, blown fuses, or burned-out components.

NOTE

> The TTY casework is cast and, therefore, somewhat fragile. Exercise care when removing and reinstalling it.

GARBLING: The following are possible sources of intermittent character change (printing or sending wrong characters).

TTY	Incorrect power supply output
TTY	Incorrect motor speed
TTY	Incorrect range adjustment
TC	Incorrect TC frequency
TTY-TC	Incorrect loop current (too low or too high)

[redacted]

5.3 REAL-TIME CLOCK

RTC maintenance consists of running the RTC test program, troubleshooting, and making repairs, if required. The RTC test program (part number 98A0107-035), described in the 620 test programs manual (document number 98 A 9952 06x), in conjunction with the Varian 70 Series processor maintenance manual, helps isolate an error condition. Troubleshooting is facilitated by familiarization with the operation of the RTC and use of the logic diagram.

5.3.1 Equipment

The following is a list of recommended test equipment for maintaining the RTC.

a. Oscilloscope, Tektronix type 547

b. Multimeter, Triplett type 630

5.3.2 Test Program

The RTC test program is an integral part of the MAINTAIN II test program system. It is provided as part of the regular troubleshooting package.

RTC operation should be periodically checked with the RTC test program, under the control of the test executive program (part number 92A0107-001). The variable interval and memory overflow interrupts and the free-running counter are software-timed and tested. Malfunctions are reported in the form of error messages and/or codes.

The RTC test program is described in detail in the 620 test programs manual (document number 98 A 9952 06**x**).

5.3.3 Troubleshooting

Troubleshooting the RTC comprises verification of critical input power and timing parameters.

INPUT POWER: Verify +5V dc ± 5 percent and common (ground).

SCHMITT-TRIGGER CIRCUIT: Verify 24V rms, 50- or 60-Hz ± 5 percent sinewave input at P01-A05 and 50- or 60-Hz ± 5 percent squarewave output at E3. Check that the output signal follows the input signal.

EXTERNAL TIMING SOURCE: If a user-supplied external timing source (TP1) is jumpered to either pin E18 (free-running counter) or pin E14 (variable interval counter):

a. Apply +5 ± 0.5V dc to TP1. Measure +5 ± 0.5V dc at E15.

b. Apply 0.0 ± 0.5V dc to TP1. Measure 0.0 ± 0.5V dc at E15.

5.4 MEMORY PROTECTION

MP maintenance consists of running the MP test program, troubleshooting, and making repairs, if required. The MP test program, described in the 620 test programs manual, in conjunction with the Varian 70 Series processor manual, helps isolate an error condition. Troubleshooting is facilitated by familiarization with the operation of the MP and use of the logic diagram.

5.4.1 Equipment

The following is the recommended test equipment for maintaining the MP.

a. Oscilloscope, Tektronix type 547

b. Multimeter, Triplett type 630

5.4.2 Test Program

The MP test program is an integral part of the MAINTAIN II test program system. It is provided as part of the regular troubleshooting package.

MP operation should be checked with the MP test program (92A0105-002), under the control of the test executive program (92A0107-001). MP malfunctions are reported in the form of error messages and/or codes.

The MP test program is described in detail in the 620 test programs manual (98 A 9952 06**x**).

5.4.3 Troubleshooting

Turn the computer off, open the front panel on its hinges, connect the option board in the top slot, turn the computer on, and make the following checks.

INPUT POWER: Verify +5V dc ± 5 percent and common (ground).

MANUAL TESTS: Performing the following manual tests, in conjunction with the MP test program, ensures a thorough test of the operation of the MP. Before executing the test subroutines described below, press RESET and set run mode.

MASK REGISTER ADDRESSING: Select a mask register and verify that its contents are not altered by data-transfer-out operations.

Address	Instruction	Mnemonic	Decription
000100	100045	EXC	Select mask register 0
000101	010113	LDA	
000102	103145	OAR	Load mask register
000103	010114	LDA	

(continued)

[redacted]

5.5 PARITY

PARITY maintenance consists of troubleshooting and making repairs as required. Troubleshooting with the oscilloscope and multimeter is facilitated by familiarization with the operation of the PARITY and use of the logic diagram.

Test PARITY as follows:

a. Use a functioning V70 series computer system.

b. Enable PARITY interrupts.

c. Output the bit patterns in table 5-1 from processor to memory in consecutive locations. Check that the parity bits are in the correct logical states at the time that YDMM— samples the memory bus, that memory bus timing requirements are met and that no interrupts occur.

d. Input the table from memory and check that no interrupts occur.

e. Remove jumper clip from bits 16 and 17 of the memory word on the option board so that on read-from-memory operations the parity bits to the parity circuits will be zero at all times.

f. Input the table again as in (d) and check that a parity interrupt occurs for each word input except 8 and 16. With an oscilloscope check that both EWRPR+ and EWLPR+ are low at the high-to-low transition of EKRDD-.

g. Repeat the above tests with interrupts disabled by grounding ENTR1- and check that no interrupts occur.

h. Repeat the above tests (c) through (f) with interrupts disabled by grounding ENTR1- and check that no interrupts occur.

i. Repeat the above tests (c) through (f) with interrupts disabled by the system reset on the front panel and check that no interrupts occur.

j. Repeat the above tests (c) through (f) and individually force FINTE- and BINTE- low and check that PRNE- is high.

k. With computer in step, perform system reset and check that PRNE- is high when FOS- is low and that PRNE- goes low when FOS- goes high.

If any of the above tests fail, repair the cause of the failure by tracing backwords through the logic, finding the cause, and replacing the defective component.

5.6 PRIORITY MEMORY ACCESS

PMA maintenance consists of running test programs, troubleshooting, and making repairs, if required. Troubleshooting is facilitated by familiarization with the operation of the PMA and controllers, and with their respective logic diagrams.

The PMA test and associated hardware tests are not part of this manual as they are designed for in-house testing only. This section of the manual is intended for field-service troubleshooting.

Table 5-1. Parity Check Pattern

Byte		Left Byte									Right								
Memory Bits		15	14	13	12	11	10	9	8	7	7	6	5	4	3	2	1	0	16
	1	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1
	2	0	1	1	0	0	0	0	0	1									1
Word No.	3	0	0	1	1	0	0	0	0	1									1
	4	0	0	0	1	1	0	0	0	1									1
	5	0	0	0	0	1	1	0	0	1									1
	6	0	0	0	0	0	1	1	0	1									1
	7	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	1
	8	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
	9	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	1
	10									1	0	1	1	0	0	0	0	0	1
	11									1	0	0	1	1	0	0	0	0	1
	12									1	0	0	0	1	1	0	0	0	1
	13									1	0	0	0	0	1	1	0	0	1
	14									1	0	0	0	0	0	1	1	0	1
	15	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	1
	16	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

SECTION 6
MNEMONICS

This section contains lists of option-board signal mnemonics. Immediately below are the mnemonics for the common logic, followed in the subsequent subsections by lists for the individual options.

Mnemonic	Description
DRYX-I	I/O bus data ready
EB00-I	I/O bus bit 00
EB01-I	I/O bus bit 01
EB02-I	I/O bus bit 02
EB03-I	I/O bus bit 03
EB04-I	I/O bus bit 04
EB05-I	I/O bus bit 05
EB06-I	I/O bus bit 06
EB07-I	I/O bus bit 07
EB08-I	I/O bus bit 08
EB09-I	I/O bus bit 09
EB10-I	I/O bus bit 10
EB11-I	I/O bus bit 11
EB12-I	I/O bus bit 12
EB13-I	I/O bus bit 13
EB14-I	I/O bus bit 14
EB15-I	I/O bus bit 15
FRYX-I	I/O bus function ready
IUAX-I	I/O bus interrupt acknowledgment
IUCX-I	I/O bus interrupt clock
IUJX-I	I/O bus interrupt jump
IURX-I	I/O bus interrupt request neg.
LK82 +	Free-running 82-nanosecond clock
LK82–	Free-running 82-nanosecond clock neg.
MFC–	165-nanoseconds full clock
MFCO +	Memory free-running clock
MFHCO +	
MHC–	165-nanoseconds half clock
MHCO +	
MOCLK±	Option board common logic clock
MTRC–	Option board common logic clock input for RTC and TTY
OBOUTA–	Option board common logic receiver disabler A
OBOUTB +	Option board common logic driver enabler B
OBOUTB–	Option board common logic receiver disabler B
OBOUT +	Option board common logic driver enabler A
OBOUT–	Option board common logic output driver enabler
OB00–	Option board common logic bit 00
OB01±	Option board common logic bit 01
OB02±	Option board common logic bit 02
OB03±	Option board common logic bit 03
OB04±	Option board common logic bit 04
OB05±	Option board common logic bit 05
OB06±	Option board common logic bit 06
OB07±	Option board common logic bit 07
OB08±	Option board common logic bit 08
OB09±	Option board common logic bit 09
OB10±	Option board common logic bit 10
OB11±	Option board common logic bit 11
OB12±	Option board common logic bit 12
OB13±	Option board common logic bit 13
OB14±	Option board common logic bit 14
OB15±	Option board common logic bit 15
ODRYX +	Option board common logic data ready
OFRYX +	Option board common logic function ready
OIUAX +	Option board common logic-interrupt ack.
OIUCX +	Option board common logic-interrupt clock
OIUJX +	Option board common logic-interrupt jump
OIURX±	Option board common logic interrupt request
OSYRT +	Option board common logic system reset
SYRT-I	I/O bus system reset
TR82 +	82.5-nanosecond clock to RTC and TTY

6.1 TELETYPE CONTROLLER

Mnemonic	Description
IUAA-I	Output (write) interrupt line to PIM
IUBB-I	Input (read) interrupt line to PIM
OBOUT–	Option board common I/O logic output driver enable R
OB00–	Option board common I/O logic bit 00
OB01–	Option board common I/O logic bit 01
OB02–	Option board common I/O logic bit 02
OB03–	Option board common I/O logic bit 03
OB04–	Option board common I/O logic bit 04
OB05–	Option board common I/O logic bit 05
OB06–	Option board common I/O logic bit 06
OB07–	Option board common I/O logic bit 07
OB11–	Option board common I/O logic bit 11
OB12–	Option board common I/O logic bit 12
OB13–	Option board common I/O logic bit 13
OB14–	Option board common I/O logic bit 14
ODRYX +	Option board common I/O logic data ready
OFRYX +	Option board common I/O logic function ready
OIUAX +	Option board common I/O logic-interrupt ack.
OSYRT +	Option board common I/O logic system reset
SERX-I	Sense response
TAQD +	Output of first 4-bit counter
TBQD +	Output of second 4-bit counter
TBREA +	TC input character ready
TBRL–	MOS transmitter load buffer
TB00–	Internal I/O bus bit 00
TB05–	Internal I/O bus bit 05
TCCLK +	Internal 82-nanosecond clock
TCLK +	MOS transmitter/receiver clock pos.
TCLK–	MOS transmitter/receiver clock neg.
TCQB +	QB output of third 4-bit counter
TCQC +	QC output of third 4-bit counter
TCQD +	QD output of third 4-bit counter

[redacted]

Mnemonic	Description
OSYRT +	Option board common I/O logic system reset
PRRTCI–	Priority-in input from a higher-priority controller
PRRTCO–	Priority-out to a lower-priority controller
RBS03	Inverted source of OB03–
RBS04–	Inverted source of OB04–
RBT14 +	Interval memory input-latch bit 14 pos.
RB01 +	Interrupt address or free-running counter bit 1 pos.
RB02 +	Interrupt address or free-running counter bit 2 pos.
RB05 +	Interrupt address of free-running counter bit 5 pos.
RCA +	Output of first stage of the variable-interval counter
TR82 +	82.5-nanosecond clock from common logic
RCB +	Output of second stage of the variable-interval counter
RCC +	Output of third stage of the variable-interval counter
RCLK +	Internal 12 MHz clock
RC08 +	Output of first-programmed counter
RC128 +	Output of second programmed counter
RDA47 +	Device address 047 decoded pos.
RDA47–	Device address 047 decoded neg.
RDDTI +	Data-transfer-in flip-flop input
RDHT	RDTI flip-flop preset
RDRY–	Inversion of ODRYX
RDTIA +	Data-transfer-in flip-flop pos. repowered
RDTI +	Data-transfer-in flip-flop pos.
RDTI–	Data-transfer-in flip-flop neg.
RDTO +	Data-transfer-out flip-flop pos.
RDTO–	Data-transfer-out flip-flop neg.
RFDS–	Preset memory overflow test enabler flip-flop neg.
REMT +	Memory overflow test flip-flop enabler pos.
RETR +	Internal/external timing source pos.
RETR–	Internal/external timing source neg.
REXCE–	Enable EXC instruction decoder
REX00–	EXC instruction decoder - 0
REX01–	EXC instruction decoder - 1
REX02–	EXC instruction decoder - 2
REX03–	EXC instruction decoder - 3
REX04–	EXC instruction decoder - 4
REX05–	EXC instruction decoder - 5
REX06–	EXC instruction decoder - 6
RFIP +	Clock RMFL flip-flop
RFMN–	Execute memory-overflow interrupt neg.
RFRCK +	Free-running counter clock
RFRCK +	Input to free-running counter from frequency selection circuit
RFRCK–	Input to free-running counter from frequency selection circuit neg.
RFREN +	Preset term to RKFCL flip-flop
RFRY +	Internal RTC function-ready pos.
RFRY–	Internal RTC function-ready neg.
RFR00 +	Free-running counter output bit 00 pos.
RFR01 +	Free-running counter output bit 01 pos.
RFR01–	Free-running counter output bit 01 neg.
RFR02 +	Free-running counter output bit 02 pos.
RFR02–	Free-running counter output bit 02 neg.
RFR03 +	Free-running counter output bit 03 pos.
RFR04 +	Free-running counter output bit 04 pos.
RFR05 +	Free-running counter output bit 05 pos.
RFR05–	Free-running counter output bit 05 neg.
RFR06 +	Free-running counter output bit 06 pos.
RFR07 +	Free-running counter output bit 07 pos.

Mnemonic	Description
RFR08 +	Free-running counter output bit 08 pos.
RFR09 +	Free-running counter output bit 09 pos.
RFR10 +	Free-running counter output bit 10 pos.
RFR11 +	Free-running counter output bit 11 pos.
RFR12 +	Free-running counter output bit 12 pos.
RFR13 +	Free-running counter output bit 13 pos.
RFR14 +	Free-running counter output bit 14 pos.
RFR15 +	Free-running counter output bit 15 pos.
RICCL–	Output flip-flop following variable clock source neg.
RINT–	RTC executing interrupt neg.
RIR00–	Interval-selection register output bit 00 neg.
RIR01–	Interval-selection register output bit 01 neg.
RIR02–	Interval-selection register output bit 02 neg.
RIR03–	Interval-selection register output bit 03 neg.
RIR04–	Interval-selection register output bit 04 neg.
RIR05–	Interval-selection register output bit 05 neg.
RIR06–	Interval-selection register output bit 06 neg.
RIR07–	Interval-selection register output bit 07 neg.
RIR08–	Interval-selection register output bit 08 neg.
RIR09–	Interval-selection register output bit 09 neg.
RIR10–	Interval-selection register output bit 10 neg.
RISTB–	Variable interval interrupt to interrupt-generation logic
RIUA +	Internal RTC interrupt acknowledgment pos.
RIUA–	Internal RTC interrupt acknowledgment neg.
RRIUC +	Internal RTC interrupt clock
RIUJ +	Internal RTC interrupt jump
RJM +	Enable internal RTC interrupt jump
RJM–	Internal RTC interrupt jump jumper
RKFCL +	Free-running counter clock
RKFRY–	Clock data-transfer flip-flops
RKMFI +	K-input to memory overflow interrupt flip-flop
RKVIC +	Variable-interval-counter clock
RKVII +	Variable-interval-interrupt flip-flop clock
RK10K +	Clock 10 KHz flip-flop
RK660 +	660-nanosecond free-running clock pos.
RK660–	660-nanosecond free-running clock neg.
RLF +	Internal line frequency source pos.
RLF–	Internal line frequency source neg.
RLVIC–	Load variable-interval counter neg. true to load
RMFI +	Memory-overflow-interrupt flip-flop pos.
RMFI–	Memory-overflow-interrupt flip-flop neg.
RMFL +	Memory overflow latch flip-flop pos.
RMFL–	Memory overflow latch flip-flop pos.
RMOE +	Memory overflow enabling flip-flop
RMTFF +	Memory overflow test flip-flop pos.
ROUT +	Output data to I/O common logic pos.
RPDTO–	Preset data-transfer-out flip-flop
RPRM +	Internal RTC priority-in line
RPVIE–	Preset varible-interval-interrupt flip-flop
RRME–	Reset memory-overflow-enabler flip-flop
RRRMT–	RMTFF flip-flop reset
RRST +	Reset RTC pos.
RRST–	Reset RTC neg.
RRVIE–	Reset varible-interval-interrupt flip-flop
RSYRT–	Inversion of OSYRT
RTC–	REMT and RMTFF flip-flops clock
RTINT +	Enable interrupt request pos.
RTINT–	Enable interrupt request neg.

[redacted]

Mnemonic	Description
EIUX1 +	Enable parity interrupt
EKENB	Enable parity interrupt flip-flop clock
EKRDD–	Delayed parity-error flip-flop clock
EKRD–	Parity-error flip-flop clock
ENTR1–	Parity option disabled by jumper
ERDL +	Parity check (left byte)
ERDR +	Parity check (right byte)
EREAD +	Enable parity-error flip-flop clock
ERERR–	Parity-error flip-flop direct reset
ERER +	Parity-error flip-flop
ERERS +	Parity-error flip-flop set
EWLPR +	Parity error left byte
EWRPR +	Parity error right byte
EWTL +	Input parity left byte
EWTR +	Input parity right byte
FINTE–	Power failure active
FOS–	Power-failure one-shot
MAKO +	Memory cycle for PMA
MWLY +	Memory left byte input
MWRY +	Memory right byte input
MYDxx	Memory data-bus bits
OBOUT–	Option board common I/O logic output driver enabler
OBxx–	I/O bus bit
OFRYX +	Interrupt address clock
OIUAX +	Interrupt cycle
OIUCX +	Interrupt clock
OIURX–	Parity interrupt request
PRNE–	Parity priority out
YDNM +	Memory acknowledgment

6.6 PRIORITY MEMORY ACCESS

Mnemonic	Description
AAxx–	PMA address-buffer bit xx
ABxxS +	PMA address-buffer bit xx (receiver output)
ACK	PMA OR gate
ACKEN	Acknowledgment clock enabled
ACKJ	PMA jumper
ACKx	Channel x acknowledgment
ACKxS	Channel x acknowledgment set
ACOMP	Compatible mode selected
ADDN	Memory cycle completed repowered
ADFx	Data-sequencer bit x
ADxS	Data-sequencer bit x set
ADxx	PMA data-buffer bit xx
ADxxL +	PMA data-buffer bit xx
ADxxL–	PMA data-buffer bit xx repowered
ADxxS	Memory data-buffer bit xx (receiver output)
AGO	PMA-bus GO delayed and converted
AGOL	GO-latch flip-flop
AGOLS	GO-latch flip-flop set
AGOS	GO pulse set
AHSM	Not used
AKEN	Acknowledgment enabled
AKENR	Acknowledgment enabled reset
AKMD	Memory data-buffer clock
AK165	Free-running 165-nanosecond clock
AK41	Free-running 41-nanosecond clock
AK82	Free-running 82-nanosecond clock
AMDCI	Memory-sequencer idle
AMDCx	Memory-sequencer state x
AMDEN	Memory data-bus enabled
AMFx	Memory-sequencer bit-x flip-flop
AM1A1	Memory-sequencer bit 1 OR set
AM1A2	Memory-sequencer bit 1 OR set
AM1E1	Memory-sequencer bit 1 AND set
AM1E2	Memory-sequencer bit 1 AND set
AM1S	Memory-sequencer bit 1 set
APDEN	PMA data-bus enabled
APRIx	Bit-x priority gate
ARDCI	Request-sequencer decoder idle
AREAD	Memory-sequencer output flip-flop
ARFx	Request-sequencer bit x
AROQ	82-nanosecond request flip-flop
AROQS	82-nanosecond request flip-flop set
ARQ	Request present
ARQA	Request flip-flop AND reset
ARQR	82-nanosecond request flip-flop reset
ARQx	Request bit-x flip-flop
ARQxS	Request bit-x (receiver output)
ARST	PMA reset
ARSTA	PMA reset (receiver output)
ART1S	Countdown-counter bit 1 set
ARTx	Countdown-counter bit x
AR123R–	Request-sequencer bits 1, 2, and 3 OR reset
ARxR	Request-sequencer bit-x flip-flop reset
ARxS	Request-sequencer bit-x flip-flop set
ASTRT +	Start flip-flop (not used)
ATGO	Countdown completed
CPMAST	Start PMA controllers (not used)
DBxx	Data-buffer bit xx
FINTE	Power-failure/restart active
GO	PMA controller acknowledgment
GOA	PMA controller acknowledgment (receiver output)
GOAD	GOA + delayed 30 nanoseconds
HOG	PMA continuous-request to memory
HOGA +	PMA continuous-request inverted
HOGA–	PMA continuous-request (receiver output)
LK82 +	Common-logic 82-nanosecond clock
MAKO	Memory-priority acknowledgment
MFCO–	Memory free-running clock
MHGY	PMA to HOG memory if PF/R inactive
MOCLK–	Common-logic free-running clock
MRQY	Memory-port priority request
MWLY	Memory-bus read/write left byte
MWRY	Memory-bus read/write right byte
MYABxx	Memory-address bit xx
MYDBxx	Memory data-bus bit xx
NHCO +	Common-logic 165-nanosecond clock
ORQM	165-nanosecond request flip-flop
PAxx	PMA-bus address bit xx
PDxx	PMA-bus data bit xx
PMRS	PMA-controller to reset PMA
READ	PMA-bus output
READA	PMA-bus output (receiver output)
REQx	PMA request x
START	PMA-bus start (not used)
SYRT-P	PMA-bus reset (controller cleared)
YDNM	Memory cycle completed